Characterisation & Optimisation of Computational Functional Blocks for ATM Switches in GaAs MESFET Technology

Eric Chu B.Sc., B.E.(Hons)

A thesis submitted to the Faculty of Engineering for the Degree of Master of Engineering Science

Centre for Gallium Arsenide VLSI Technology
Department of Electrical and Electronic Engineering
The University of Adelaide
South Australia

May 1994

Contents

Abstract

Broadband Integrated Services Digital Network (B-ISDN) is thought to be the next step in the evolution of communication networks. Amongst all possible realisations, Asynchronous Transfer Mode (ATM) has shown the most promise. Currently, many telephone companies in the world are already providing ISDN services to their customers. CCITT has proposed two data rates for ISDN: 155Mb/s and the 622Mb/s. In a national Danish project, an experimental ATM switch fabric of size *1024 x 1024* operating at up to 622Mb/s will be constructed using only three different types of chips in a commercial Gallium Arsenide MESFET process. It uses a Benes type network architecture together with a concept of bit-rate conversion to improve performance. This switch fabric is designed to utilise only *2 x 2* switching elements (which have a maximum input data rate of 4.8Gb/s). The main objective in this thesis is to design the control logic of a buffer chip and highlight some important issues on GaAs VLSI design in contrast to conventional Silicon.

The first step in logic design is to select and optimise the appropriate logic families. A "mixed" logic approach based on the DCFL family is presented which is designed for high speed, low power and high temperature operations.

The control logic of a buffer chip is described in this thesis and the entire chip is currently being fabricated in the Thomson Composants Microondes (TCS) 0.8μm Self-Aligned Gate (SAGA) E/D GaAs process. An isochronous clocking strategy is used due to the large geometry of the chip. The input section runs at a maximum frequency of 600MHz while other sections run at half that frequency. The control block is realised as three modules: an input control, a buffer manager, and an output control module. The entire chip contains over 70,000 transistors and has a total area of 26.1mm^2. Simulations at maximum frequency predict a power consumption of less than 1W with a 1.5V power supply at

125°C. This demonstrates a superiority in power-delay product of GaAs over Silicon at high operating frequencies.

Declaration

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis made, when deposited in the University Library, being available for loan and photocopying.

SIGNED: ..... DATE: .....12/10/94.....

Acknowledgments

The author is indebted to his supervisor **Assoc. Prof. Kamran Eshraghian** for all his invaluable and untiring guidance and support given during the entire research period as well as the exchange opportunity to work in Denmark. Furthermore, the helpful discussions given by **Dr. Ken Sarkies** and **Mr. Michael Liebelt** is very much appreciated in the area of telecommunications and digital logic design. The author would also like to thank his colleagues **Mr. Andrew Beaumont-Smith** and **Mr. Warren Marwood** for sharing their expertise in design tools as well as all the useful advice and discussions for the long hours in the laboratory where the deadline for chip submission was near, **Mr. Jonathan Main** for his help in understanding many fundamental concepts in B-ISDN and ATM switching, **Mrs. Song Cui** and **Mr. Alireza Moini** for their collaboration during the first three months of the research period in completing a survey as well as the optimisation and comparison of various logic families in GaAs. **Mr. Tarmo Rohtla** is to be acknowledged for his assistance and support in computing facilities. Also, the assistance provided by many other staff and postgraduates in this university is very much appreciated.

During his stay in Denmark, the author would like to express his most sincere thanks to Danish researcher **Mr. Jens Jakobsen** and the entire VLSI design team in **Jydsk Telefon**, Århus for their support as well as providing a very nice and friendly working environment despite the author's poor Danish. The author would also like to acknowledge **Mr. Preben Hvas** and **Mr. Poul Gødsvang** for their advice, guidance and much useful information regarding the specifications of this project and **Mr. Finn Barrett** for design tools and computing facilities support.

Regarding the preparation of this thesis, the author is most thankful to **Assoc. Prof. Kamran Eshraghian, Dr. Ken Sarkies, Mr. Michael Liebelt, Dr. Neil Burgess, Mr.**

Jens Jakobsen, Mr. Andrew Blanksby, Mr. Andrew Beaumont-Smith, Mrs. Song Cui, Mr. Shannon Morton and **Mr. Michael Mcgeever** for proof-reading this thesis and many constructive feedback and suggestions. Otherwise, the submission date would definitely be postponed.

Last but not least, the author would like to acknowledge his ex-girlfriend **Miss Catherine Law** for her ever-lasting spiritual support. It was this support that drove him to complete this research. Many thanks to her indeed.

Abbreviations

ATM:	Asynchronous Transfer Mode
B-ISDN:	Broadband-Integrated Services Digital Network
CAD:	Computer Aided Design
CCITT:	International Consultative Committee on Telephone and Telegraph
DCFL:	Direct Coupled FET Logic
DFET:	Depletion-type Field Effect Transistor
EFET:	Enhancement-type Field Effect Transistor
FIFO:	First In First Out
GaAs:	Gallium Arsenide
Gb/s	Giga-bit per second
HEC:	Header Error Control
IC:	Integrated Circuit
ISDN:	Integrated Services Digital Network
kb/s	Kilo-bit per second
Mb/s	Mega-bit per second
MESFET:	Metal Semi-conductor Field Effect Transistor
MOSFET:	Metal Oxide Semi-conductor Field Effect Transistor
N-ISDN:	Narrowband-Integrated Services Digital Network
SBFL:	Super Buffer FET Logic
SDCFL:	Source-follower Direct Coupled FET Logic
UBFL:	Ultra Buffer FET Logic
VHDL:	Very high speed integrated circuit Hardware Descriptive Language
VLSI:	Very Large Scale Integration

Chapter 1. Introduction

VLSI technology and communications are the two most rapidly growing areas in Electrical Engineering. The combination of both enables the provision of an entirely new spectrum of services to become available to everyday customers. Today, different services use different and independent networks (eg. the telephone network for voice services, the InterNet for data communications and CableTV for TV/Video services) which are expensive to maintain, incompatible and lack capability for sharing traffic between them. This leads to the demand for a single integrated network which is flexible and efficient enough to replace all current networks. This is what Broadband ISDN aims to realise.

For a network to support data rate in the order of multi-gigabit per second, the technology will have a significant impact on the hardware realisation. Currently, the semi-conductor industry has been dominated by Silicon technology. However, as the speed of operation keeps increasing, the limit of Silicon technology will soon be reached. Gallium Arsenide (GaAs), on the other hand, has a superior power-delay product compared to Silicon technology at high operating speed. A brief comparison between Silicon and GaAs as a semi-conductor will also be discussed in this chapter.

1.1 B-ISDN and ATM

ISDN is the mainstream in the next generation of communication networks. It is designed to integrate all current networks into a single one. Currently, the detailed specification of ISDN is still under investigation [I.121]. However, two draft specifications expected [DEPRYCKER] are listed below:

- The cell loss probability for high priority cells should be comparable to that of optical fibre, that is, in the order of 10^{-12} and,
- Cell re-ordering is not allowed to reduce cost

ISDN comes in two flavours: Narrowband ISDN (N-ISDN) and Broadband ISDN (B-ISDN) [STALLINGS], [DEPRYCKER]. N-ISDN can be thought as the transition between the current telephone network and B-ISDN. In N-ISDN, voice and data pass over separate networks but appear to the customer as a single network with a common access interface. It uses the same circuit switching technique as conventional telephone network to transmit voice and data at a single 64kb/s rate. For this reason, N-ISDN is also known as the 64kb/s ISDN. N-ISDN can also be thought of as several separate low speed data networks switched by software and hence, has no possible expansion to video or other high data rate services apart from very primitive ones. B-ISDN, on the other hand, uses an entirely different switching technique called Asynchronous Transfer Mode (ATM) to increase its efficiency and flexibility so that other high bandwidth services can be supported. Currently, two standard transmission rates are specified [I.121]: the 155Mb/s and the 622Mb/s B-ISDN. Higher transmission rates can be expected in the near future. A table of comparison between conventional circuit switched and ATM networks is shown in Table 1.1 [DEPRYCKER], [PARTRIDGE].

The main advantages of B-ISDN are discussed below:

- Flexible and Future Safe

  An integrated network which supports different types of services will need to adapt itself to different traffic behaviour (eg. bursty traffic for data services, smooth traffic for voice services and intermediate traffic for video services).

- Efficient in Resource Use

  Since all available resources can be shared between all different services, an optimal statistical resource sharing can be achieved.

- Less Expensive

  Since only one network is required to be designed, manufactured and maintained, the overall designing, manufacturing, operating and maintenance costs will be lower.

	Transmission Method	
	Circuit Switching	ATM
Multiplexing Method	• Time Division Multiplexing (TDM) • Time is divided into frames • Frames are divided into slots	• Time Division Multiplexing (TDM) • Time is divided into cell slots
Transmission Rate	• Low speed (in the order of kb/s)	• High to very high speed (155Mb/s or 622Mb/s)
Types of Services Supported	• Typically voice and some low bandwidth data services	• All current services as well as future services which are not yet fully defined
Calls Transmission	• Assigned to a particular slot in each frame	• May transmit in any available cell slot
Delay Contributions	• Propagation delay on transmission links • Processing delay in the switches	• Transmission delay • Packetization and depacketization delay • Switching delay
Signalling	• Usually done in one slot in each frame	• Done in OAM cells
Advantages	• Good for fixed bandwidth constant bit-rate calls (eg. voice services)	• Flexible and future safe • Efficient use of bandwidth for both constant and variable bit rate services • Less expensive
Disadvantages	• Inflexible for calls of different bandwidth to basic rate • Inefficient for variable bit rate calls	• Larger overheads

Table 1.1: Comparison between circuit switching and ATM networks

ATM can be thought to be a very general protocol which is capable of providing a wide range of different services. Some characteristics of ATM networks are [DEPRYCKER]:

- Fully Integrated for Low Cost

  Since only one network is required for all kinds of service, the operation and maintenance costs can be reduced.

- Supports All Known Services

  Due to the flexibility of ATM, all known services as well as the ones which have not yet been fully defined can be supported.

- Connection Oriented

  This refers to the fact that no cells can be sent down from the user-terminal to the network until a logical/virtual connection is established. This allows the network to reserve and check if there is statistically enough resources available to support the connection. If there are insufficient resources available, the connection is refused.

- High Speed

  The high speed operation in ATM networks is achieved by hardware switching as well as using small and fixed length packets which have the properties of:

  - Reduced header functionality

    The ATM header has a very limited function to enable fast processing in the network. Its main function lies in the identification of the virtual connection by an identifier which is selected at call set-up and guarantees a proper routing of each cell in the network. In addition, it allows an easy multiplexing of different virtual connections over a single link.

Due to the limited functionality in the header, the implementation of the header processing in the ATM nodes is simple and can be done at very high speeds (eg. 155Mb/s and 622Mb/s). This results in very low processing and queuing delays.

- No error protection or flow control on a link-by-link basis

  In contrast to packet switching, no re-transmission of lost cells takes place in a link of the connection. This is due to the fact that very high quality links are used in a network so a low bit error rate is achieved. Hence, no error control is required.

  Flow control will also not be supported in ATM networks. Proper resource allocation and queue dimensioning in the network ensure a tolerable number of queue overflows which cause packet loss. Values of packet loss probability of 10^{-8} down to 10^{-12} are intended.

- Relatively small information field length

  In order to reduce the internal buffers in the switching nodes, and to limit the queuing delays in those buffers, the information field length is kept relatively small. Indeed, small buffers guarantee a small delay and delay jitter as required by real time services.

In this thesis, all designs are required to support both 155Mb/s and 622Mb/s transmission rates for compatibility reasons.

1.2 Benes Network and Bit-Rate Conversion

The Benes network was developed by [BENES] in the 1960's and is reported in [TOBAGI] to be the smallest interconnection network for which all permutation patterns are realisable. It is an architecture originally developed for circuit switched networks for building a large non-blocking switch by smaller switching elements. A Benes network can be constructed by two Banyan networks [GOKE&L] arranged in a mirror image configuration which is symmetrical about the centre stage. To realise an $N \times N$ switch, it requires:

$$\frac{N}{M} \cdot \left(2 \cdot \frac{\log(N)}{\log(M)} - 1 \right)$$

$M \times M$ switching elements.

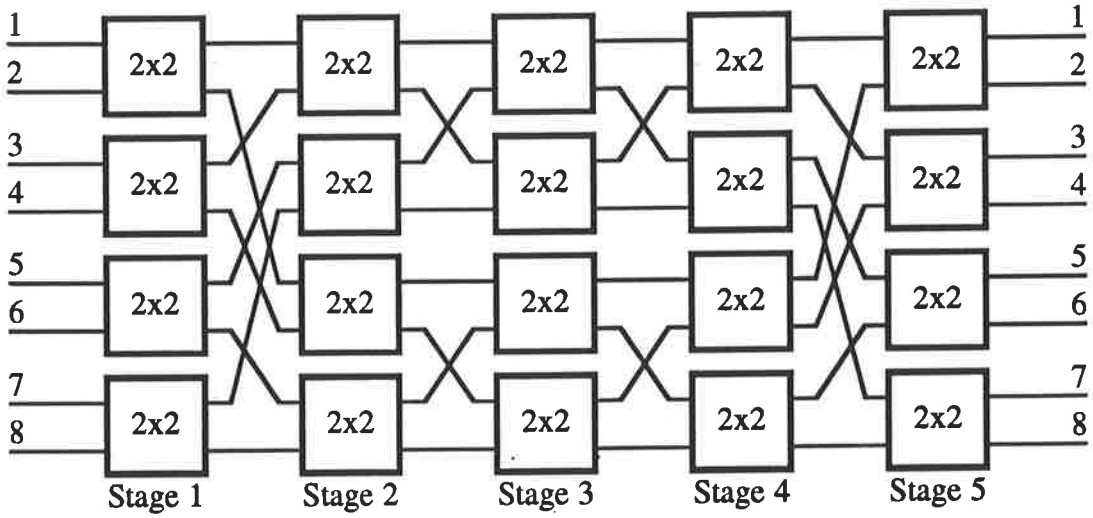


Figure 1.1: An *8 x 8* switch constructed by *2 x 2* switching elements as a Benes network

An *8 x 8* switch can be constructed by *2 x 2* switching elements as shown in Figure 1.1. In circuit switching, the Benes network is known as a "Rearrangeably Non-blocking" switch which means that the connections in the circuit switch can always be rearranged in a way so that a desired connection can be made. However, there is no equivalent for ATM. In ATM, the switch is used as a distribution-routing switch in which cells are randomly

scattered over the centre stage to break up long term congestion in the routing stage due to lots of cells converging onto one link. Cell loss can still occur but it is shared evenly by different connections and is considerably less than that for a Banyan network. In Figure 1.2, different paths to route an incoming cell from input 1 to output 8 are shown.

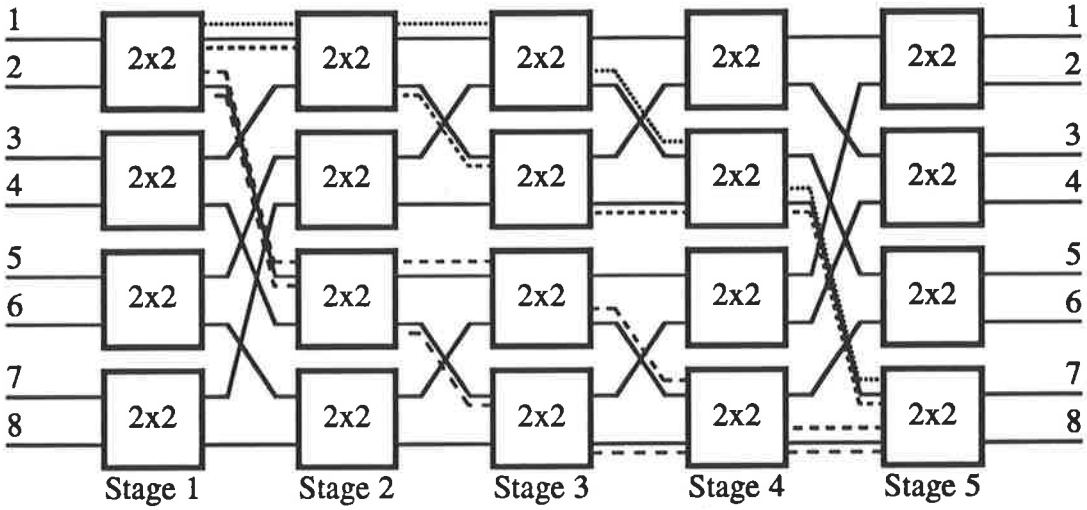


Figure 1.2: Different paths to route an incoming cell from input 1 to output 8

In this project, a "call level randomising" routing strategy is used for the Benes network in which each call is given a random path through the first half of the network and routed to the desired output in the second half. Cell re-sequencing is not necessary since all cells belong to the same call take the same path through the network. However, call set-up blocking still occurs but can be minimised by employing the bit rate conversion technique.

The concept of bit rate conversion is to increase the data rate by a factor of k at the inputs before entering the switching elements. It is then reduced back to its original speed before leaving the output. This effectively reduces the switch size by a factor of k and hence the number of switching elements and the delay. It can be shown that the power dissipation can also be reduced. However, this is done at the expense of very high speed switching hardware.

1.3 Gallium Arsenide as a Semi-Conductor

Gallium Arsenide was first discovered in 1926. However, its potential as a high speed semi-conductor was not realised until the 60's [ESHRAGHIAN]. The advantages of GaAs over Si as a semi-conductor are discussed below: [BUSHEHRI], [ESHRAGHIAN], [FIRSTENBERG], [HSU], [SIMONS], [SZE]

(i) At normal doping levels the saturated carrier drift velocity for GaAs and Silicon are 1.4 x 10^7cm/s and 1 x 10^7cm/s respectively. However, the drift velocity of GaAs reaches a peak at an electric field strength of around 0.35V/μm. As the electric field strength increases further, the drift velocity reduces and approaches the saturated velocity. On the other hand, the drift velocity of Silicon increases with increasing electric field strength while the saturated velocity is obtained at an electric field strength of about four times that of GaAs. As a result, up to 70% reduction in power dissipation can be achieved in GaAs over the fastest Silicon technology such as Emitter Coupled Logic (ECL).

(ii) The electron mobility of GaAs is about six to seven times higher than that of Silicon. Consequently, GaAs MESFETs with typical gate lengths of 0.5μm to 1μm can achieve transit times as short as 10ps to 15ps. This corresponds to a current gain-bandwidth product in the range of 15GHz to 25GHz, which is a factor of three to five times higher than Silicon.

(iii) Due to the absence of a gate oxide layer to trap charges, GaAs devices are more radiation resistant than Silicon ones. This has a significant impact on space applications where radiation is a major concern.

(iv) A large bandgap offers GaAs semi-insulating properties. A high resistivity in the range of 10^7—10^9Ωcm at room temperature is another advantage for high performance

devices. It does not only minimise the parasitic capacitances but also reduce the leakage current between devices on the same substrate.

(v) A wider operating temperature for GaAs devices is possible due to the larger bandgap. Typical operating temperature varies over the range from -200°C to +200°C.

(vi) Schottky barriers can be realised on GaAs with a large variety of metal such as Aluminium, Platinum and Titanium. This leads to high quality Schottky junctions with excellent ideality (less than 1) and low reverse currents (< $1A/cm^2$).

(vii) The direct bandgap of GaAs allows efficient radiative recombination of electrons and holes. This means the forward biased p-n junctions can be used as light emitters. Thus, an efficient integration of electrical and optical function is possible.

A table summarising the electrical properties between GaAs and Silicon is shown in Table 1.4 [GLOANEC].

Physical Properties	GaAs	Si
Electron mobility (cm^2/Vs)	5000	800
Maximum electron drift velocity (cm/s)	2×10^7	1×10^7
Hole mobility (cm^2/Vs)	250	350
Energy gap (eV)	1.43	1.12
Type of gap	Direct	Indirect
Density of states in conduction band (cm^3)	5×10^{17}	3×10^{19}
Maximum resistivity (Ωcm)	10^9	10^5
Minority carrier life time (s)	10^{-8}	10^{-3}
Breakdown field (V/cm)	4×10^5	3×10^5
Schottky barrier height (V)	0.7-0.8	0.4-0.6

Table 1.4: Comparison of electrical properties between GaAs and Si

Despite all the desirable properties that GaAs may possess, its disadvantages lie mainly in the device physics [LONG&BUTNER], [NUNEZ] which limit the yield and result in a high fabrication cost. The disadvantages of GaAs are discussed as follows:

(i) GaAs wafers have a large density of dislocations in the crystal lattice structure. This coupled with the inadequacy and brittleness of the material together with the extra difficulty in controlling the doping and threshold voltage over the wafer results in a lower yield than Silicon. As a result, GaAs ICs must be smaller in area and have a smaller transistor count. Furthermore, the fabrication cost for GaAs ICs is approximately two orders of magnitude higher than Silicon of the same technology.

(ii) The lack of a gate oxide (which isolates the gate metal and the underlying conductive channel) in GaAs makes the Schottky junction easily forward-biased. As a result, a large gate current can flow through and limits the gate-to-source voltage to the range of 0.7—0.8V depending on the type of gate metal used. Consequently, it is more difficult for GaAs devices to match the operation conditions of existing Silicon ICs.

(iii) The problems associated with the reduction of drain current in a MESFET or heterojunction field effect transistor (HFET) by the presence of other nearby neighbouring FETs are known as the backgating or sidegating effect. This effect is mainly caused by the capacitive coupling of the channel of a MESFET to the floating substrate. As a result, the substrate can modulate the drain current of the MESFET as a backgate, and adjacent devices as sidegates. The remedy is to place transistors further away from each other. Unfortunately, this conflicts with the mentality to put devices in close proximity for good matching as well as achieving a high packing density in a VLSI environment.

From the above discussions, it seems unlikely for GaAs to replace Silicon as a semiconductor. However, GaAs can be used as a fast front-end interface (eg. interfacing with

optical fibres) that processes high speed serial data to produce lower rate parallel substreams which are further processed by Silicon subsystems.

1.4 Scope of This Thesis

This thesis is intended to address the design details regarding the control logic within a buffer chip, which is part of a chip set for the realisation of a high bandwidth packet switch fabric in GaAs MESFET technology. An outline of the following chapters in the thesis is presented below:

Chapter 2 describes the choice of logic families and their optimisation. A "mixed" logic design approach based on DCFL is used in the realisation of the control logic. This work has been published in the 11th NorChip seminar held in Trondheim, Norway on November 1993. The actual paper is attached in Appendix D.

Chapter 3 describes the overall system specifications for the entire *1024 x 1024* switch fabric. Issues such as the internal cell format, the systems architecture and the required performance are addressed.

The logic layout style and the design tools used throughout the project are described in Chapter 4. The "ring notation" layout style is employed to minimise the coupling between high speed signals and to achieve a high packing density. A design flow diagram is presented as a guideline for designing and verifying high complexity VLSI circuits.

Chapter 5 describes the basic primitives used in the construction of the buffer chip. This library is the same as the one used in the VHDL descriptions to achieve a one-to-one correspondence between the structural descriptions and the transistor netlists generated from the layout.

Chapter 6 describes the author's design of the control logic. It is realised as three main modules: an input control, a buffer manager and an output control. The modules operate at different speeds and communicate asynchronously. All circuit and timing diagrams, layouts and simulation results are included for completeness. This chapter constitutes the main context of this thesis. Other modules in the buffer chip were designed by Mr. Jens Jakobsen at Jydsk Telefon, Denmark.

Finally, a conclusion about this research is presented in Chapter 7. A note about the future work on this project is also given.

Chapter 2. Choice of Technology & Optimisation

This chapter describes the logic families used in the switch fabric design. Initially, a survey of the logic families is carried out. Direct Coupled FET Logic (**DCFL**) and three other static "normally-off" families (**SDCFL, SBFL** and **UBFL**) based on it are chosen due to their simplicity, compatibility in signal levels, power consumption and speed requirement. A "mixed" logic approach based on these four families is used to realise the chip sets for the switch fabric. The work regarding the optimisation and comparison of these four families has been published in [CHU&JAKOBSEN] and is included in Appendix D.

2.1 Logic Families in GaAs

There exists a large number of logic families in the GaAs MESFET technology. They can be classified into two main categories: the "Normally-On" and the "Normally-Off" logic families [ESHRAGHIAN].

"Normally-on" logic utilises depletion type MESFETs which are normally "ON" devices and when used as switching elements they are required to be turned off. This class of logic includes the following approaches:

- Buffered FET Logic (**BFL**) [VANTUYL1], [VANTUYL2];
- Capacitively Coupled Domino Logic (**CCDL**) [HOE&SALAMA];
- Capacitor Coupled FET Logic (**CCFL**) [MELLOR], [WELBOURN];
- Capacitor Diode FET Logic (**CDFL**) [EDEN1];
- Feed-Forward Static Logic (**FFSL**) [ESHRAGHIAN];
- Inverted Common Drain Logic (**ICDL**) [ABDEL,R&Y];

• Schottky Diode FET Logic (**SDFL**) [EDEN,L,L,W&Z], [EDEN2], [HELIX,J,C&S], [LONG];

• Source Coupled FET Logic (**SCFL**) [KATSY], [VU&PECZALSKI] and

• Unbuffered FET Logic (**UFL**) [BARNA&L].

"Normally-on" logic requires two power supplies (both VDD and VSS) and extra level shifting diodes. This implies a higher complexity and area consumption and hence is not suitable for VLSI implementation.

"Normally-Off" logic, on the other hand, utilises enhancement type MESFETs which are normally "OFF" devices and when used as switching elements they are required to be turned on. This class of logic includes the following approaches:

• Direct Coupled FET Logic (**DCFL**) [BOSCH], [CHU&JAKOBSEN], [ISHIKAWA], [PECZALSKI], [SUYAMA];

• Feedback FET Logic (**FBFL**) [FULKERSON];

• FET FET Logic (**FFL**) [LARUE,W&C];

• Junction FET Logic (**JFL**) [ZULEEG,N&T];

• Pseudo Current Mode Logic (**PCML**) [DUNCAN,S&S];

• Quasi-FET Logic (**QFL**) [NUZILLAT];

• Super Buffer FET Logic (**SBFL**) [CHU&JAKOBSEN], [LONG&BUTNER], [NAKAMURA];

• Source-follower Direct Coupled FET Logic (**SDCFL**) [CHU&JAKOBSEN], [DAVENPORT], [ESHRAGHIAN92];

• Source Follower FET Logic (**SFFL**) [ESH,C,M&C], [ESH,B,S,L,B&B];

• Two-phase Dynamic FET Logic (**TDFL**) [NARY&LONG] and

• Ultra Buffer FET Logic (**UBFL**) [CHU&JAKOBSEN].

As well as the above logic classes, technology incorporating pass transistors has been developed for use in digital logic design. "Differential Pass Transistor Logic" (**DPTL**) has been reported in [PASTERNAK&S] to take advantage of the pass transistor structure without sacrificing important design parameters such as noise margins by using buffers at the output. However, it requires differential inputs and frequent buffering which makes its use limited. While dynamic logic has desirable properties of being simple and low power, they are highly sensitive to process variations and operating conditions. These coupled with the fact that they possess a minimum frequency of operation makes their use limited. As a result, only static "normally-off" logic are considered here due to speed requirements (the chip must work at a frequency range from 20MHz to 600MHz). Amongst all candidates, DCFL has shown the most promise. It has the desirable properties of being simple as well as having a constant and low power dissipation. However, the main drawback lies in its low noise margins and weak output driving capabilities. A mixed logic approach using pure NOR structures based on DCFL, SDCFL, SBFL and UBFL has been developed in [CHU&JAKOBSEN] (see Appendix D) to achieve high speed, low power, low noise and high packing density designs. The four logic families are introduced in section 2.1.1 to 2.1.4.

2.1.1 Direct Coupled FET Logic (DCFL)

DCFL is the simplest logic family in the GaAs MESFET technology. A DCFL inverter is shown in Figure 2.1(a). Its structure closely resembles the nMOS structure in Silicon design except for the presence of the Schottky diode at the gate of the EFETs which clamps the steady state high voltage level to one diode drop (approximately 0.7V) instead of the full swing to VDD. As a result, the pull-up DFET always stays in saturation. The operation of a DCFL inverter can be describe as follows: When the input is a logic "low", the pull-down EFET is cut-off. Its equivalent circuit is shown in Figure 2.1(b). On the other hand, when the input is a logic "high", the pull-down EFET operates in the linear region and thus behaves as a voltage controlled resistor (VCR) and the equivalent circuit is shown in

Figure 2.1(c). From its operation, it can be seen that DCFL gates have a large delay when the output load is large. This is due to the limited current available to pull up the output load, hence a large rise-time. Also, the noise margin in DCFL is low since the output low level depends upon the on-resistance of the EFET. As a result, a high noise margin can only be achieved by increasing the width of the EFET. However, as shown in Appendix D, this reduces the speed of the gate. Consequently, there is a trade-off between the speed and the noise margins on the sizing of DCFL gates.

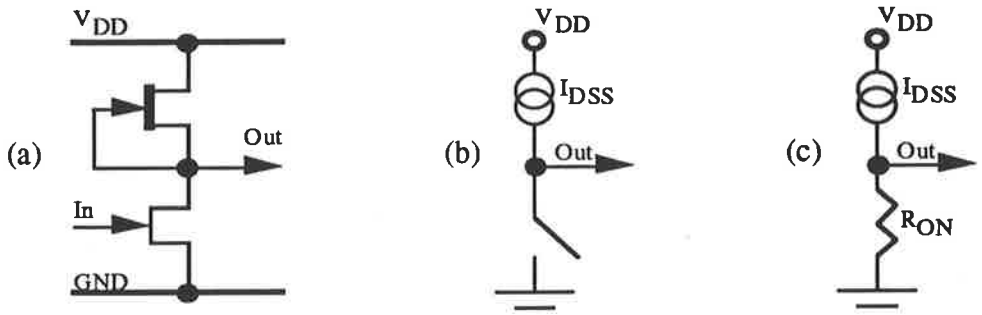


Figure 2.1: (a) DCFL inverter

(b) Equivalent circuit when the input is low and

(c) Equivalent circuit when the input is high

Besides inverters, DCFL also supports nor functions which can be realised by simply adding extra parallel pull-down EFETs. The simplicity of DCFL together with the fact that it has a constant and low power dissipation makes it an excellent candidate for VLSI implementation since a high packing density, quiet power bus and low switching energies can be achieved. Moreover, DCFL has been reported to have the smallest power-delay product over any other static logic families in GaAs [ESHRAGHIAN]. However, the main drawback in DCFL lies in its low noise margins (which makes it difficult to realise large fan-in nor gates and a stringent process requirement is imposed) and a weak driving capability. Three other logic families presented in the following sections are designed to increase the noise margin and/or the driving capability of DCFL by means of output buffering.

2.1.2 Source-follower Direct Coupled FET Logic (SDCFL)

As its name suggests, SDCFL improves the performance of DCFL by appending a source follower at its output. A basic SDCFL inverter is shown in Figure 2.2(a). With reference to Figure 2.2, the operation of an SDCFL inverter can be described as follows:

When the input is a logic "low", J2 is cut-off while J1 operates in the linear region and hence the internal node is pulled high to 2 diode drops. However, the difference between the gate-to-source voltage (V_{gs}) of J3 and its threshold voltage (≈200mV) is less than its drain-to-source voltage (V_{ds}). As a result, J3 saturates while J4 operates in the linear region. The equivalent circuit for a logic "low" input is shown in Figure 2.2(b). On the other hand, when the input is a logic "high", J1 saturates while J2 operates in the linear region and behaves as a voltage controlled resistor (VCR). Consequently, the internal node is pulled down and hence J3 is cut-off while J4 operates in the linear region. The equivalent circuit is shown in Figure 2.2(c). From its operation, it can be seen that SDCFL gates will have better noise margins than DCFL since the buffer stage improves the output low level. However, it has a large fall-time when the output load is large. This is due to the fact that SDCFL has a large pull-down RC time constant since the pull-down DFET has a large on-resistance. One way to improve the fall-time in SDCFL is to tie a small negative voltage (eg. -200mV) instead of ground to the pull-down DFET at the buffer stage so that it stays in saturation for a longer period. It also gives a lower output low voltage which in turn increases the noise margin. However, due to practical difficulties and complexity issues, it is not considered here.

In additional to inverters and nor gates, SDCFL also supports Or-And-Invert (OAI) structures which is illustrated in Figure 2.3. In OAI structures, the output of two SDCFL gates are connected together to form an "or" operation. Note that the total width of the pull-down DFET is halved in order to maintain proper output high level. OAI structures enables very high speed operation as there is no gate delay involved and a very compact layout can be achieved.

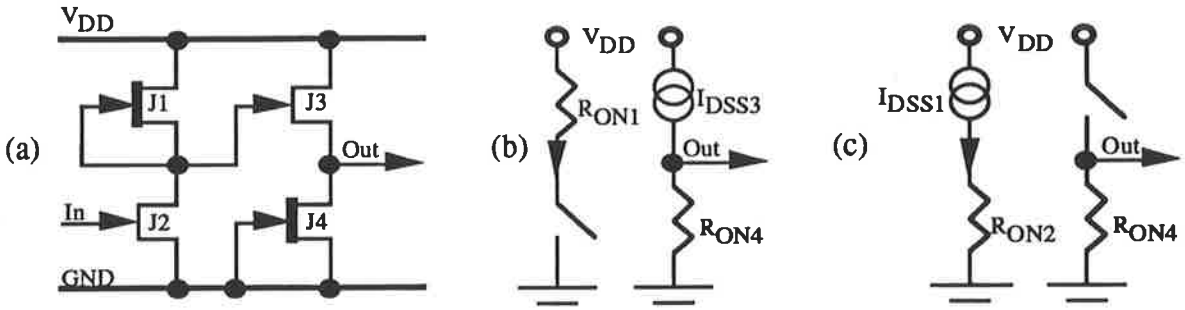


Figure 2.2: (a) SDCFL inverter

(b) Equivalent circuit when the input is low

(c) Equivalent circuit when the input is high

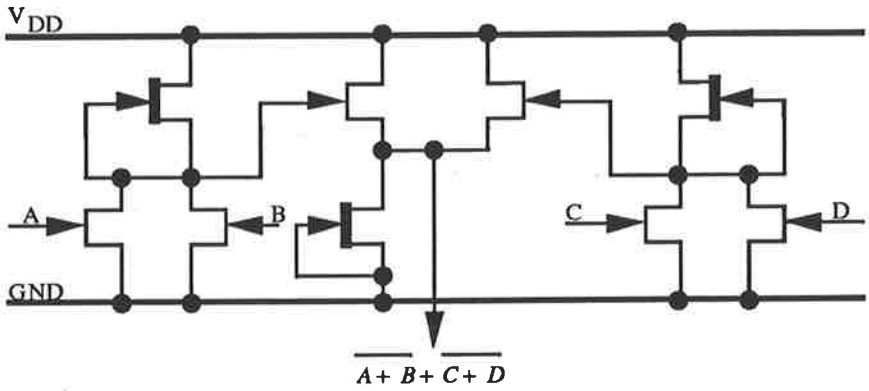


Figure 2.3: Or-And-Invert (OAI) structure in SDCFL

The realisation of nor gates with higher fan-ins than DCFL is possible since SDCFL has higher noise margins. Moreover, the source follower also increases the output driving capability. However, this is at the expense of a higher power dissipation and complexity. Furthermore, the power dissipation of SDCFL gates depends on the output logic state as the steady-state output high level is maintained by the pull-up EFET (J3) at the source follower stage which causes noise to be injected into the power busses whenever the gate switches.

2.1.3 Super Buffer FET Logic (SBFL)

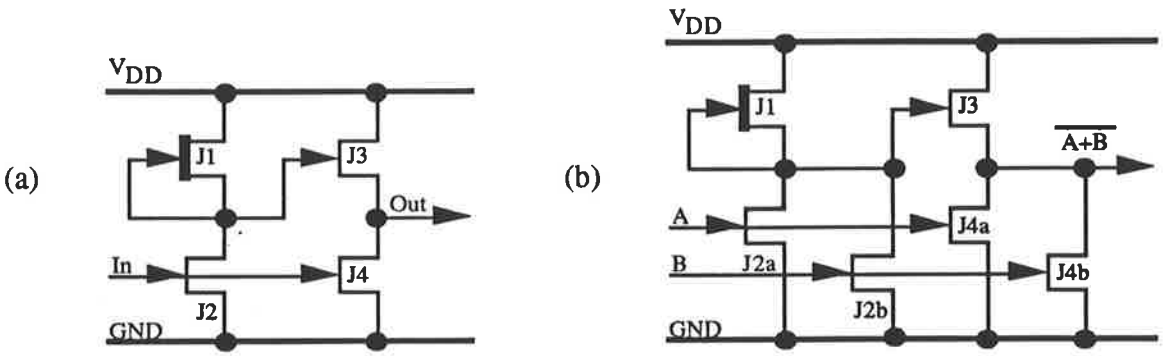


Figure 2.4: SBFL inverter and 2-input nor gate

An SBFL inverter and 2-input nor gate is shown in Figure 2.4. Similar to SDCFL, SBFL enhances the performance of DCFL by appending a push/pull super buffer at its output. The operation of an SBFL inverter can be described as follows:

When the input is a logic "low", J1 operates in the linear region while J2 is cut-off and the output of the first stage is pulled high to 2 diode drops. However, the drain-to-source voltage (V_{ds}) of J3 is greater than the difference between its gate-to source voltage (V_{gs}) and the threshold (≈200mV). As a result, J3 saturates while J4 is cut-off. The equivalent circuit for a logic "low" input is shown in Figure 2.5(b). On the other hand, when the input is a logic "high", J1 saturates while J2 operates in the linear region and behaves as a voltage controlled resistor (VCR). Consequently, the internal node is pulled down and hence J3 is cut-off while J4 operates in the linear region. The equivalent circuit is shown in Figure 2.5(c). From its operation, it can be seen that SBFL have both short rise-times (due to the strong pull-up EFET J3) and fall-times (due to the small RC time constant as the resistance of the pull-down EFET J4 is low). However, it has dynamic problems: Consider the case where the input is initially a logic "low", J3 is turned on while J4 is cut-off. A positive transition of the input will turn J4 on while J3 is conducting. This in turn creates a DC conduction path between VDD and GND [LONG&BUTNER]. Hence, the performance of the neighbouring

logic blocks may be affected. As a result, SBFL should be only be used with a wide power bus.

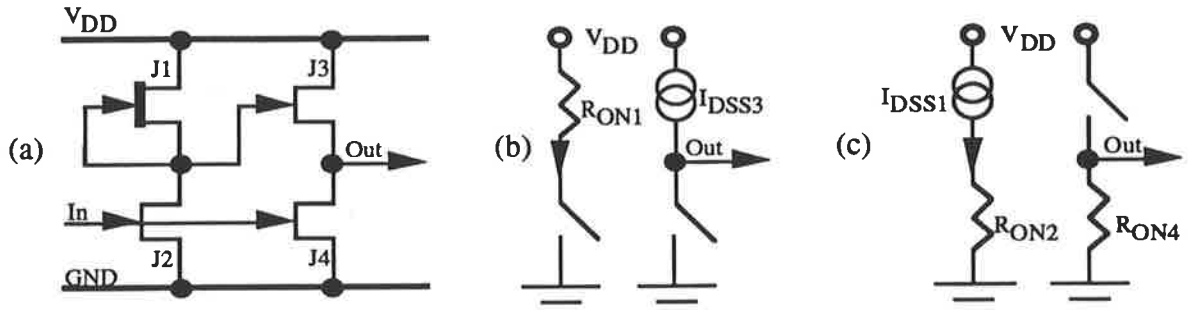


Figure 2.5 (a): SBFL inverter

(b) Equivalent circuit when the input is low

(c) Equivalent circuit when the input is high

SBFL has both higher driving capabilities and noise margins than SDCFL. However, this is at the expense of higher power dissipation. Also, nor structures are considerably more complex than SDCFL as parallel pull-down EFETs have to be included in both the driving and the buffering stages. The problem with DC conduction path between the power bus possess some extra design considerations. Similar to SDCFL, the power dissipation in SBFL also depends on the output logic state as the steady state high level is maintained by the pull-up EFET J3 at the buffering stage.

2.1.4 Ultra Buffer FET Logic (UBFL)

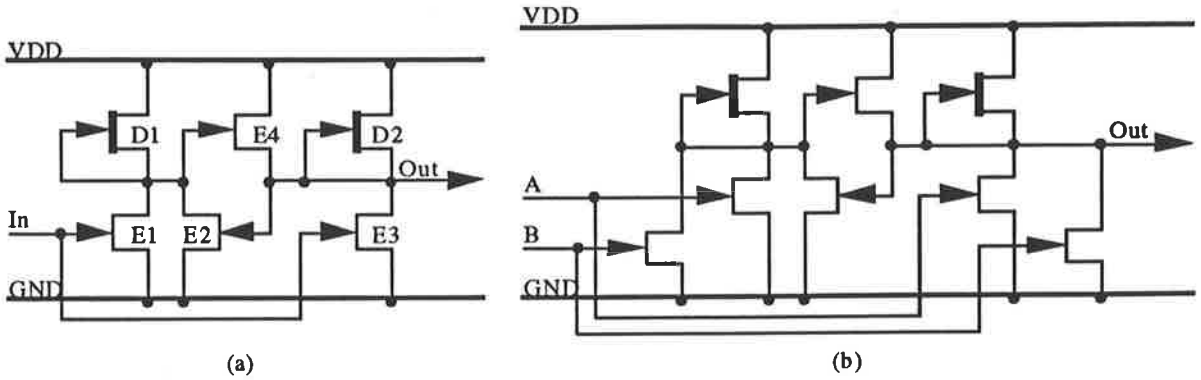


Figure 2.6: (a) UBFL inverter and (b) UBFL 2-input nor gate

A UBFL inverter and a 2-input nor gate are shown in Figure 2.6. UBFL is designed to improve SBFL by maintaining a constant power dissipation irrespective of the steady-state logic states. Hence, the noise injection to the power buses is kept to a minimum. This is achieved by positively feeding the output back to the input stage. With reference to the UBFL inverter in Figure 2.6, its operation can be described as follows:

When input is high, both E2 and E4 are off and the steady state output low voltage is maintained by the DCFL inverter (D2 and E3). As the input voltage starts to decrease, E1 and E3 get cut-off while the internal node pulls up which turns E4 on. Consequently, the output load is pulled up by E4 and thus a short rise-time is achieved. As the output voltage start to increase, E2 gradually gets turned on and reduces the strength of E4. When the output reaches the steady state high level, E4 gets cut-off completely and the output is solely maintained by D2. This is the mechanism which enables UBFL to have constant power dissipation irrespective of the steady-state logic states.

Despite the fact that UBFL has a constant and low power dissipation, its main disadvantage lies in its low noise margins which can be seen from its structure (UBFL is

essentially a 2-input DCFL nor gate follower by a DCFL inverter). Similar to SBFL, it is complex to realise nor gates in UBFL.

2.2 Logic Family Optimisation

The optimisation process is started by noting the operating environment of the chip:

• Power Supply

DCFL can work with a supply voltage as low as 0.9V. However, due to speed and noise injection considerations, the pull-up DFET should operate in saturation at all times. On the other hand, SDCFL, SBFL and UBFL require a higher supply voltage (typically 2 diode drops) for proper logic operation. As a result, a 1.5V supply is chosen to ensure all four logic families will function properly and achieve a reasonable speed.

• Temperature Requirement

A die temperature of 125°C is chosen for the optimisation process to ensure the chip set works under military specifications.

• Backgating Voltage

A backgating (substrate) voltage of 0.6V is chosen for simulation purpose according to the foundry supplied design manual [CMP] as all logics have positive signal levels despite the fact that DFETs are used as pass transistors in the D flip-flop designs (see Chapter 5) and require a negative voltage to be turned off.

• Process Variations

All logics are optimised with their typical parameters to reflect their real-life performance. The effect of process variations is only considered after the optimisation process.

2.2.1 Optimising DCFL

The optimisation process is started by using the Curtice Model [CURTICE] to calculate the appropriate pull-up/pull-down ratio for DCFL inverters. The Curtice Model can be written as:

$$I_{ds} = \beta \cdot \left(\frac{W}{L}\right) \cdot \left(V_{gs} - V_t\right)^2 \cdot \left(1 + \lambda \cdot V_{ds}\right) \cdot \tanh\left(\alpha \cdot V_{ds}\right)$$

where β denotes the transconductance parameter of the device

I_{ds} denotes the drain-to-source current of the device

W denotes the width of the device

L denotes the gate length of the device

V_{gs} denotes the gate-to-source voltage of the device

V_t denotes the threshold voltage of the device

λ denotes the channel-length modulation parameter

V_{ds} denotes the drain-to-source voltage of the device

α denotes the saturation voltage parameter

Consider the case where the inverter voltage V_{inv} is applied to the input of a DCFL inverter such that the output has the same level: both the DFET and the EFET operate in saturation. The current flowing through the EFET and DFET can be written as:

$$I_{dse} = \beta_e \cdot \frac{W_e}{L_e} \cdot \left(V_{in} - V_{te}\right)^2 \cdot \left(1 + \lambda_e \cdot V_o\right) \cdot \tanh\left(\alpha_e \cdot V_o\right)$$

and $I_{dsd} = \beta_d \cdot \frac{W_d}{L_d} \cdot \left(V_{td}\right)^2 \cdot \left[1 + \lambda_d \cdot \left(Vdd - V_o\right)\right] \cdot \tanh\left[\alpha_d \cdot \left(Vdd - V_o\right)\right]$ respectively where Vdd denotes the supply voltage and V_O denotes the output voltage. Note that the subscript "e" denotes parameters associated with the EFET while the subscript "d" denotes parameters associated with the DFET.

By equating the two currents, the following expression can be obtained:

$$\frac{W_d}{W_e} = \frac{\frac{\beta_e}{L_e}\cdot(V_{in}-V_{te})^2\cdot(1+\lambda_e\cdot V_o)\cdot\tanh(\alpha_e\cdot V_o)}{\frac{\beta_d}{L_d}\cdot(V_{td})^2\cdot[1+\lambda_d\cdot(Vdd-V_o)]\cdot\tanh[\alpha_d\cdot(Vdd-Vo)]}$$

which turns out to be around 1/18 by substituting the appropriate parameters for constant gate lengths for both devices. This provides a starting point for the optimisation process. A mathematical treatment of other design parameters of DCFL are given below [CHU&JAKOBSEN]:

• Model Simplification

An abridged device model is required to simplify the mathematical expressions. In the following analysis, second order effects such as voltage saturation (denoted by the voltage saturation parameter α) and channel-length modulation (denoted by the channel-length modulation parameter λ) are ignored. These translate to the following mathematical expressions:

(i) $\tanh(\alpha\cdot V_{ds}) = 1$ for $\alpha\cdot V_{ds}\geq 1$ (saturation)
$\tanh(\alpha\cdot V_{ds}) = \alpha\cdot V_{ds}$ for $\alpha\cdot V_{ds} < 1$ (linear region) and

(ii) $\lambda = 0$

As a result, the drain-to-source current for a MESFET operating in cut-off, saturation and linear region can be re-written as:

$$I_{dse} = \begin{cases} 0 & \text{for } V_{gs} < V_{te} \\ \beta_e\cdot\frac{W_e}{L_e}\cdot(V_{gs}-V_{te})^2 & \text{for } V_{gs} > V_{te};\ V_{ds}\geq\frac{1}{\alpha_e} \\ \alpha_e\cdot\beta_e\cdot\frac{W_e}{L_e}\cdot(V_{gs}-V_{te})^2\cdot V_{ds} & \text{for } V_{gs} > V_{te};\ V_{ds}<\frac{1}{\alpha_e} \end{cases}$$

• Inverter Threshold

The inverter threshold voltage V_{inv} is defined as the voltage such that when it is applied at the input of an inverter, the same output voltage will be obtained. This occurs when the saturation currents for both the EFET and the DFET are equal which can be written as:

$$I_{dse} = \beta_e \cdot \frac{W_e}{L_e} \cdot (V_{inv} - V_{te})^2$$

and

$$I_{dsd} = \beta_d \cdot \left(\frac{W_d}{L_d}\right) \cdot (V_{td})^2$$

respectively. By equating the two currents, V_{inv} can be written as:

$$V_{inv} = V_{te} - \frac{V_{td}}{\sqrt{x}}$$

where $x = \frac{\beta_e \cdot W_e \cdot L_d}{\beta_d \cdot W_d \cdot L_e}$ denotes the pull-up/pull-down ratio

• Inverter Characteristics

Given the inverter threshold V_{inv}, the inverter transfer characteristics can be determined as:

$$V_{out} = \begin{cases} V_{OH} & \text{for } V_{in} << V_{inv} \\ \dfrac{V_{td}^2}{x \cdot \alpha_e \cdot (V_{in} - V_{te})^2} & \text{for } V_{in} >> V_{inv} \end{cases}$$

Note that this is only an approximate model of the inverter's behaviour and is not accurate when the input voltage V_{in} is in the vicinity of V_{inv}.

• Noise Margin

Noise margin is a measure of the noise tolerance of the circuit. Many different definitions of noise margins have been reported in [HILL], [LONG&BUTNER], [WESTE&E] and it is shown in [LOHS,S&DEG] that they have the same mathematical equivalence. Amongst various definitions, the "Intrinsic" method as described in [LONG&BUTNER] is used here due to the ease of measurements. The test structure for the measurement is shown in Figure 2.7.

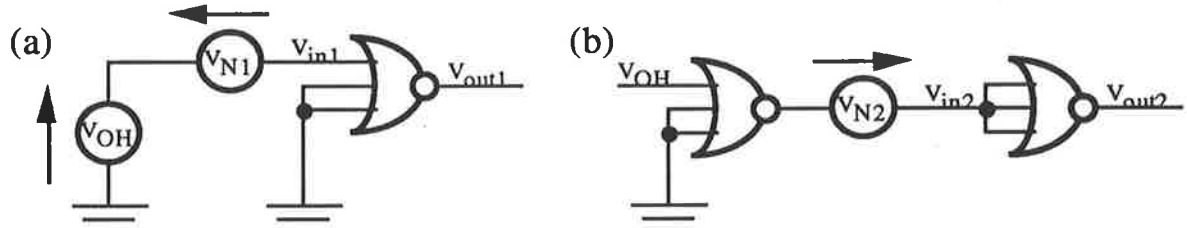


Figure 2.7: Test circuits for determining (a) the intrinsic high and (b) the intrinsic low noise margin

With reference to Figure 2.7(a) the intrinsic high noise margin is defined as the maximum noise voltage V_{N1} which can be applied in the direction as shown such that the output V_{out1} will not change from low to high state. Thus it can be written as:

$$V_{NH} = V_{OH} - V_{inv} = V_{OH} - V_{te} + \frac{V_{td}}{\sqrt{x}}$$

Similarly with reference to Figure 2.7(b), the intrinsic low noise margin can be defined as the maximum noise voltage V_{N2} which can be applied in the direction as shown such that the output V_{out2} will not change from high to low state. Thus it can be written as:

$$V_{NL} = V_{inv} - V_{OL} = V_{te} - \frac{V_{td}}{\sqrt{F_{in} \cdot x}} - \frac{V_{td}^{\,2}}{x \cdot \alpha_e (V_{OH} - V_{te})^2}$$

where F_{in} denotes the amount of fan-in of the gate.

Note that V_{NH} and V_{NL} need not necessarily be the same. The intrinsic noise margins for an inverter and multi-input nor gates are plotted against x in Figure 2.8. It can be seen that the intrinsic low noise margin degrades rapidly as the fan-in increases and the value of x should be chosen between 8 and 16 to achieve high noise margins.

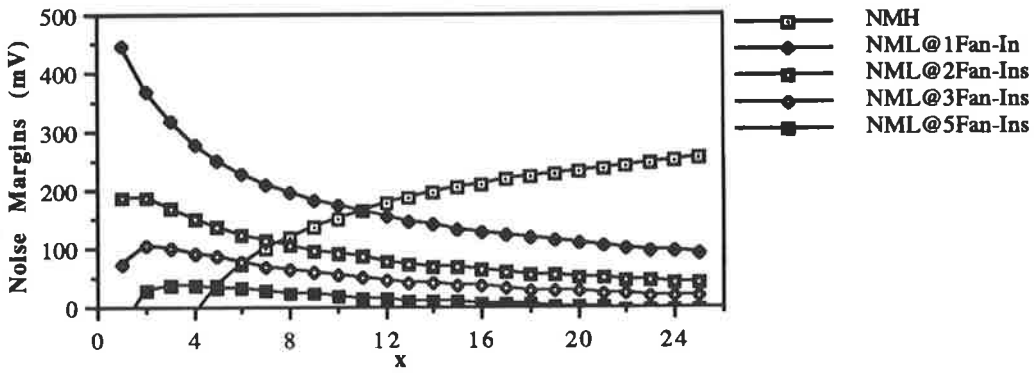


Figure 2.8: Intrinsic noise margins of DCFL gates as a function of x at 125°C

• Inverter Delay

The term delay has been defined ambiguously [LONG&BUTNER]. It can be referred to as the propagation delay, the average between the rise and fall times and the ring-oscillator delay. Here, the delay associated with a DCFL inverter is modelled as the average between the rise and fall times for its mathematical simplicity. It can be written as:

$$\text{Delay} = \frac{\tau_r + \tau_f}{2} = \frac{C_L}{2}\left(\frac{\Delta V}{I_{dsd}} + R_{ON}\right)$$

where τ_r denotes the rise time, τ_f denotes the fall time, C_L denotes the load capacitance, ΔV is the voltage swing, I_{dsd} denotes the saturated drain-to-source current associated with the DFET and R_{ON} is the "on" resistance of the EFET. If we assume C_L to be directly proportional to the width of the EFET then it follows that the fall-time is constant. The rise-time, however, is proportional to C_L/I_{dsd} at a first order approximation. Thus the rise-time is directly proportional to x. The choice of x is therefore a trade-off between the noise margin and the speed.

• Power-Delay Product

The power-delay product is a measure of the effectiveness of the device scaling within the gates [LONG&BUTNER] as well as an indication of the level of integration achievable [ESHRAGHIAN]. Once x is determined the size of the devices should be chosen accordingly. The power-delay product for DCFL gates can be written as:

$$\text{Power-Delay Product} = \frac{Vdd \cdot C_L}{2} \cdot (\Delta V + R_{ON} \cdot I_{dsd})$$

where C_L is the load capacitance and is the sum of the driver, fan-out and the interconnect capacitances. It can be written as:

$$C_L = C_{gdd} + F_{out} \cdot C_{gse} + C_I$$

where C_{gdd} denotes the gate-to-drain capacitance of the DFET, F_{out} denotes the number of fan-outs, C_{gse} denotes the gate-to-source capacitance of the EFET and C_I denotes the interconnect capacitance. For wide pull-down EFETs, C_{gdd} is small compared to C_{gse}. For short EFETs, however, the gate length of the DFET has to be scaled in order to achieve a weak pull-up. In such case C_{gdd} becomes comparable to, or even exceeds C_{gse}. The power delay product of a DCFL inverter is shown in Figure 2.9 at one fan-out and constant x. It can be seen that the power delay product is minimal when $W_e \approx 8\mu m$. Minimum delay, however, can be obtained by using larger devices.

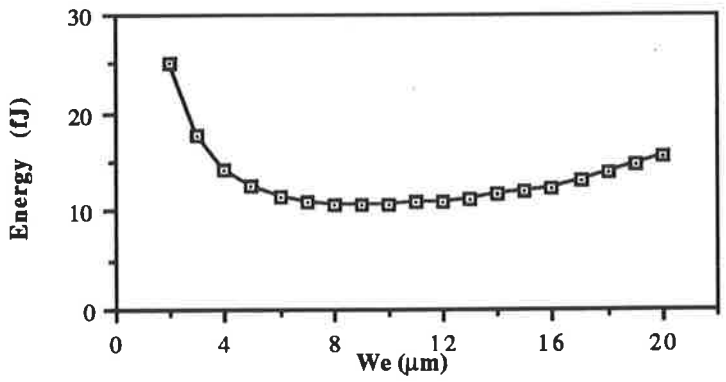


Figure 2.9: Power delay product of DCFL inverters as a function of W_e

Note: C_{gdd} is minimum when $W_e = 16\mu m$

From the above analysis, it is found that by using a pull-down EFET of width 8μm and a 1.2μm long gate together with a pull-up DFET of width 2μm and a 4μm long gate gives a good trade-off between speed, noise margins and power delay-product (which translates to the level of integration achievable as reported in [ESHRAGHIAN]). The optimised DCFL inverter is shown in Figure 2.10.

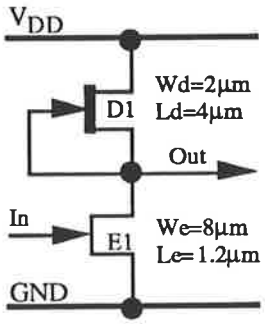


Figure 2.10: Optimised DCFL inverter

2.2.2 Optimising SDCFL

The optimisation of SDCFL is first started by the DCFL stage taking into account that the output high level is at 2 diode drops (≈1.2V) and its output low level can be higher than that of a conventional DCFL gate but lower than the threshold voltage of an EFET since it is fed to a source follower. To maintain compatibility with DCFL, the input capacitance of SDCFL is kept constant by using an EFET of the same size (8μm wide by 1.2μm long). However, SDCFL is a two-stage logic and in order to achieve the same overall speed as DCFL, the first stage is made twice as fast as a normal DCFL gate. This can be achieved by using a 2μm wide by 2.4μm long pull-up DFET. The buffering stage is optimised in a way such that it gives equal high and low noise margins as defined by the "Slope" method (which is just another convenient way of measuring both noise margins) to maximise the worst-case noise margin [WESTE&E]. Figure 2.11 shows the sizing of an optimised SDCFL inverter.

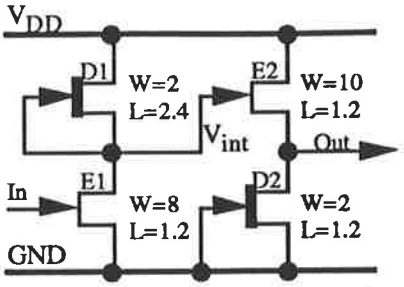


Figure 2.11: Optimised SDCFL inverter

2.2.3 Optimising SBFL

The optimisation of SBFL is similar to that of SDCFL: The input stage is kept unchanged to maintain compatibility with SDCFL whereas the EFETs in the buffering stage are chosen to have the same width to achieve equal rise and fall times. However, the input capacitance will not be the same as the input signal is fed to both pull-down EFETs at the input and buffering stages. The actual width of the output EFETs is chosen so that the optimised gate will have a delay of less than 200ps with a 200fF capacitive load at the output. This ensures a sufficient output current to drive a wire-length of up to 1mm. The sizing of an optimised SBFL gate is shown in Figure 2.12.

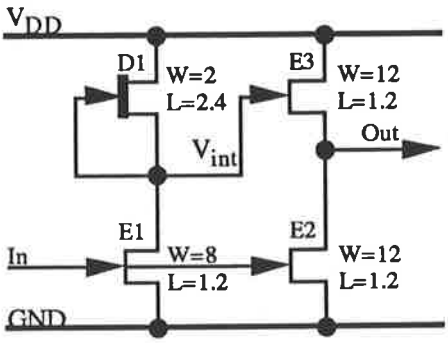


Figure 2.12: Optimised SBFL inverter

2.2.4 Optimising UBFL

The optimisation of UBFL is slightly more complicated. Initially, the DCFL stage (D1 and E1) is chosen to be the same as for SDCFL and SBFL. The size of E2 is a design parameter as it determines the strength available to E4 which translates directly to the speed and noise margin of the entire gate. The transistor pair E3 and E4 should be chosen to be the same as that for SBFL to maintain output strength compatibility. The size of the pull-up DFET D2 should be the same as that for a DCFL inverter. An initially optimised UBFL inverter is shown in Figure 2.13. However, due to the effect of backgating, the strength of D1 must be reduced to preserve the logic levels. To compensate this, the size of the transistor pair E3, E4 are increased to maintain speed. The final sizing of an UBFL inverter is shown in Figure 2.14.

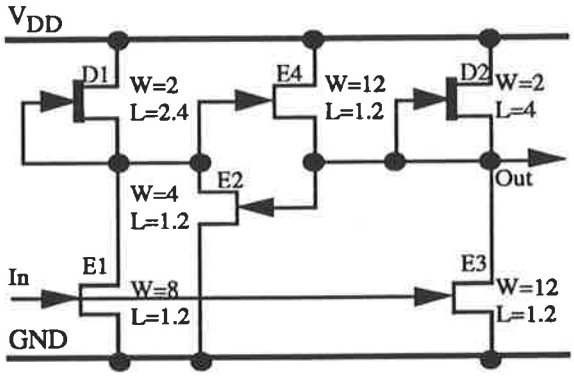


Figure 2.13: Initial optimised UBFL inverter

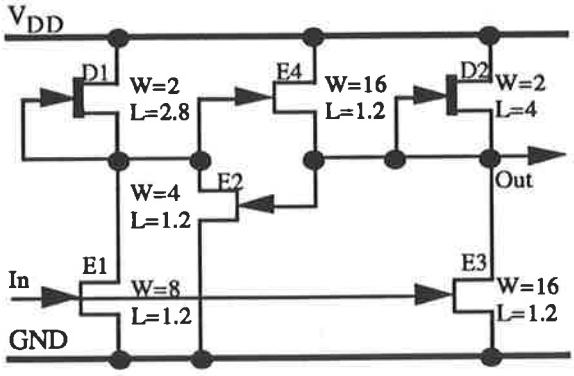


Figure 2.14: Final optimised UBFL inverter

2.3 Performance Comparison

The DC characteristics of the optimised gates are shown in Figure 2.15 where the input voltage is plotted together with the output to help identifying the inverter voltage (V_{inv}). Their performance are compared on the basis of speed, noise margin, power consumption and noise injection to the power bus at the operating condition (125°C and a substrate voltage of 0.6V). As there are many ambiguous definitions for such parameters, the benchmark used for the measurements are presented below:

• Speed: All speed measurements were carried out using a 7-stage ring oscillator.

• Noise Margin: All noise margins were measured by the "maximum square" method [HILL].

• Power Consumption: The sum of the static and dynamic power consumption was measured at 500MHz.

• Noise Injection: The noise injection to the power bus was measured by the static current balance which is expressed as a percentage of the deviation in the current drawn in different logic states and its average value. It can be written as:

$$\text{Static Current Balance} = \frac{\text{Deviation of Current Drawn in Different Logic States}}{\text{Average Current Drawn}} \times 100\%$$

which can be re-written as:

$$\text{Static Current Balance} = \frac{|I_{Hi} - I_{Lo}|}{I_{Hi} + I_{Lo}} \times 100\%$$

where I_{Hi} denotes the steady-state current drawn at the logic "Hi" state

I_{Lo} denotes the steady-state current drawn at the logic "Lo" state

The graphs showing the effect of fan-out, capacitive load and fan-in on the performance of the logic are plotted in Figure 2.16(a) to Figure 2.16(d) and are summarised

in Table 2.1. Note that SBFL and UBFL are used only as buffers due to their complexity. The current drawn from VDD and GND of various inverters are shown in Figure 2.17.

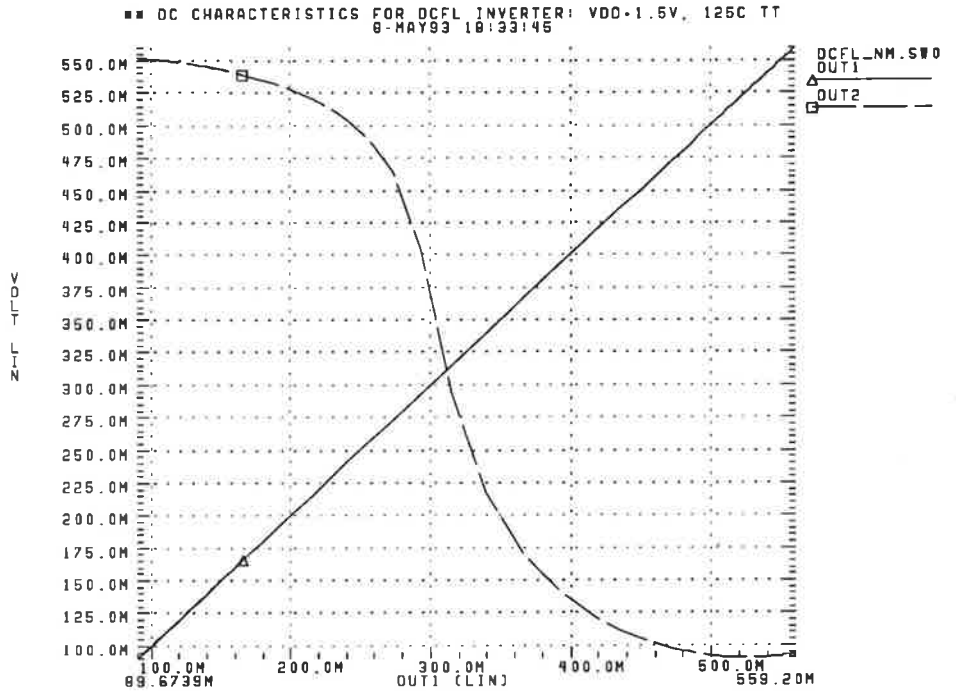


Figure 2.15(a): DC characteristics of DCFL inverter at 1 fan-out

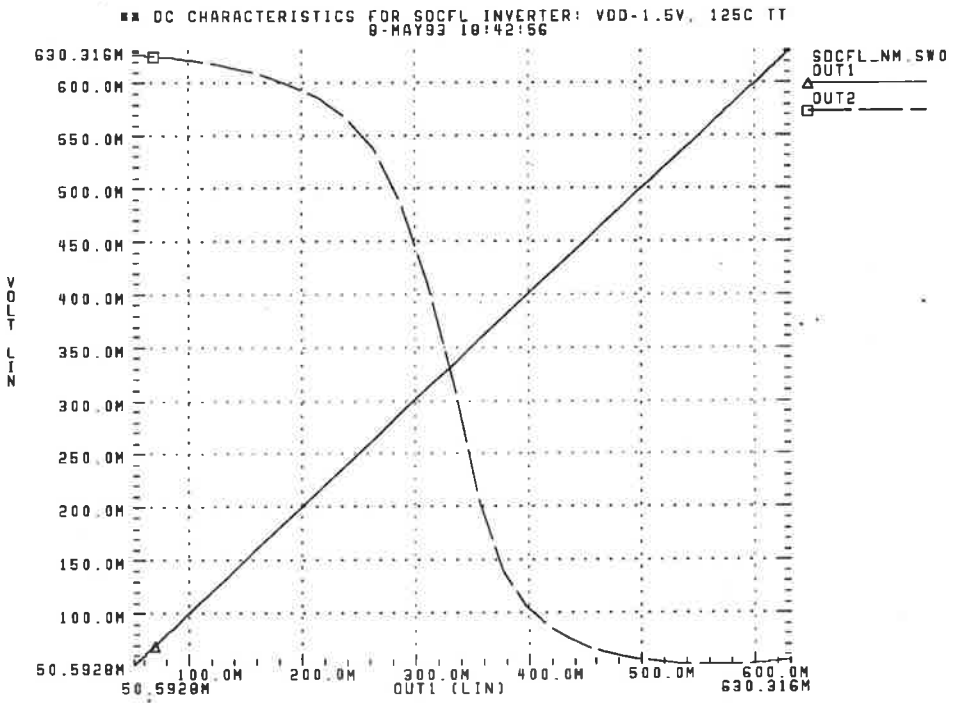


Figure 2.15(b): DC characteristics of SDCFL inverter at 1 fan-out

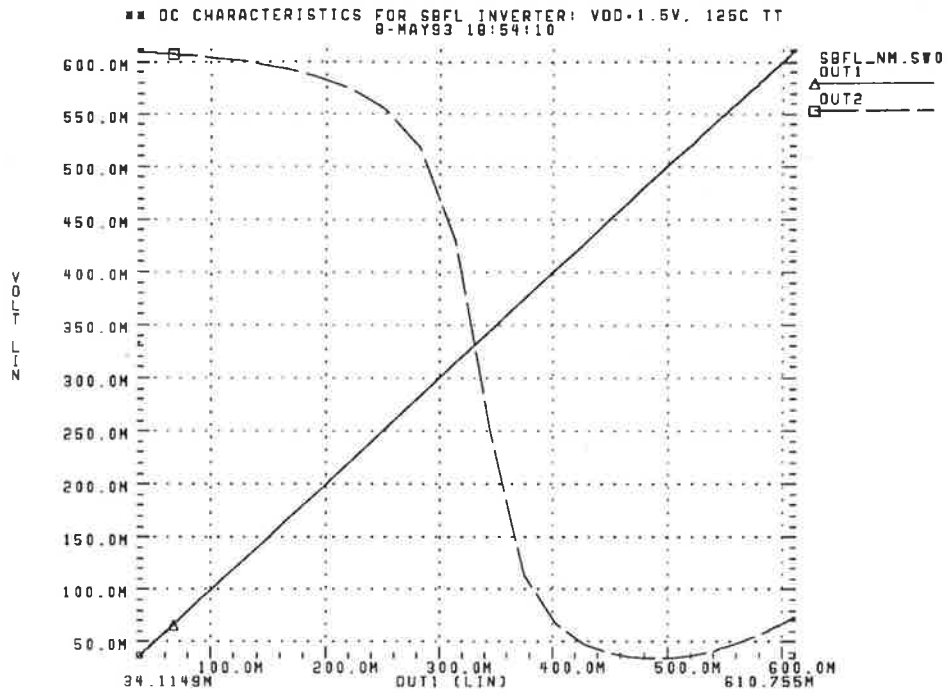


Figure 2.15(c): DC characteristics of SBFL inverter at 1 fan-out

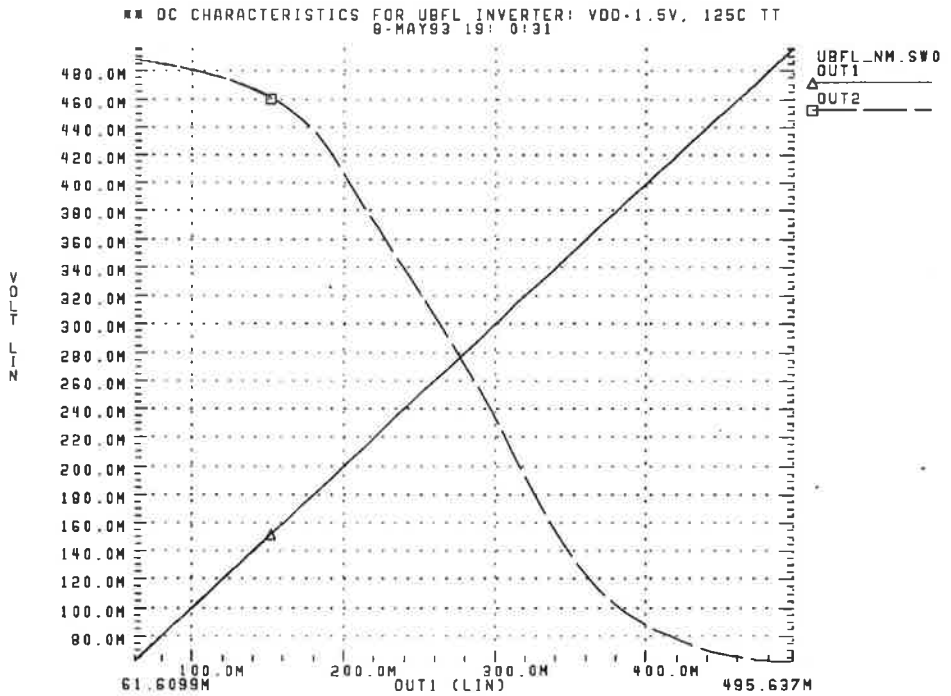
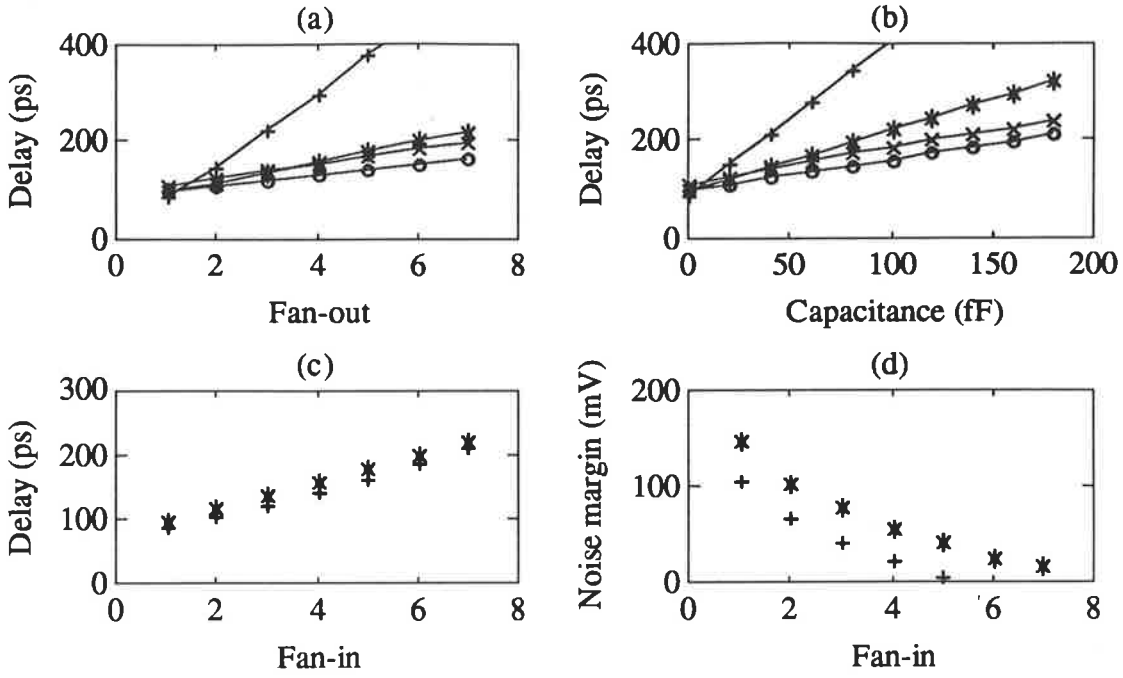


Figure 2.15(d): DC characteristics of UBFL inverter at 1 fan-out

Key: OUT1 denotes the input to the inverters

OUT2 denotes their clamped output



Key: "+" denotes DCFL
"*" denotes SDCFL
"o" denotes SBFL
"x" denotes UBFL

(a) Effect of fan-out on speed
(b) Effect of capacitive loads on speed
(c) Effect of fan-in on speed at one fan-out
(d) Effect of fan-in on noise margin

Figure 2.16: Graph showing the performance of various logic families

Performance	DCFL	SDCFL	SBFL	UBFL
Power Dissipation (μW)	110	360	409	283
Noise Margin (mV)	107	142	192	87
Inverter Threshold (mV)	311	331	331	277
Inverter Delay (ps)	88	97	97	110
Delay/Fan-Out (ps)	75	22	11	15
Delay/Fan-In (ps)	21	20	–	–
Delay/20fF Load (ps)	64	25	13	15
Static Current Balance (% deviation from mean)	7	42	49	10
Number of Transistors for an n-Input Nor Gate	n+1	n+3	2n+2	2n+4
Power-Delay Product at 1 Fan-Out (fJ)	10	35	40	31

Table 2.1: Performance summary of various static logic families in GaAs

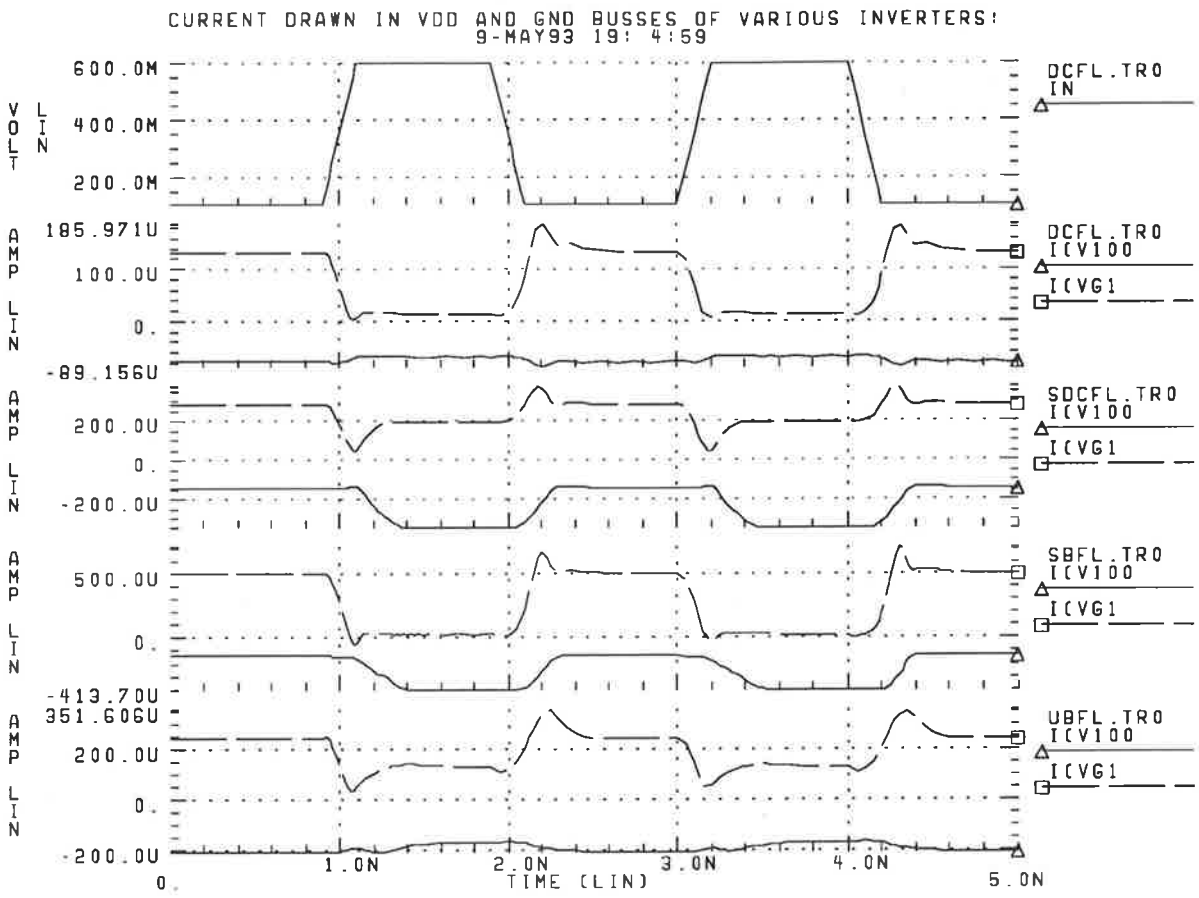


Figure 2.17: HSpice simulations on the current drawn from VDD and GND of various inverters running at 500MHz

Key: IN denotes the input to the inverters

I(V100) denotes the current drawn from VDD

I(VG1) denotes the current drawn from ground

2.4 Logic Design Methodology

After optimising and comparing the four logic families, a "mixed" logic design methodology is proposed: In this design approach, DCFL is used extensively in local logic operations where the amount of fan-in, fan-out and the interconnect length are small. This improves the overall chip performance by reducing the chip area (due to its simplicity) and the power consumption (due to its superb power-delay product). In critical paths and/or higher fan-in applications, SDCFL should be used to achieve a higher speed and noise margins. Due to the complexity of SBFL and UBFL in realising nor structures, they are used strictly as buffers. SBFL should only be used for buffering global signals (such as clocks and resets) due to its high power consumption and the amount of noise injection to the power buses. UBFL is best suited for buffering parallel buses (such as address and data bus) since the power dissipation is low and the noise injection into the power buses is the major concern.

Chapter 3. System Specifications

This chapter describes the overall system specifications of the switch fabric. Issues such as the cell format and the architecture of the switch will be discussed. All designs comply with the CCITT standard for interfacing outside the switch. However, modification to the cell format is allowed within the switch to simplify the hardware implementation.

3.1 The ATM Switch Block

Figure 3.1 shows the typical structure of an ATM switch. It consists of input and output line units, a higher level control block and the switch fabric. The line units interface between the external transmission links (eg. optical fibre) and the switch fabric. They perform both physical and ATM layer functions such as bit synchronisation, cell delineation, Header Error Control (HEC) checking and generation, policing and label translation.

The higher level control block monitors the traffic load on the switch fabric. This information is used internally to divert cells away from congested switch nodes and externally to minimise call set-up blocking by diverting cells away from congested switch blocks.

The switch fabric uses routing information contained in the cell header to route incoming cells to the appropriate output(s). Besides normal cell routing, the switch fabric also supports cell broadcast (where an incoming cell is copied to all outputs of the switch) and multicast (where the incoming cell is copied to a group of outputs). Multicast is treated initially as broadcast with any unwanted cells filtered by the output line units.

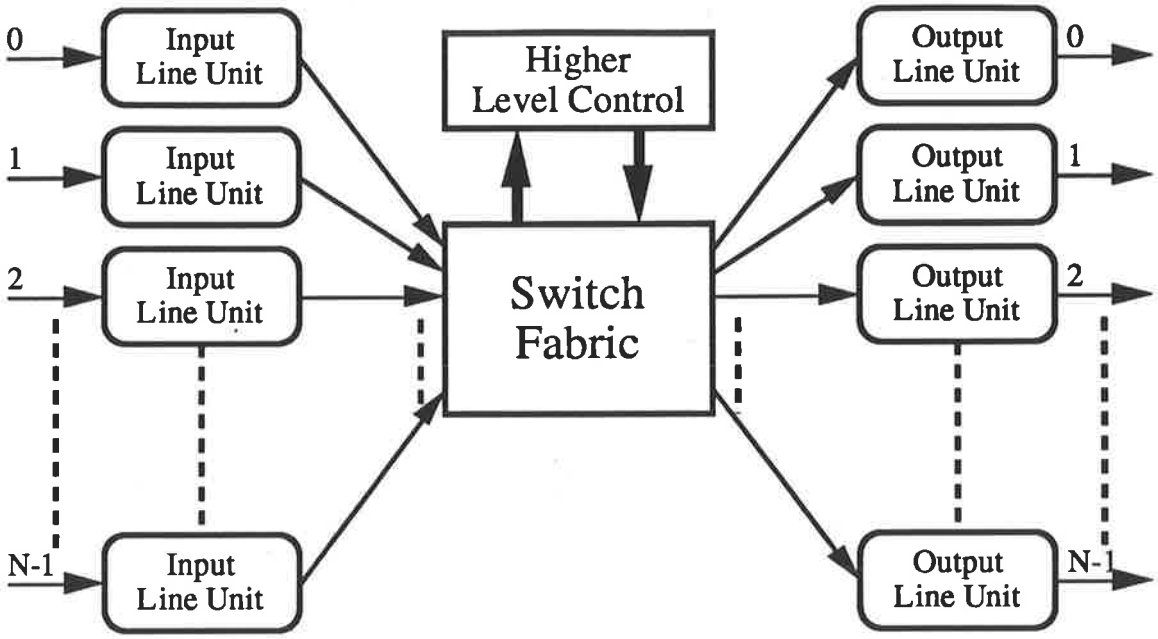


Figure 3.1: Architecture of an ATM switch block

3.2 Cell Format

The format of an ATM cell is specified in [I.361] and is shown in Figure 3.2 at the user network interface (UNI). It has a fixed length of 53 bytes and consists of a 48-byte information field (payload) and a 5-byte header. The header consists of seven different fields:

- Generic Flow Control (**GFC**):

  This field is 4 bits long and is used to monitor the amount of data entering the network to ensure that known capacities are not exceeded.

- Virtual Path Identifier (**VPI**):

  This field is 8 bits long and provides an explicit path identification for the cell.

- Virtual Channel Identifier (**VCI**):

  This field is 16 bits long and provides an explicit channel identification for the cell.

• Payload Type (**PT**):

This 2-bit long field defines the nature of the cell. It has a default value of "00" for user information. However, the value for network control purposes is not yet defined.

• Reserved (**RES**) Bit:

Reserved for future use and is set by default to "0".

• Cell Loss Priority (**CLP**) Bit:

It indicates the cell discard loss priority. It is set to "0" for high priority cells and "1" for low priority cells.

• Header Error Control (**HEC**) field:

This field is eight bits long. It contains the coefficient of a cyclic redundancy check (**CRC**) polynomial to either correct single bit errors or detect multiple bit errors in the header.

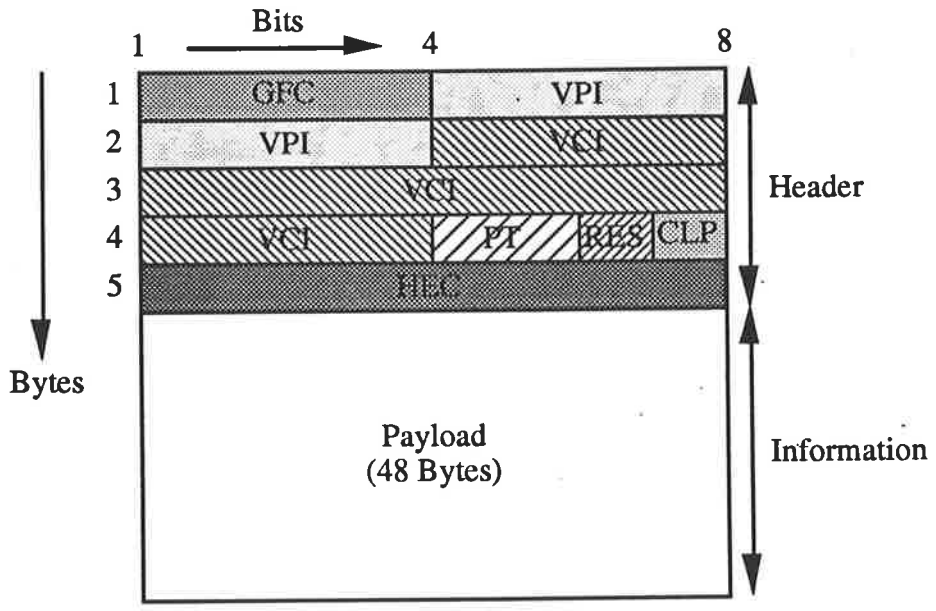


Figure 3.2: ATM cell format at the user network interface (UNI)

The transmission rate for ATM is specified in [I.121] to be either 155Mb/s or 622Mb/s. However, this information can be transferred in a parallel format instead of bit-serially to simplify synchronisation and to reduce the clock frequency. In this project, the ATM cells are carried by eight parallel lines (as described in [JTSKS93] to maintain compatibility with the **BATMAN**[1] project) and extend in time over 53 internal clock cycles. Similar to **BATMAN**, an internal header line (**IH**) is appended to simplify the hardware implementation. It contains all the necessary routing information about the cell as required by the switch fabric and is discussed below:

- Start Bit (**S**):

  This is the first bit in the internal header and is always set to "1" to indicate the beginning of a valid cell. For idle cells, the entire internal header line is set to zero. This simplifies the hardware implementation as no separate cell clock is required.

- Priority Bit (**P**):

  This is simply a copy of the cell loss priority (**CLP**) bit in the cell header.

- Broadcast Bit (**B**):

  Indicates whether a cell is to be broadcast to all outputs.

- Address Field (24 bits):

  This field contains the routing information of the cell within the switch fabric.

- Reserved Field (26 bits):

  Reserved for future use and is set to "0" here.

---

[1] Broadband ATM Access Network. It is a 155Mb/s ATM switch technology demonstrator previously developed at Jydsk Telefon, Denmark.

The format of an internal ATM cell is shown in Figure 3.3. Note that the switch fabric can only modify the Address Field within the internal header.

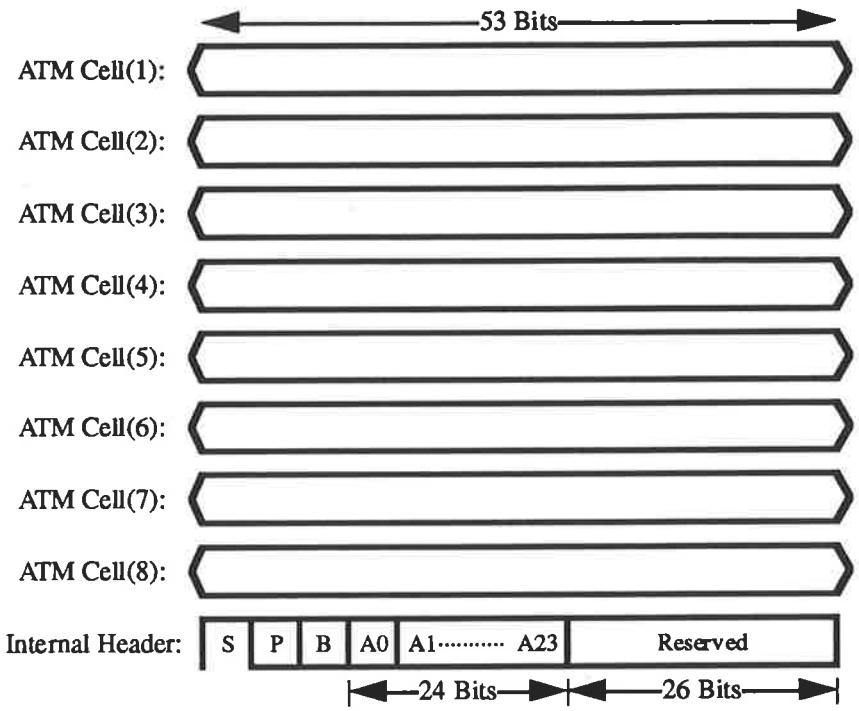


Figure 3.3: Cell format within the switch fabric

3.3 The Switch Fabric

The switch fabric is realised as an "**MSD**" switch, that is, it incorporates Multiplexers at the input, a Space switch at the centre stage and Demultiplexers at the output. The space switch is realised as a Benes network [JTSYB93], [BENES]. An $N \times N$ switch utilising the above architecture is shown in Figure 3.4 for a line speed (**Bt**) of 622Mb/s. Notice that the line speed is increased to effectively 2.4Gb/s[2] before entering the Benes network. This is known as the bit rate conversion technique as reported in [JAKOBSEN93] to reduce the call set-up blocking probability, the delay, the number of switching elements and hence the total power consumption of the switch. To realise a 155Mb/s switch, the configuration as shown

[2] The maximum data rate is determined by the technology used for the switch realisation. In this case, it is 2.4Gb/s since the ATM cell is carried by eight parallel lines each having a maximum frequency of 600MHz.

in Figure 3.5 should be used where the input line speed is increased from 155Mb/s to 2.4Gb/s before entering the Benes network for the same reason.

The issue of call set-up blocking probability in a Benes network is reported in [JAKOBSEN93] and is given by the following expression:

$$P_{Blocking} \le 2 \sum_{j=1}^{\frac{\log N}{\log M}-1} \sum_{r=L_I+1}^{M^j L_E} \frac{r - L_I}{r} \binom{M^j L_E}{r} \left(M^{-j}\right)^r \left(1 - M^{-j}\right)^{M^j L_E - r}$$

where $P_{Blocking}$ denotes the call set-up blocking probability

N denotes the switch size

M denotes the size of the switching elements ($M = 2$ in this case)

L_E denotes the number of calls placed at the external lines and is given by the expression: $L_E = B_E/B_C$

L_I denotes the maximum number of calls at any internal line and is given by the expression: $L_I = B_I/B_C$

B_E denotes the maximum total load on any external input or output line

B_I denotes the maximum total load on any internal line

B_C denotes the maximum equivalent load of a call allowable

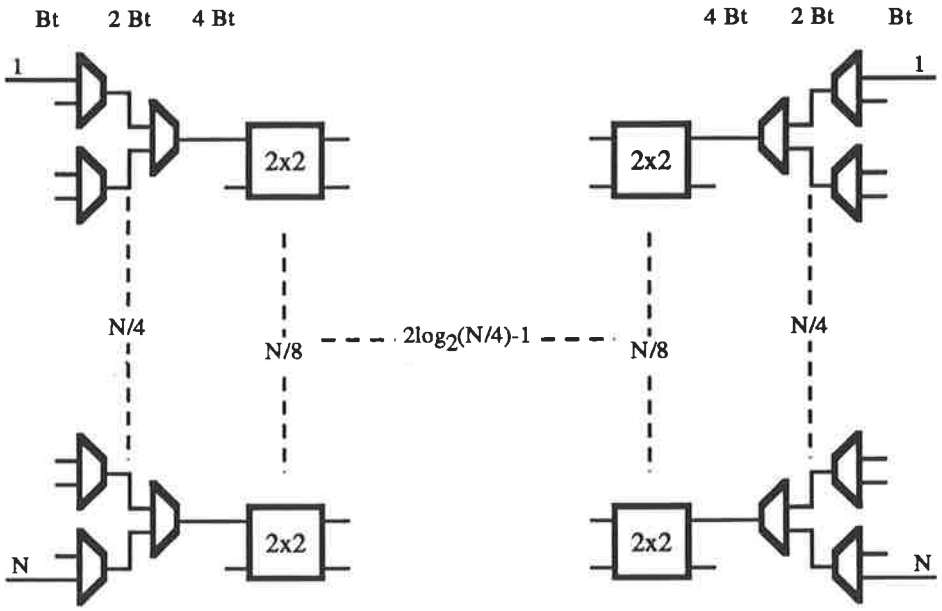


Figure 3.4: An *N x N* 622Mb/s ATM switch fabric realised by *2 x 2* switching elements in a Benes network configuration

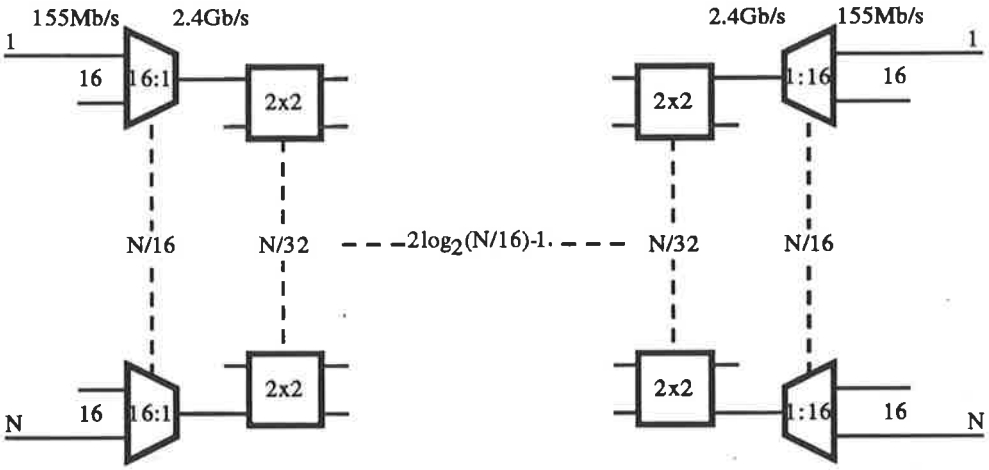


Figure 3.5: An *N x N* 155Mb/s ATM switch fabric realised by *2 x 2* switching elements in a Benes network configuration

3.3.1 Performance Requirement

The performance desired [JTSKS93] for the switch fabric is summarised in Table 3.1. Note that this is not yet a strict requirement due to the experimental status of the project.

Performance Criteria	Performance Requirement
Maximum Size	*1024 x 1024*
Maximum Delay	250μs
Cell Loss Probability	Less than 10^{-8}
Call Set-Up Blocking	Less than 1% at all load conditions
Traffic Load	60% due to the experimental status of this project (Could be increased to 80% by increasing the buffer size)
Average Bit Rate	Not to exceed 2% relative to 622Mb/s
Traffic Burstiness	Maximum burst bit rate of 10% relative to 622Mb/s
Power Consumption	Less than 10kW for a *1024 x 1024* 622Mb/s switch

Table 3.1: Performance required for the switch fabric

3.4 Realising the Switch Fabric

The entire switch fabric is designed such that it requires only three different chip-sets [JTSYB93]. They consist of a Buffer chip, a Router chip and a Multiplexer chip which are described in sections 3.4.1 to 3.4.3. The same external data interfaces are used for the three chips and is shown in Figure 3.6.

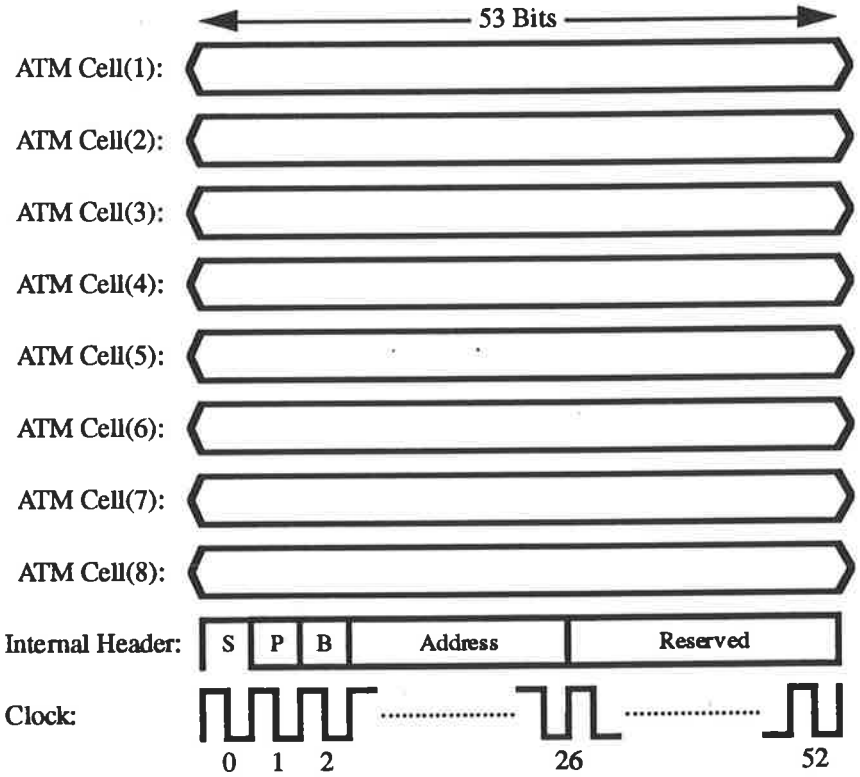


Figure 3.6: The data interface common to all three chips

3.4.1 The Buffer Chip

The buffer chip receives a single input cell stream and sends it out to the output at half the input speed. Consequently, the input data rate (2 Bt or a maximum of 4.8Gb/s for a 622Mb/s switch) is twice that of the output (Bt or a maximum of 2.4Gb/s for a 622Mb/s switch). Fluctuations in the incoming cell stream are absorbed by an internal buffer which

has a size of 32 cells[3] . The buffer size was chosen according to [TOBAGI] based on the maximum tolerable cell loss probability of 10^{-8} and a traffic load of 70% (60% at the input multiplied by a speed up factor of 1.2). Figure 3.7 shows the interface of the buffer chip.

In realising this chip, an isochronous clocking strategy is used where the input section uses the clock from the input data interface and the external clock is used for other synchronous parts. Signal communications between sections running at different speed are handled asynchronously. The main reason for using an isochronous clocking strategy lies in the different operational speed at the input and output. Furthermore, problems with clock skew and clock distribution are the major concerns for a large chip.

The anticipated power dissipation of the buffer chip is 1W in order to meet the power requirement of 10kW for a *1024 x 1024* 622Mb/s switch (see Appendix E for detailed calculations). Consequently, the on-chip 32-cell internal buffer will have to be realised by dynamic RAM (DRAM) cells to meet the power requirements.

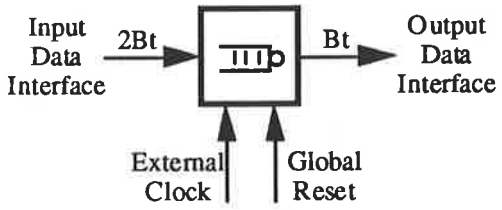


Figure 3.7: Interfacing the buffer chip

(Note that the external clock input has half the frequency as the input data interface)

3.4.2 The Router Chip

The router chip receives cells at the input and sends them out to either one or both outputs depending on the routing information contained in the internal header (**IH**) and the

[3] The buffer size was calculated by Mr. Jens Jakobsen at Jydsk Telefon, Denmark.

broadcast enable (**BE**) pin on the chip. Figure 3.8 shows the interface of the router chip. Note that there is no cell loss in this chip and the maximum data rate for both the input and output ports are 4.8Gb/s (eight lines at 600MHz each).

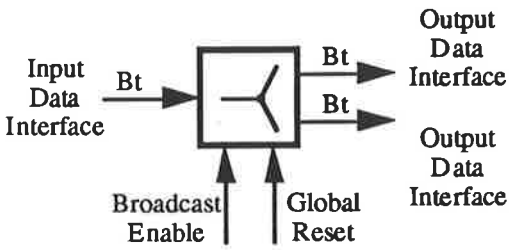


Figure 3.8: Interfacing the router chip

(Note that the broadcast enable input determines whether the chip can broadcast cells)

Due to the fact that both the input and outputs run at the same clock speed, isochronous clocking is not necessary and the clock is simply driven from the input data interface.

The anticipated power dissipation of the router chip is 0.7W to meet the power requirement of 10kW for a *1024 x 1024* 622Mb/s switch (see Appendix E for detailed calculations).

3.4.3 The Multiplexer Chip

The multiplexer chip receives two input cell streams at the input and multiplexes them to a single cell stream at the output at twice the input speed. Hence, the maximum data rate at the input and output ports are 2.4Gb/s (eight lines at 300MHz each) and 4.8Gb/s (eight lines at 600MHz each) respectively. It requires a 4-cell internal buffer for temporary storage so that no cell loss occurs. Figure 3.9 shows the interface of the multiplexer chip.

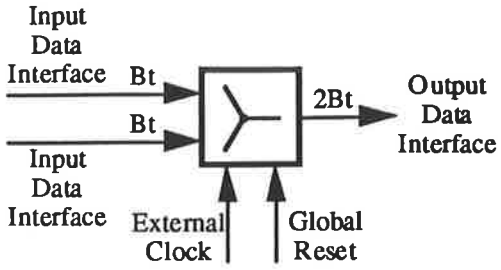


Figure 3.9: Interfacing the multiplexer chip

(Note that the external clock input runs at twice the frequency as the input data interface)

Similar to the buffer chip, isochronous clocking strategy is used since the inputs and output operate at different speed. Moreover, it simplifies the problem with clock distribution and clock skew.

The anticipated power dissipation of the multiplexer chip is 0.8W to meet the power requirement of 10kW for a *1024 x 1024* 622Mb/s switch (see Appendix E for detailed calculations).

After defining the basic elements, higher level building blocks can be constructed to realise the switch fabric. Three basic building blocks are: the *2 x 2* switch, input multiplexers and output de-multiplexers which are described in sections 3.4.4 to 3.4.6 respectively.

3.4.4 The *2 x 2* Switch

A *2 x 2* switch can be constructed by the three elements as shown in Figure 3.10. The two incoming cell streams each running at 2.4Gb/s are initially multiplexed at double speed (4.8Gb/s) by a multiplexer chip. The multiplexed cell stream is then routed to the appropriate output by a router chip and slowed down by a factor of two by buffer chips at each output. As a result, the *2 x 2* switch is output buffered.

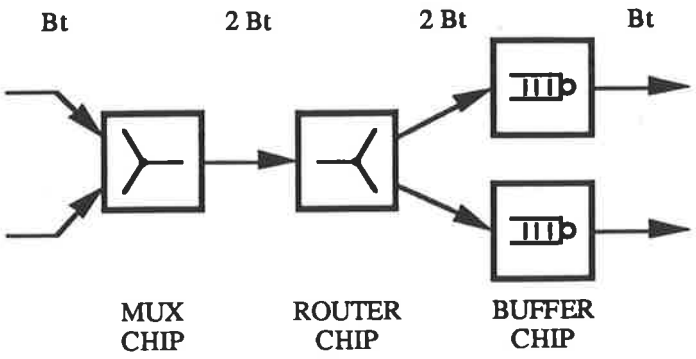


Figure 3.10: Realising a *2 x 2* output buffered switch by the three elements

3.4.5 Input Multiplexers

The input multiplexers for the 622Mb/s switch are 4-to-1. They can be realised by three multiplexer chips as shown in Figure 3.11(a). For the 155Mb/s switch fabric, the input multiplexers are 16-to-1 and can be realised by five 4-to-1 multiplexers as shown in Figure 3.11(b). The maximum bit rate on the outputs are 2.4Gb/s for both cases.

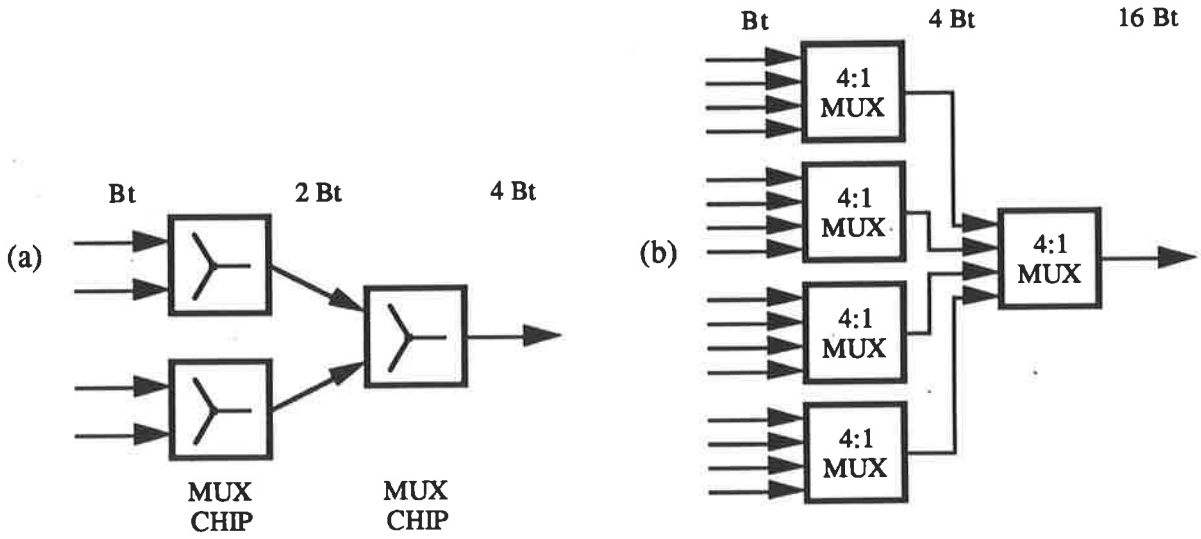


Figure 3.11: Input multiplexer for (a) 622Mb/s and (b) 155Mb/s switch

3.4.6 Output Demultiplexers

The output demultiplexers for the 622Mb/s switch are 1-to-4 and can be realised by three router chips and six buffer chips as shown in Figure 3.12(a) whereas for the 155Mb/s switch, they are 1-to-16 and can be realised by five 1-to-4 demultiplexers as shown in Figure 3.12(b). The maximum bit rate on the inputs are 2.4Gb/s for both cases.

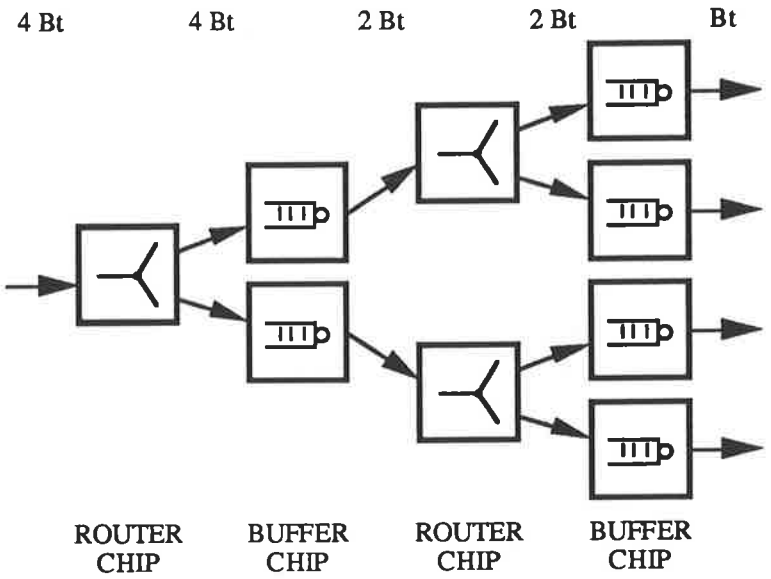


Figure 3.12(a): Output demultiplexer for the 622Mb/s switch

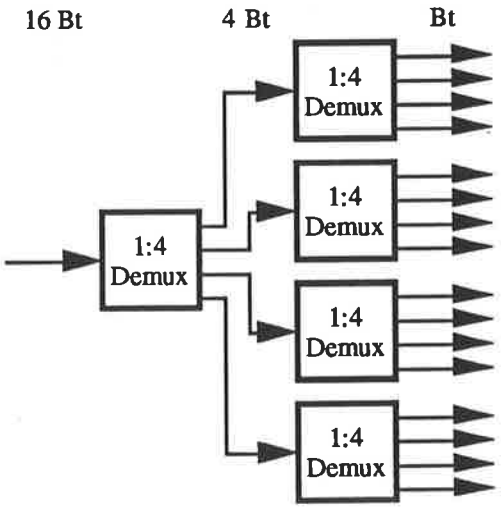


Figure 3.12(b): Output demultiplexer for the 155Mb/s switch

Chapter 4. Layout Style and Design Tools

This chapter describes the layout style used throughout the control logic within the buffer chip. Various computer aided design (CAD) tools used in the design phase are also introduced.

4.1 Layout Style

The layout style has a major impact on the performance of very high speed VLSI circuits. Some key objectives in selecting a layout style are to:

- Minimise interconnect lengths to reduce parasitic capacitances, hence the coupling between high speed signals.

- Reduce the inductance and increase the capacitance associated with the power busses to reduce voltage and current spikes.

- Achieve a high packing density.

The "Ring Notation" as reported in [ESH,S,C&N], [ESHRAGHIAN] and [ESHRAGHIAN92] is used explicitly to achieve these goals. It is a generic term given to a free form topological symbolic layout in which graphical symbols are placed relative to each other rather than in an absolute manner. Figure 4.1 illustrates the "Ring Notation" layout style for a DCFL inverter where the VDD and ground busses are placed in parallel and in close proximity of one another to increase the decoupling capacitance. Transistors are placed below the ground bus to reduce noise injection because the AC currents in the ground bus are smaller, and isolate the transistors from the effects of the larger AC currents in the power

bus. Furthermore, all transistors are placed in the same orientation to minimise mismatch due to process variations.

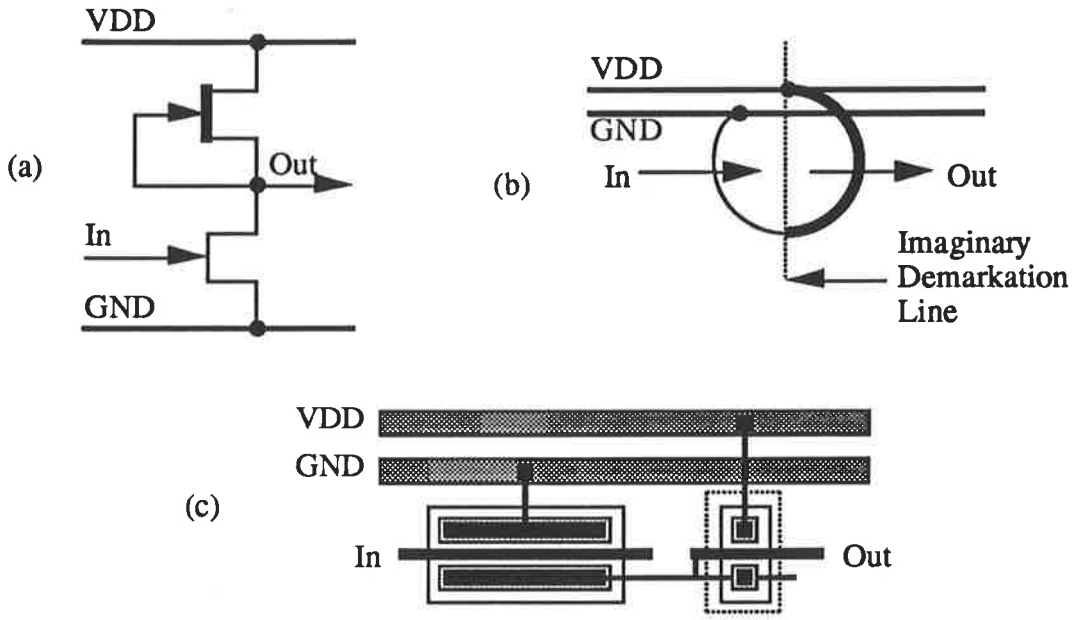


Figure 4.1: (a) Circuit diagram
(b) Ring notation representation and
(c) Symbolic notation of a DCFL inverter

It is reported in [ESH,C,M&C] that the noise injection to the power busses can be further reduced by decreasing the characteristic impedance of the lines. This can be achieved by using a grid like connection for VDD and ground busses as shown in Figure 4.2. It is shown in [ESH,C,M&C] that the noise amplitude can be reduced to one-third of the original after arriving at the first cross-point. In a circuit's point of view, this translates to reduced inductance and resistance as well as increased capacitance associated with the power busses since they are connected in parallel.

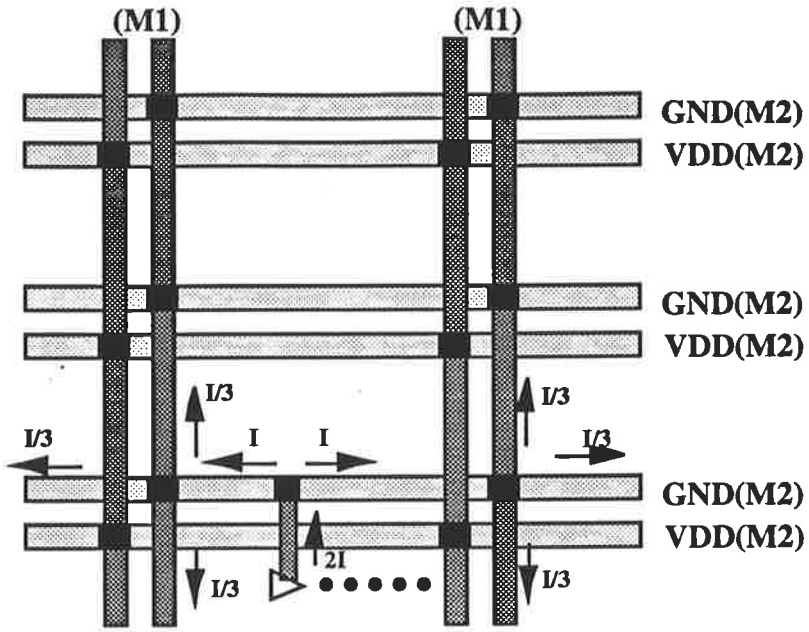


Figure 4.2: Using net-like power lines to reduce the noise injected into the power buses

Note: Triangle denotes an inverter

Vertical lines denotes global power lines

Horizontal lines denotes local power lines

4.2 Design Methodology

The design phase can be divided into three main stages: The specification, the realisation and the verification stage.

In the specification phase, a VHDL behavioural description is written for the design to describe its functionality. It is then simulated in **ASIMUT** (a circuit simulation tool within the **ALLIANCE**[4] package) with a predefined set of input stimuli to verify its behaviour.

---

[4] ALLIANCE is a set of CAD tools for VLSI design by the Universite Pierre et Marie Curie (Paris 6, FRANCE).

In the realisation stage, a VHDL structural description is first written for the design based on standard library cells (see Chapter 5) which is then simulated and compared with the behavioural description to achieve a one-to-one correspondence. The structural description for the circuit is then laid-out in **MAGIC**[5] with standard cells.

The verification stage is carried out in three phases:

• Netlist Verification

The laid-out circuit is first converted to a spice netlist by the program **ext2sp** which is then converted into the **VALID**[6] format by the program **magic2valid**. This netlist is then compared by the program **NETCOMPARE** with a similar one generated from the VHDL structural description to ensure both circuits match by containing the exact type and number of transistors and connections.

• Functional Verification

Two levels of simulation are used. In the first phase, the layout is extracted and converted to a **sim** format by the program **ext3sim. IRSIM**, a switch level simulator within **MAGIC**, is then used to verify its connectivity and functionality. The merit for using **IRSIM** lies in its ability to check any undefined inputs and outputs which **ASIMUT** fails to handle.

In the second phase, a full **HSPICE**[7] simulation is carried out after the extracted circuit is converted to spice format by the program **ext2sp** to verify the speed requirement as well as providing an estimate of the circuit's power consumption. The relative merits of various simulation tools are summarised in Table 4.1.

---

5 MAGIC is a full-custom layout tool by University of California at Berkeley.
6 VALID is a schematic layout and simulation tool by CADENCE Software.
7 HSPICE is a spice simulation tool by Meta-Graphics.

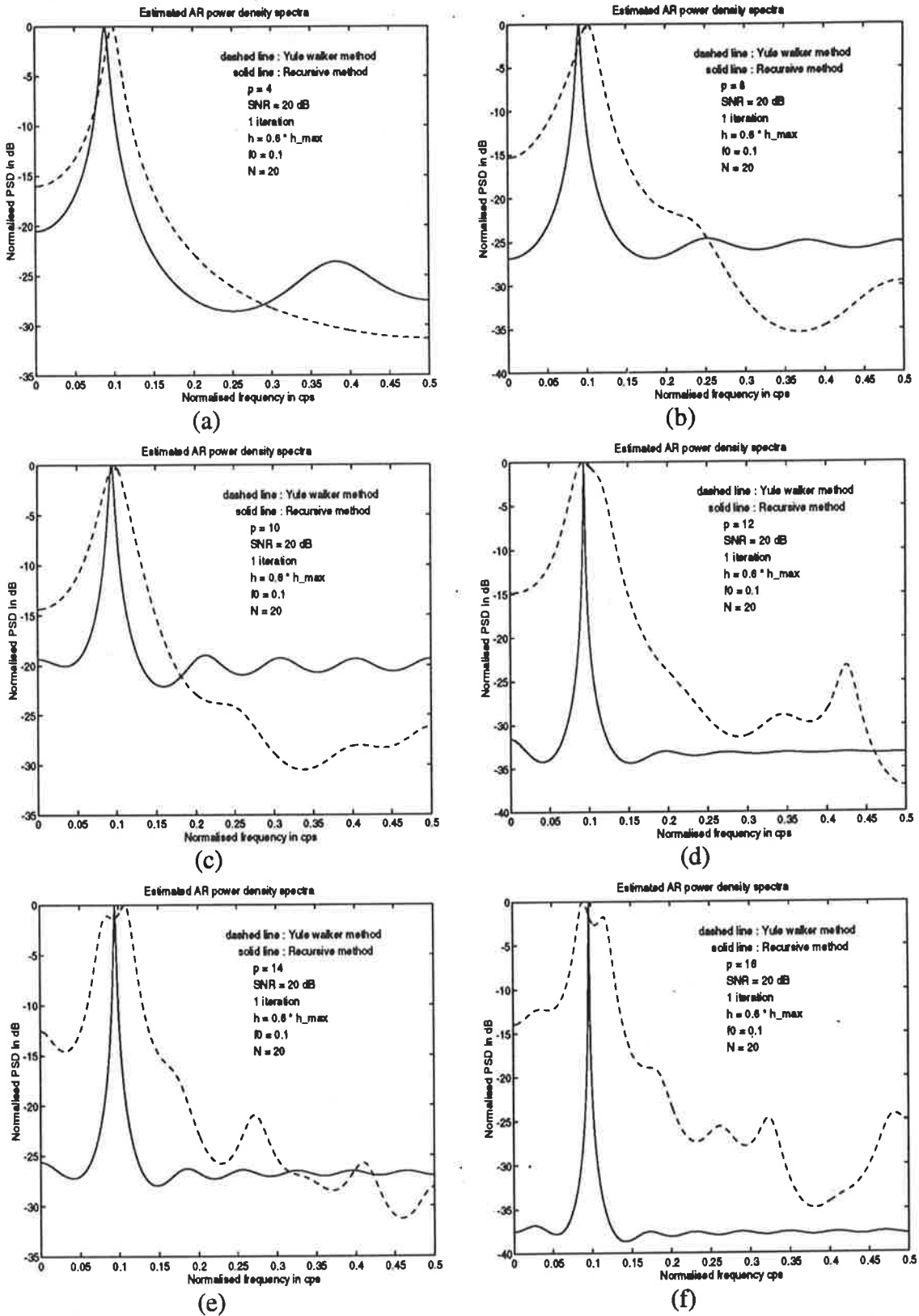


FIGURE 4.5 Comparison of AR PSDs between Yule Walker method and Recursive method at a low frequency 0.1 cps with h = 0.6 * h_max and SNR = 20 dB after 1 iteration. {(a) p = 4, (b) p = 8, (c) p = 10, (d) p = 12, (e) p = 14 and (f) p = 16)

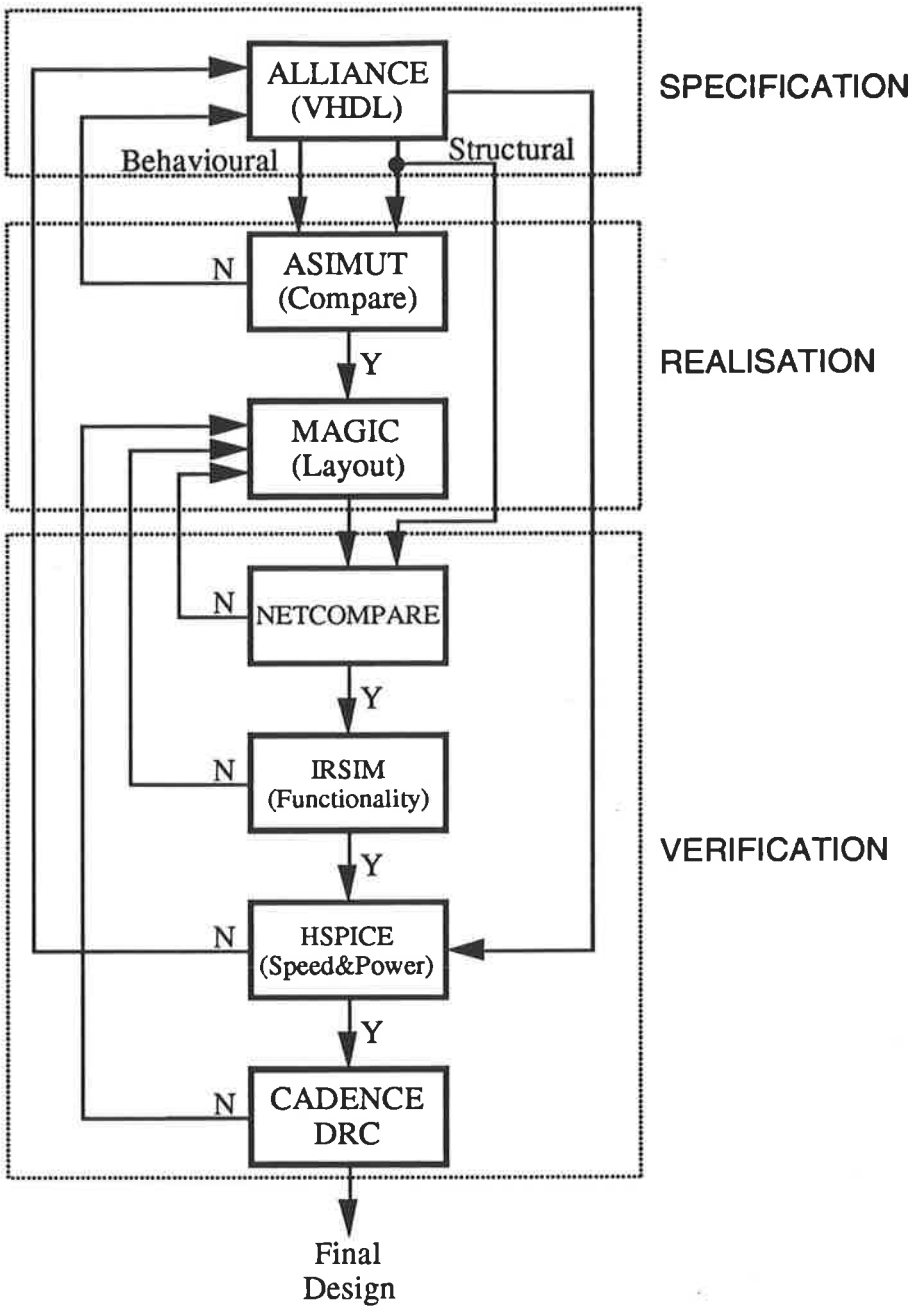


Figure 4.3: Flow chart showing different design phases

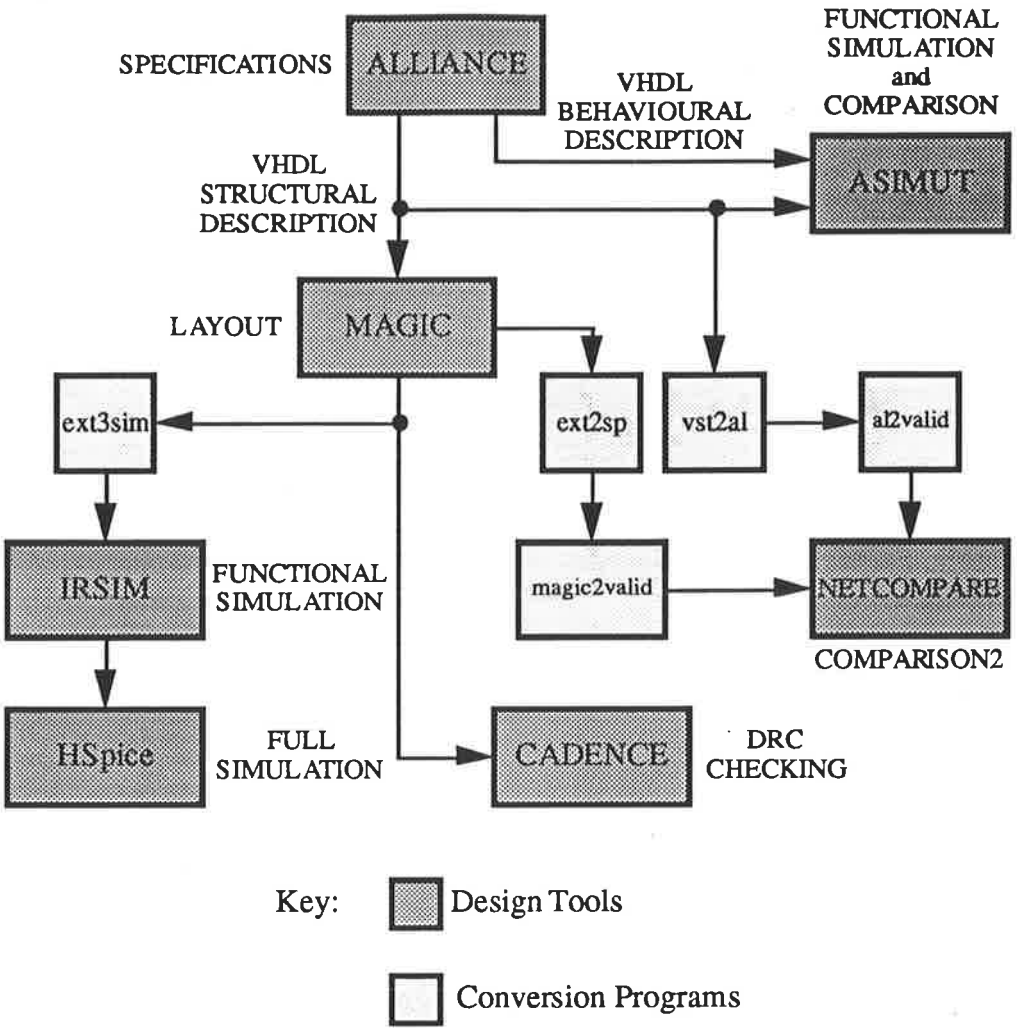


Figure 4.4: A flow diagram showing various tools and conversion programs used in the design phase

Chapter 5. Primitives

This chapter describes the basic primitives used throughout the buffer chip. It is also intended to provide a complete library for the VHDL descriptions of the lowest level cells [CHU1]. All sizing shown in this chapter are drawn size only, the actual gate lengths are 0.4μm shorter.

5.1 Logic Gates

All four logic families as described in Chapter 2 are used in the construction of the buffer chip. Due to the complexity of SBFL and UBFL, they are used strictly as buffers. Moreover, all primitives are given a unique name which are consistent with the instance names used in the VHDL structural descriptions.

5.1.1 Inverters

Inverters from all four logic families were used according to the output load, speed and noise injection requirements (see Chapter 2). They were optimised for high temperature operation (125°C) and a speed of approximately 100ps at one fan-out. The sizing of each inverter are also shown for convenience.

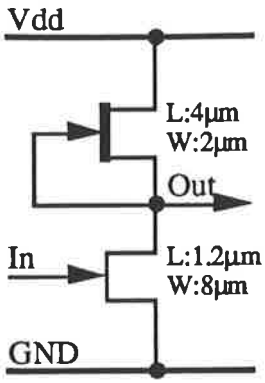


Figure 5.1: DCFL inverter "nd"

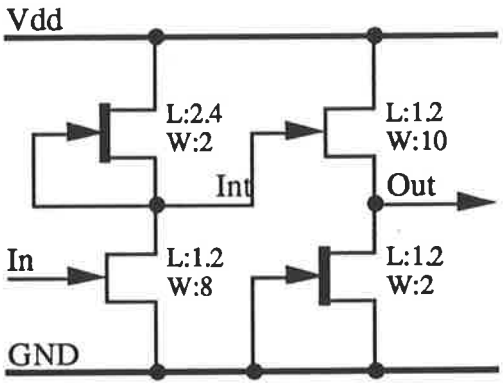


Figure 5.2: SDCFL inverter "ns"

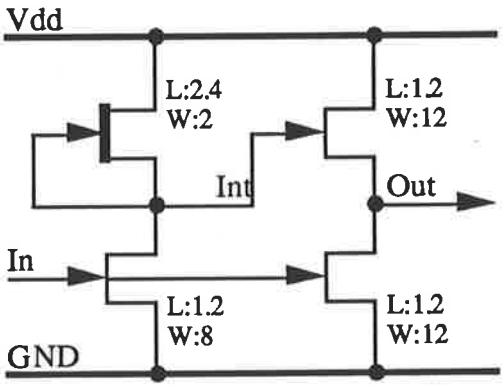


Figure 5.3: SBFL inverter "nv"

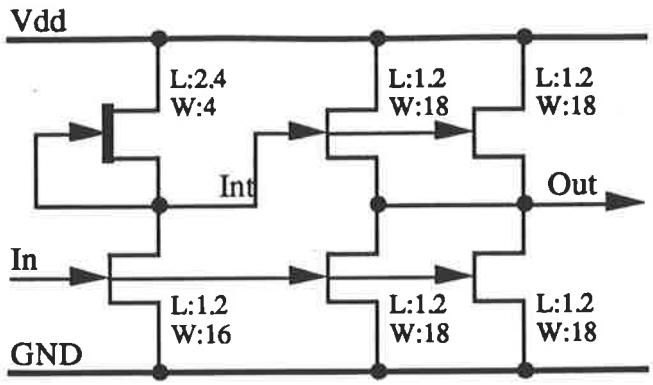


Figure 5.4: Three times normal sized SBFL inverter "n3v"

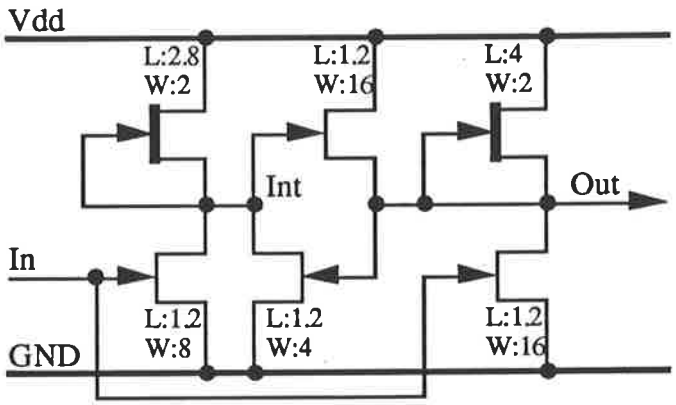


Figure 5.5: UBFL inverter "nu"

5.1.2 2-Input Nor Gates

Only DCFL and SDCFL nor gates are considered.

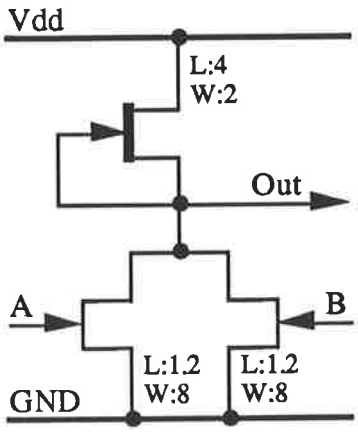


Figure 5.6: 2-Input DCFL nor gate "o2nd"

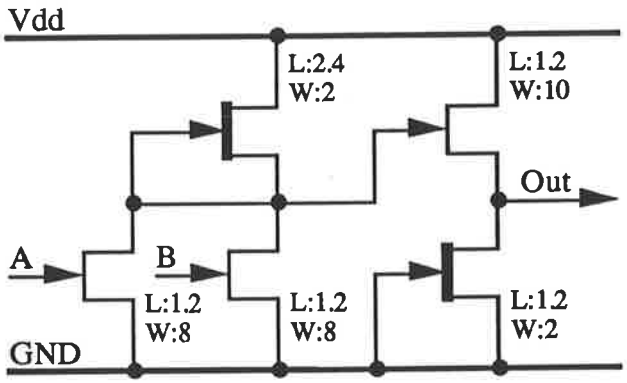


Figure 5.7: 2-Input SDCFL nor gate "o2ns"

5.1.3 3-Input Nor Gates

As for the case of 2-input nor gates, only DCFL and SDCFL 3-input nor gates are considered. Due to the low noise margin of the 3-input DCFL nor gate, it is only used with one fan-out.

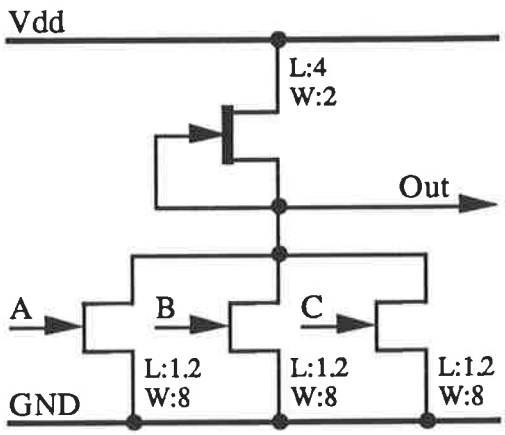


Figure 5.8: 3-Input DCFL nor gate "o3nd"

(Only single fan-out allowed due to noise margin requirement)

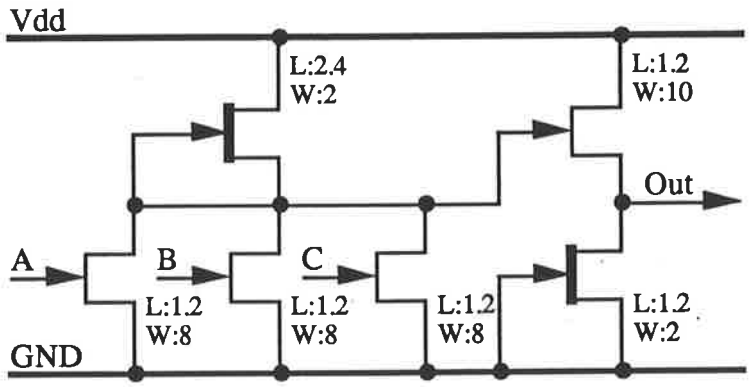


Figure 5.9: 3-Input SDCFL nor gate "o3ns"

5.1.4 4-Input Nor Gates

Only SDCFL can support 4-input nor gates due to noise margin requirement.

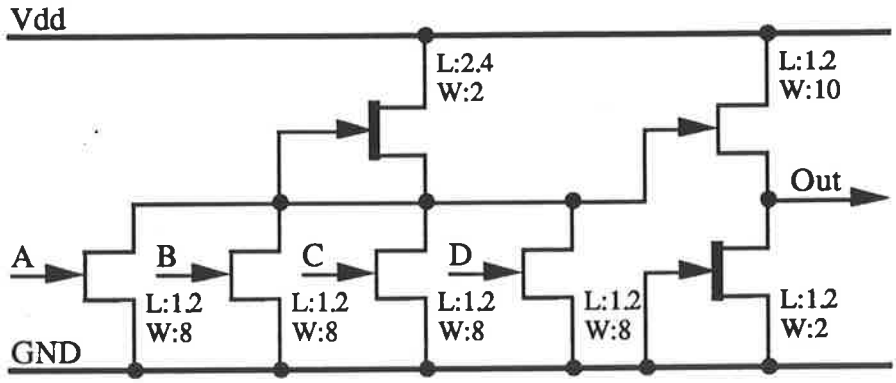


Figure 5.10: 4-Input SDCFL nor gate "o4ns"

5.1.5 Or Gates

"Or" gates are realised by appending an inverter to the output of nor gates. Two DCFL "or" gates are used as primitives.

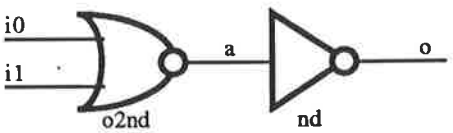


Figure 5.11: Circuit diagram for DCFL 2-input or gate "o2dd"

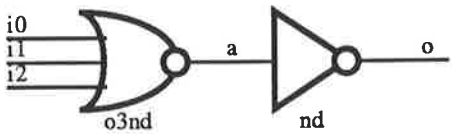


Figure 5.12: Circuit diagram for DCFL 3-input or gate "o3dd"

5.2 RS Flip-Flops

Four different types of RS flip-flops are used in the construction of the buffer chip with their circuit diagrams shown in Figure 5.13 to Figure 5.16. They are chosen due to the need for reset and different output buffering requirements. Table 5.1 shows their application remarks.

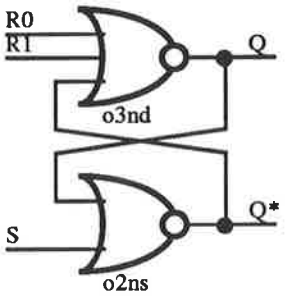


Figure 5.13: Circuit diagram for RS flip-flop "rrsds"

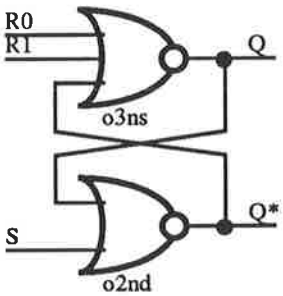


Figure 5.14: Circuit diagram for RS flip-flop "rrssd"

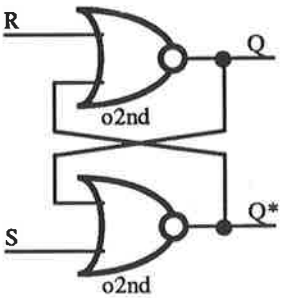


Figure 5.15: Circuit diagram for RS flip-flop "rsdd"

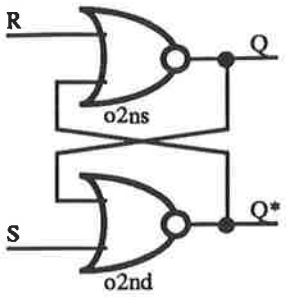


Figure 5.16: Circuit diagram for primitives "rssd"

Instance Name	Application Remarks
rrsds	• RS flip-flop with reset • Q output cannot be used • Q* output can drive another 3 fan-outs or a 60fF capacitive load
rrssd	• RS flip-flop with reset • Q output can drive another 2 fan-outs or a 40fF capacitive load • Q* output can drive an extra fan-out or a 20fF capacitive load
rsdd	• RS flip-flop with no preset/clear • Q defaults "Lo" and Q* defaults "Hi" in VHDL behavioural description • Both Q, Q* outputs can drive an extra fan-out or a 20fF capacitive load
rssd	• RS flip-flop with no preset/clear • Q defaults "Lo" and Q* defaults "Hi" in VHDL behavioural description • Q output can drive another 4 fan-outs or an 80fF capacitive load • Q* output can drive an extra fan-out or a 20fF capacitive load

Table 5.1: Instance name and application remarks of RS flip-flops used in buffer chip

5.3 Latches

Five different data latches with instance name "j" are used in the construction of the buffer chip. Table 5.2 shows the module name in the VHDL structural descriptions together with their differences in application regarding output buffering and the need for preset/clear. Their individual circuit diagrams are shown in Figure 5.17 to Figure 5.21 respectively.

Instance Name	Application Remarks
jdd	• Data latch with no preset/clear • Q defaults "Lo" and Q* defaults "Hi" in VHDL behavioural description • Both Q, Q* outputs can drive an extra fan-out or a 20fF capacitive load
jcdd	• Data latch require manual clear on start-up • Q output cannot be used • Q* output can drive an extra fan-out or a 20fF capacitive load
jds	• Data latch with no preset/clear • Q defaults "Hi" and Q* defaults "Lo" in VHDL behavioural description • Q output can drive an extra fan-out or a 20fF capacitive load • Q* output can drive another 4 fan-outs or an 80fF capacitive load
jsd	• Data latch with no preset/clear • Q defaults "Lo" and Q* defaults "Hi" in VHDL behavioural description • Q output can drive another 4 fan-outs or an 80fF capacitive load • Q* output can drive an extra fan-out or a 20fF capacitive load
jpsd	• Data latch require manual preset on start-up • Q output can drive an extra fan-out or a 20fF capacitive load • Q* output cannot be used

Table 5.2: Instance name and remarks of latches used in buffer chip

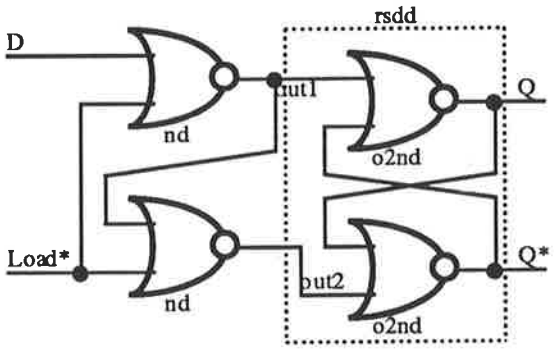


Figure 5.17: Circuit diagram for data latch "jdd"

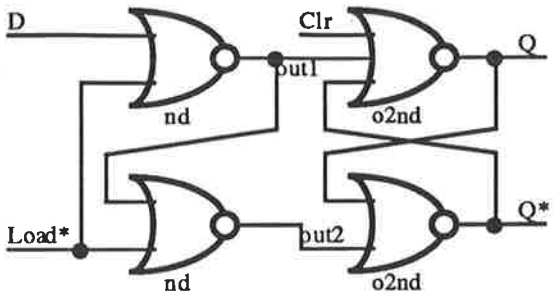


Figure 5.18: Circuit diagram for data latch "jcdd"

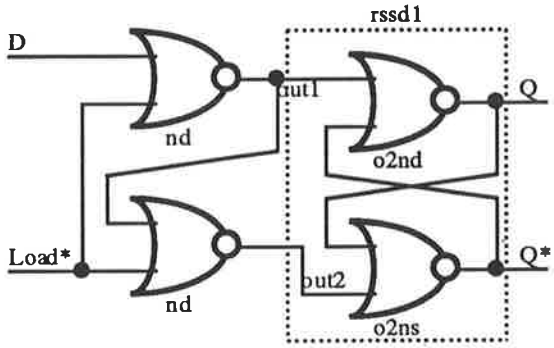


Figure 5.19: Circuit diagram for data latch "jds"

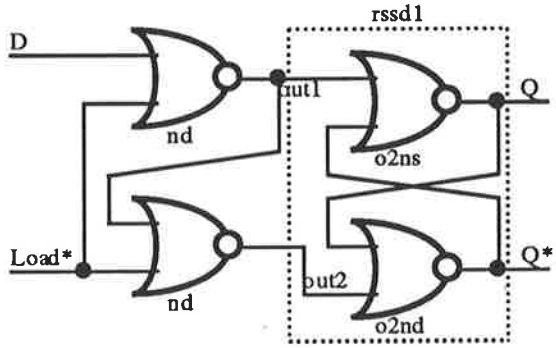


Figure 5.20: Circuit diagram for data latch "jsd"

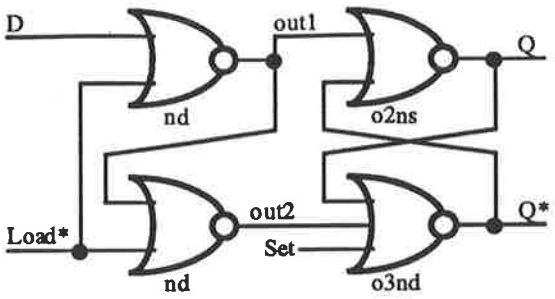


Figure 5.21: Circuit diagram for data latch "jpsd"

5.4 Pass Transistors

Four different types of pass transistor arrangements are used in the construction of D flip-flops within the buffer chip. More specifically, double depletion type transistors are utilised to allow the use of two-phase non-overlapping clock to simplify the D flip-flop designs. All pass transistors in this section are given the instance name "p" in the VHDL structural descriptions. Due to the different needs of clamping diodes at the inputs and the strength of the input signals, four variations of the pass transistors are presented. Their circuit diagrams are shown in Figure 5.22 to Figure 5.25 respectively. Note that all diodes are constructed by connecting the source and the drain of the DFETs together.

Figure 5.22 shows the configuration for a double pass transistor with clamping diodes at both inputs. The clamping diode is used to limit the input voltage swing from two diode drops to a single one so that the output swing is also clamped. The reason for a clamped output is to increase the speed of the logics at the following stage. This arrangement is used when both inputs are driven by normal sized DCFL, SDCFL or SBFL gates with no other fan-outs.

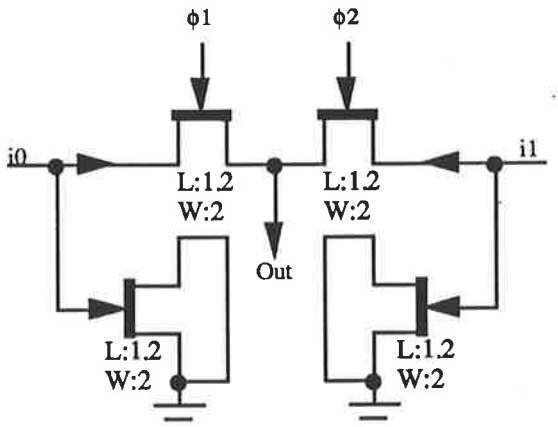


Figure 5.22: Circuit diagram for double pass transistors "pcc" with clamping diodes at both inputs

Figure 5.23 shows the configuration for a double pass transistor with a clamping diode at the input i1. This arrangement is used when input i0 is already clamped or driven by UBFL while input i1 is driven by normal sized DCFL, SDCFL or SBFL gates with no other fan-outs.

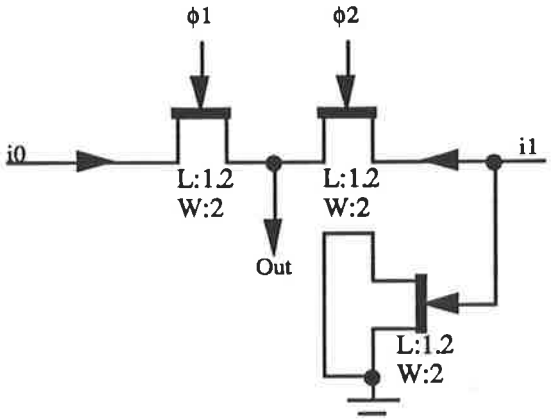


Figure 5.23: Circuit diagram for double pass transistors "pc" with one clamping diode

Figure 5.24 shows the configuration for a double pass transistor with no clamping diode at either inputs. This arrangement is used both inputs are already clamped or driven by UBFL.

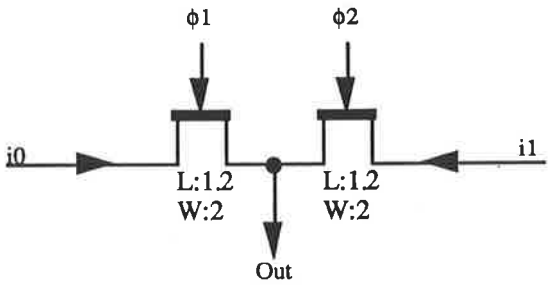


Figure 5.24: Circuit diagram for double pass transistors "p" with no clamping diode

Figure 5.25 shows the configuration for a double pass transistor with no clamping diode at either inputs. Note that this arrangement is the same as the one in Figure 5.24 except it is only used when both inputs are driven by doubled sized UBFL gates. This ensures the pass transistors are not weakened by the effect of backgating where the input signals leak through the output.

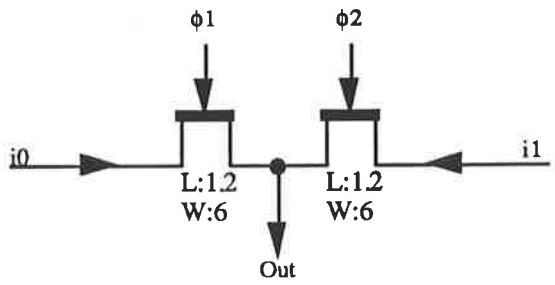


Figure 5.25: Circuit diagram for triple sized double pass transistors "p3" with no clamping diode

5.5 D Flip-Flops

Seven different D flip-flops are used due to various output load, power and speed requirements. They are given the instance name "l" for normal D flip-flops and "f" for differential ones. Clamping diodes are included to limit the voltage swing to DCFL level in order to increase speed and reduce noise injection to the power busses. All diodes are constructed by connecting the source and the drain of the DFETs together as illustrated in Section 5.4.

Figure 5.26 shows a D flip-flop realised by pure DCFL gates. This configuration should only be used in situations where both Q and Q* outputs have less than three fan-outs with a minimum interconnect length (ie. less than 30μm) as both outputs have weak driving capabilities.

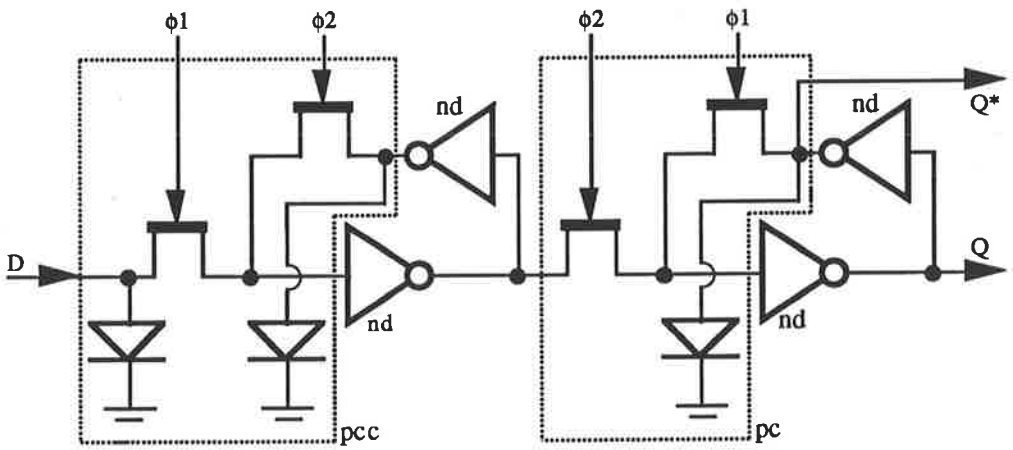


Figure 5.26: Circuit diagram for D flip-flop "ldddd"

In situations where a stronger output driving capability is only required at the output Q, the configuration as shown in Figure 5.27 should be used where Q is driven by an SDCFL inverter. This increases the maximum fan-out at Q to four or an equivalent capacitive load of 80fF while the maximum load at output Q* remains unchanged.

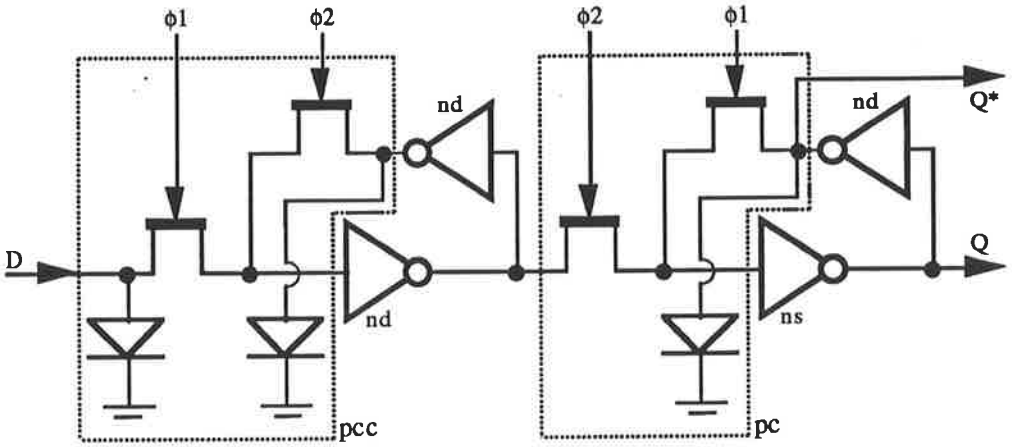


Figure 5.27: Circuit diagram for D flip-flop "lddsd"

If a strong output driving capability at both output Q and Q* is required, the configuration as shown in Figure 5.28 should be used where both outputs are driven by SDCFL inverters. This increases the maximum fan-out for both outputs to four or an equivalent capacitive load of 80fF.

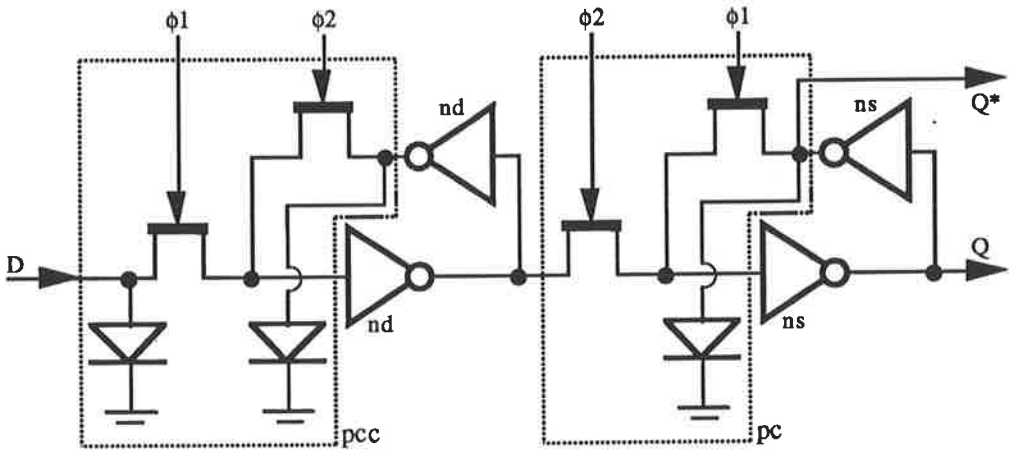


Figure 5.28: Circuit diagram for D flip-flop "lddss"

In situations where there is a long interconnect line or a large fan-out (eg. five) at the output Q, the configuration as shown in Figure 5.29 can be used where Q is driven by a UBFL inverter. This increases the maximum fan-out at Q to six or an equivalent capacitive load of 100fF while the maximum load at output Q* remains that of a DCFL inverter.

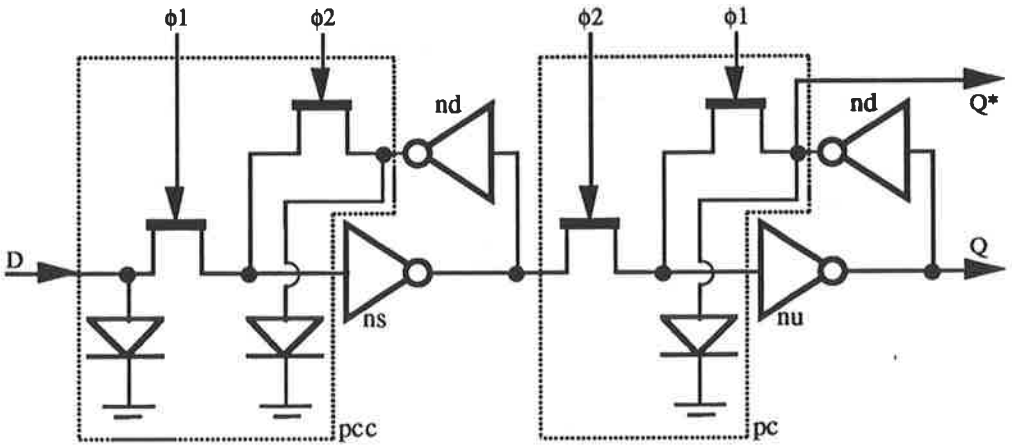


Figure 5.29: Circuit diagram for D flip-flop "lsdud"

If there are long interconnect lines or large fan-outs at both outputs (Q and Q*), the configuration as shown in Figure 5.30 can be used where both outputs are driven by UBFL inverters. This increases the maximum fan-out to six or an equivalent capacitive load of 100fF.

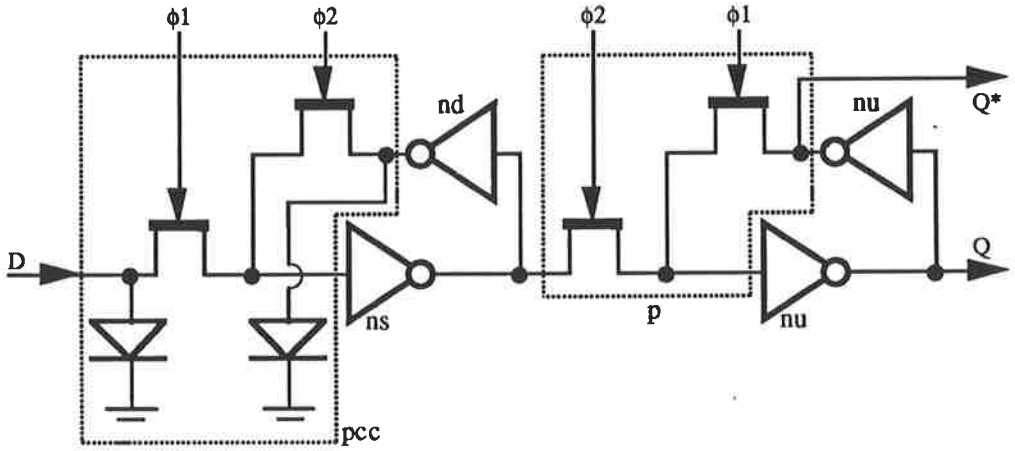


Figure 5.30: Circuit diagram for D flip-flop "lsduu"

If speed and the skew in the outputs (Q and Q*) are of major concern, then the configuration as shown in Figure 5.31 is best suited. It is a differential D flip-flop so there is little/no skew in the two outputs and they are driven by SBFL inverters. This is the fastest arrangement available without sizing the gates to a larger ratio. Both outputs can drive over seven fan-outs or an equivalent capacitive load of 160fF. However, it is associated with a large power dissipation which makes its use limited. Nevertheless, it is useful in critical paths where speed is of major concern.

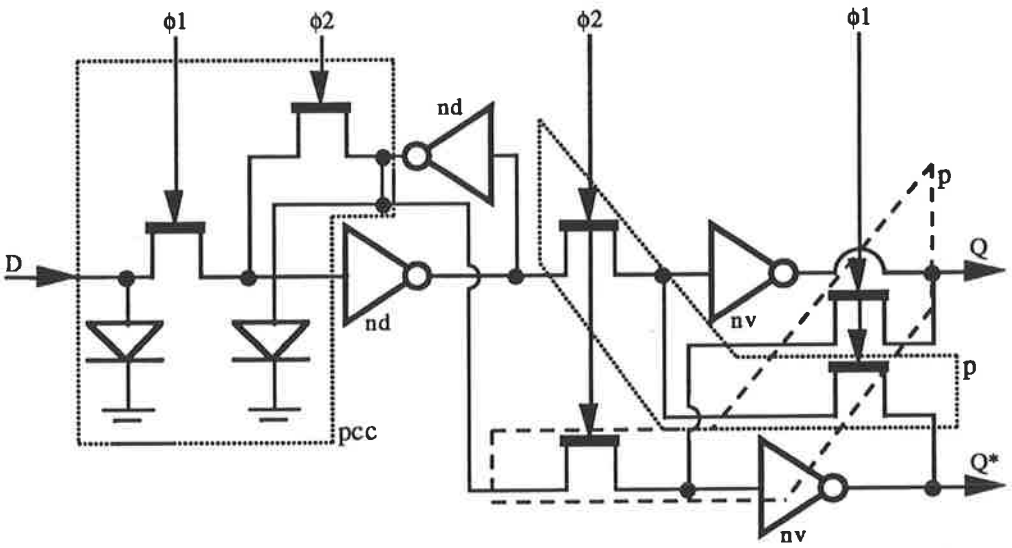


Figure 5.31: Circuit diagram for differential D flip-flop "fddvv"

For synchronous signals that have an extremely long interconnect (such as the ones between different functional blocks which could be as long as 1mm) and a large fan-out (eg. 100), the output driving capability of normal sized logic gates will be insufficient. In this case, triple sized SBFL inverters are used at the output (Q and Q*). Consequently, the pass transistors will have to be triple sized to avoid leakage due to the effect of backgating. This configuration is illustrated in Figure 5.32. Note that the first stage of the D flip-flop is composed of two normal sized SBFL inverters to minimise the noise injection to VDD as they are connected in series. It also provides sufficient current drive for the second stage.

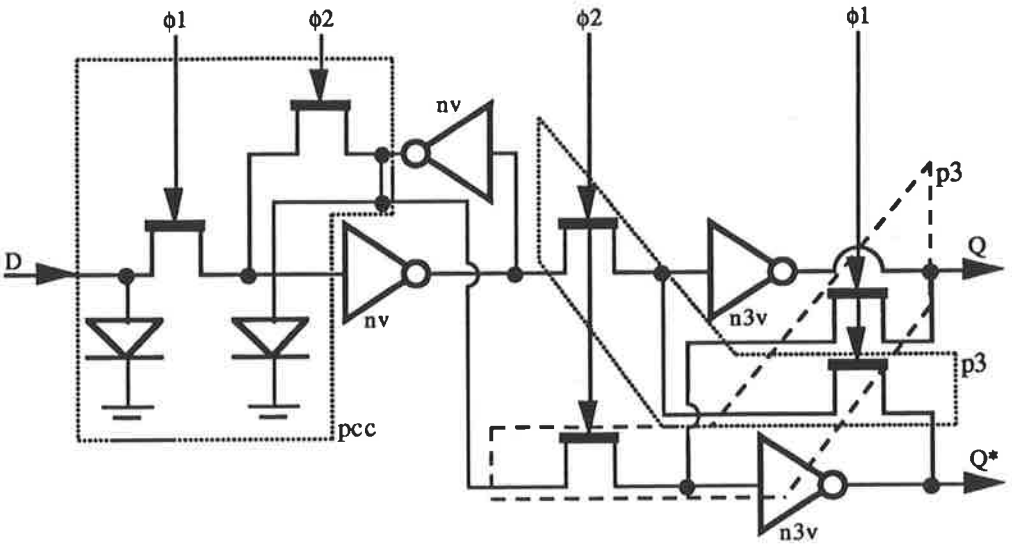


Figure 5.32: Circuit diagram for differential D flip-flop "fvv3v3v"

To drive an exceptionally large output load (such as clocks and reset lines), the buffering strategy as described in [ESHRAGHIAN] can be employed. This is illustrated in Figure 5.33 where all SBFL inverters are driven by a half-sized one in the preceding stage. The number on top of the inverter indicates the ratio to the previous one. Although it has been shown in [ESHRAGHIAN] that a ratio of e provides the minimum delay, it does not guarantee a maximum operational speed.

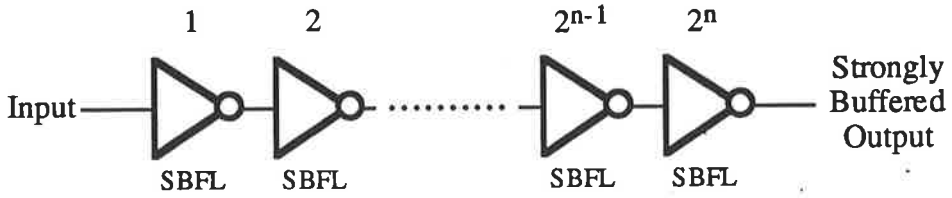


Figure 5.33: Buffering scheme used to drive exeptionally long lines

5.6 Multiplexers

Four different types of multiplexers are described with instance name "mux":

Figure 5.34 shows the simplest configuration for a multiplexer whose truth table is shown in Table 5.3. It utilises only 2-input DCFL nor gates and hence the output current drive is the same as a DCFL 2-input nor gate.

Input				Output
s0	s1	i0	i1	o
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	X	X	i0
1	0	X	X	i1
1	1	X	X	1

Table 5.3: Truth table of the multiplexer "mux"

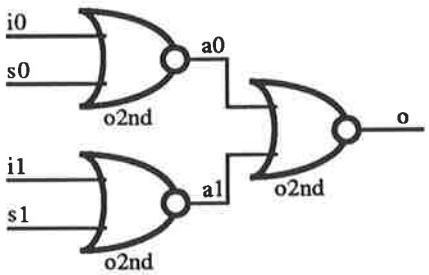


Figure 5.34: Circuit diagram for multiplexer "mux"

If reset is required, the configuration as shown in Figure 5.35 can be used. The truth table of the circuit is shown in Table 5.4. Note that the reset signal on the last DCFL nor gate is active high. Since output is driven by a DCFL 3-input nor gate, it can only support one fan-out with a minimum interconnect length.

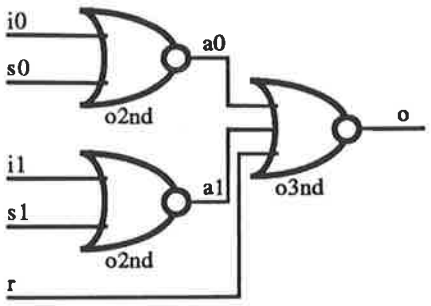


Figure 5.35: Circuit diagram for multiplexer "muxr"

Input					Output
r	s0	s1	i0	i1	o
1	X	X	X	X	0
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	X	X	i0
0	1	0	X	X	i1
0	1	1	X	X	1

Table 5.4: Truth table of the multiplexer "muxr"

In situations where a stronger output driving capability is required, the circuit as shown in Figure 5.36 can be used. The only difference between this circuit and the one presented in Figure 5.35 is that the output in the latter one is driven by an SDCFL 3-input nor gate. As a result, it can drive three fan-outs or an equivalent capacitive load of 60fF at the output.

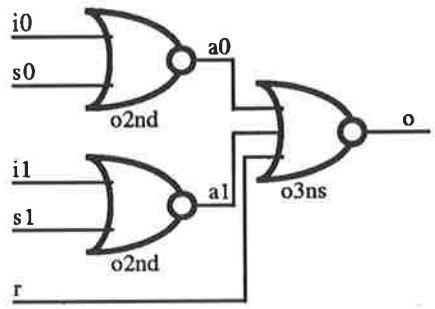


Figure 5.36: Circuit diagram for multiplexer "muxrdds"

If an active high preset is required, the circuit as shown in Figure 5.37 can be used. The truth table of the circuit is shown in Table 5.5. Note that only DCFL gates are used in its construction. Consequently the output can only drive two fan-outs or an equivalent capacitive load of 20fF.

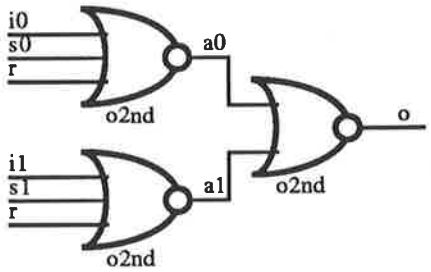


Figure 5.37: Circuit diagram for multiplexer "muxs"

Input					Output
r	s0	s1	i0	i1	o
1	X	X	X	X	1
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	X	X	i0
0	1	0	X	X	i1
0	1	1	X	X	1

Table 5.5: Truth table of the multiplexer "muxs"

Chapter 6. Implementing the Control Logic

This chapter describes the design details of the control logic within the buffer chip which can be divided into three main functional blocks: the **input control**, the **buffer manager** and the **output control**. Other modules of the chip were designed by Mr. Jens Jakobsen at Jydsk Telefon, Denmark. The floor plan of the entire chip is shown in Figure 6.1.

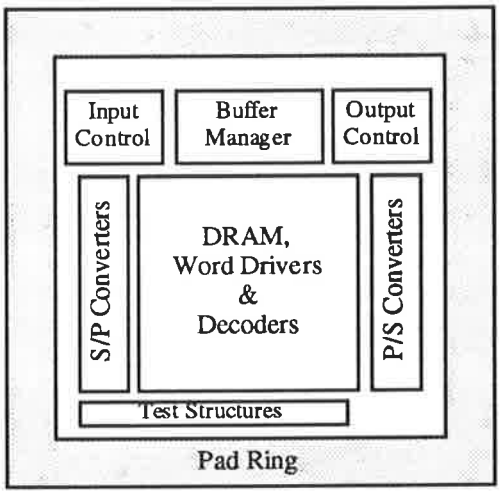


Figure 6.1: Floor plan of the buffer chip

Throughout the whole chip, two-phase non-overlapping clocks are used (which are derived from the input single-phase clocks by level shifting and complementing) to simplify the D flip-flop designs (see section 5.5). The two-phase clocks have 0V for logic high and -1.5V for logic low to ensure the DFETs, which are used as pass transistors, can be switched off properly. As discussed in section 3.4.1, an isochronous clocking strategy is used to simplify the clock distribution and minimise the clock skew. More specifically, the input serial-to-parallel (S/P) converters and the input control use the two-phase clock derived from the input data interface while the buffer manager, the output control and the output parallel-to-serial (P/S) converters use the two-phase clock derived from the external input clock (which runs at half the input clock speed). The signal communications between the input control and

the buffer manager as well as that between the buffer manager and the output control are done asynchronously. That is, all incoming signals must first pass through two Data flip-flops to get synchronised with the local clock as described in [CMP] to avoid meta-stability.

As mentioned in Chapter 3, the input sections of the buffer chip have to run at 600MHz to match the input data rate of 4.8Gb/s. However, the output sections will only need to run at 300MHz as the output data rate is only 2.4Gb/s. This implies the input control must operate at 600MHz to generate proper control signals to the input serial-to-parallel converters whereas the output control will only need to operate at 300MHz to control the output parallel-to-serial converters. The buffer manager, on the other hand, can be designed to operate at both 600MHz or 300MHz as it needs to interface with both the input and output control modules. It is decided in [JTFEB93] that the buffer manager should operate at 300MHz to enable more flexibility in the hardware design process. The detailed design of the three modules are presented in section 6.1, 6.2 and 6.3 respectively.

6.1 The Input Control

The input control detects the arrival of non-idle cells and sends out input block request signals (**IBR0** or **IBR1**) to the buffer manager according to their cell priorities [CHU2]. Depending upon the decision of the buffer manager, the request will either be granted (which will be accompanied by a 5-bit block address) or not granted. If the request is granted, the input control will generate the full 7-bit write address (**WA0..6**) and a write enable (**WE**) signal to the dynamic memory (**DRAM**) together with an input convert (**IC**) signal to the input serial-to-parallel (S/P) converters at the appropriate moments (see later). Otherwise, no action will be taken and the incoming cell is discarded. Figure 6.2 shows the interface of the input control and Table 6.1 provides an indication of the signals' function.

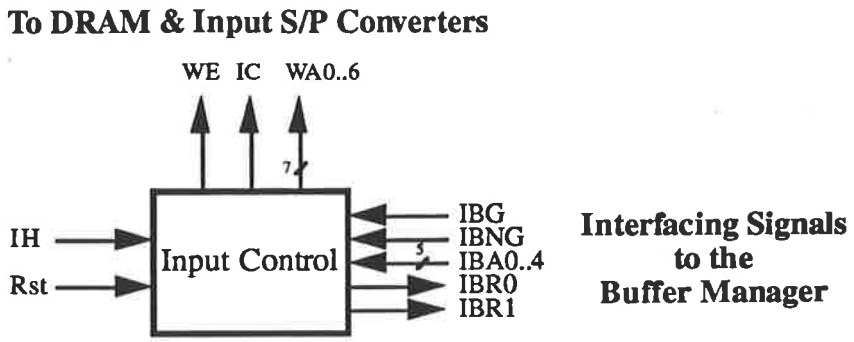


Figure 6.2: Interfacing the input control

In order to achieve a lower cell loss probability for high priority cells, a two-level priority scheme is used such that when the DRAM is at or more than three-quarters full (which corresponds to 24 cells), any further incoming low priority cells will be discarded whereas high priority cells are only discarded only when the DRAM is full.

Signal Name	Description
Internal Header (IH)	Internal Header of the incoming cell
Reset (Rst)	Resets the entire input control module
Input Block Request 0 (IBR0)	Request write permission for a high priority cell from the buffer manager
Input Block Request 1 (IBR1)	Request write permission for a low priority cell from the buffer manager
Input Block Granted (IBG)	Granting write permission for the incoming cell
Input Block Not Granted (IBNG)	Not granting write permission for the incoming cell
5-Bit Input Block Address (IBA0..4)	Unlatched top 5-bit write address
7-Bit Write Address (WA0..6)	The full 7-bit write address to memory
Input Convert (IC)	Instruct the input S/P converters to load its contents to the DRAM
Write Enable (WE)	Enabling memory write sequence

Table 6.1: Description of interfacing signals in the input control

In order to maximise memory access time and to preserve a reasonable chip geometry, it is desirable to split the incoming cells into sub-cells. For a 32-cell memory, a four sub-cell configuration of 14 bits each would result in a memory size of 126 bits by 128 words. Consequently, there are 56 DRAM cells assigned for each parallel line and since there are only 53 bits of valid data available, the extra three DRAM cells are filled by storing some bits twice. In this case, it is decided in [JTFEB93] that the first sub-cell (which corresponds to the 1st to the 14th bit on one incoming line of the cell) contains no redundant bit to ensure there is sufficient time to process the priority information while the other three sub-cells (which corresponds to the 14th to 27th, 27th to the 40th, and 40th to the 53rd bits respectively) each contains one extra bit from the previous sub-cell. Consequently, the 14th, 27th and the 40th bit in each parallel line are stored twice in the DRAM. Note that special signalling pulses (**Q11**, **Q26**, **Q39**, and **Q52**) are required to indicate the end of subcells. The operation of the input control can be summarised in the flow chart as shown in Figure 6.3.

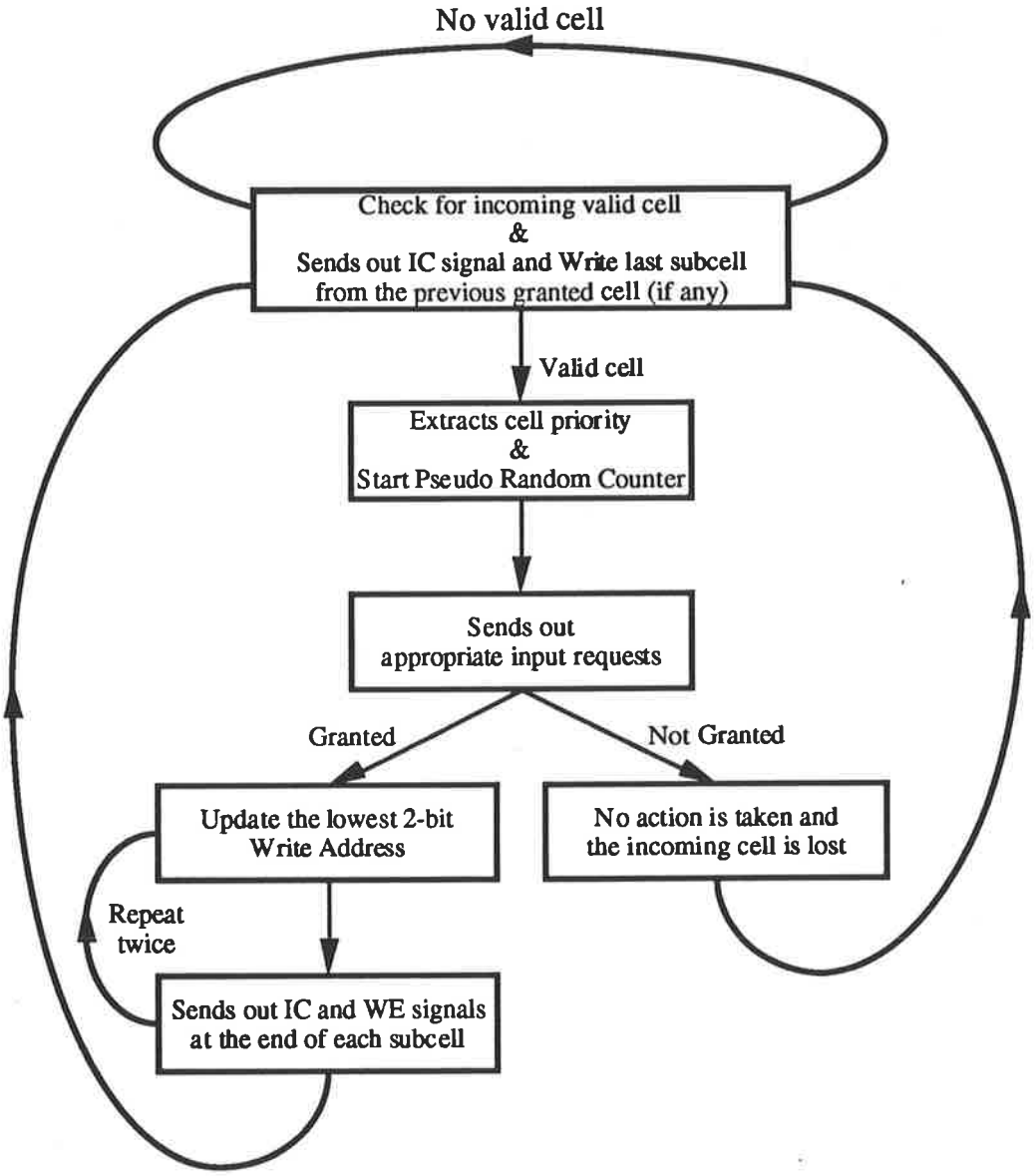


Figure 6.3: Flow chart summarising the functionality of the input control

The input control can be realised in five main modules as shown in Figure 6.4. Table 6.2 describes the functionality of each module.

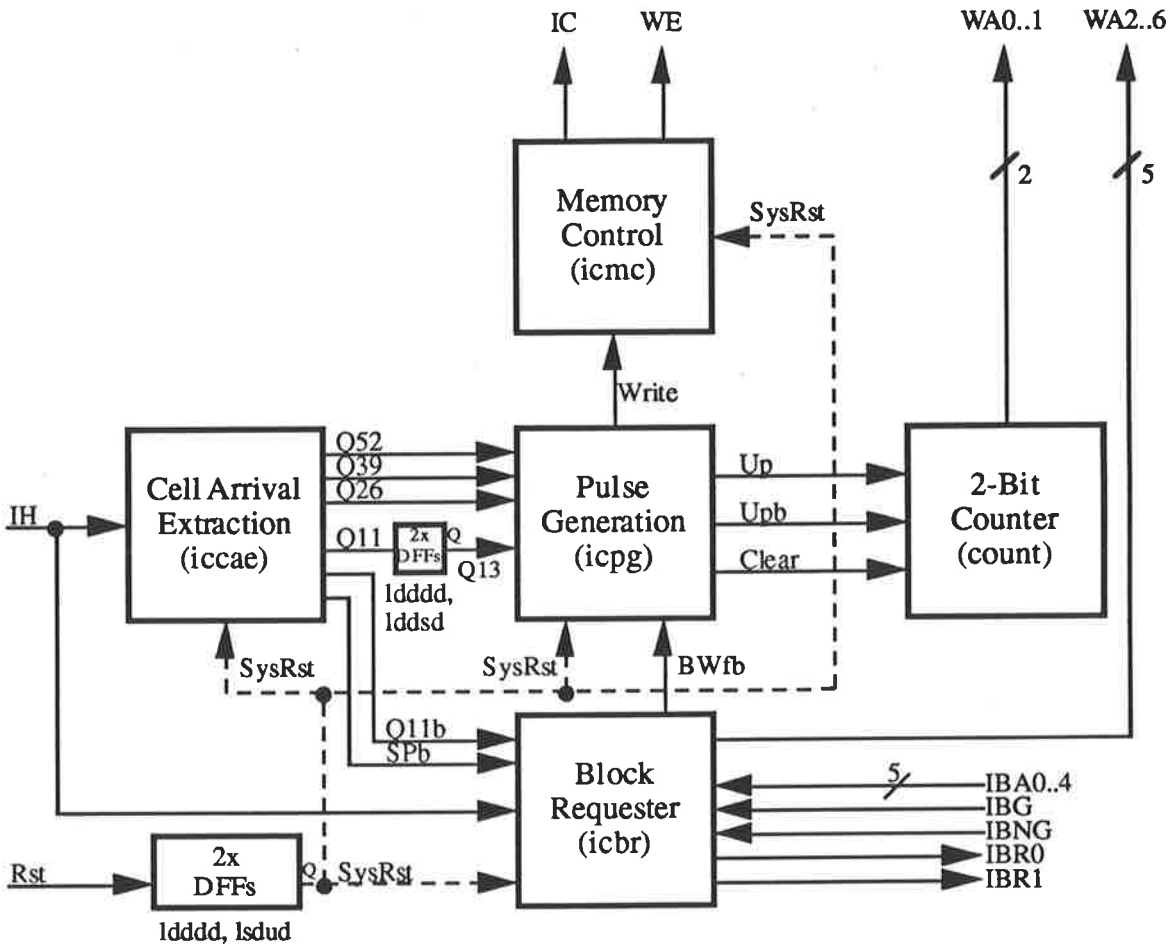


Figure 6.4: Block diagram representation of the input control

Module Name	Functionality
Cell Arrival Extraction (iccae)	• Extracts a start pulse from the incoming cell and indicates the end of sub-cells
Block Requester (icbr)	• Extracts cell priorities • Sends out appropriate input block requests • Latches out the top 5-bit write address
Pulse Generation (icpg)	• Generates inputs for the 2-bit counter as well as appropriate control signals for the input S/P converters and memory write process
Memory Control (icmc)	• Generates an input convert signal to the input S/P converters and a memory write enable signal to the DRAM from a **Write** pulse from the Pulse Generation module
2-Bit Counter (count)	• Generates the lowest 2-bit write address

Table 6.2: Functional description of the modules within the input control

6.1.1 The Cell Arrival Extraction Module

The core of the cell arrival extraction module is a 6-bit pseudo random counter [JTMOB93] which is chosen for its simplicity and high speed operation. Pipelined decoders are included to indicate the end of sub-cells which are used to control the input S/P converters as well as the memory write process. The schematic of the module is shown in Figure 6.5 and the resulting state transition table shown in Table 6.3. A timing diagram of the cell arrival extraction module is shown in Figure 6.6. Note that the Start Pulse (**SP**) is only one clock period long and is generated two clock periods after the arrival of a valid (non-idle) internal header (**IH**).

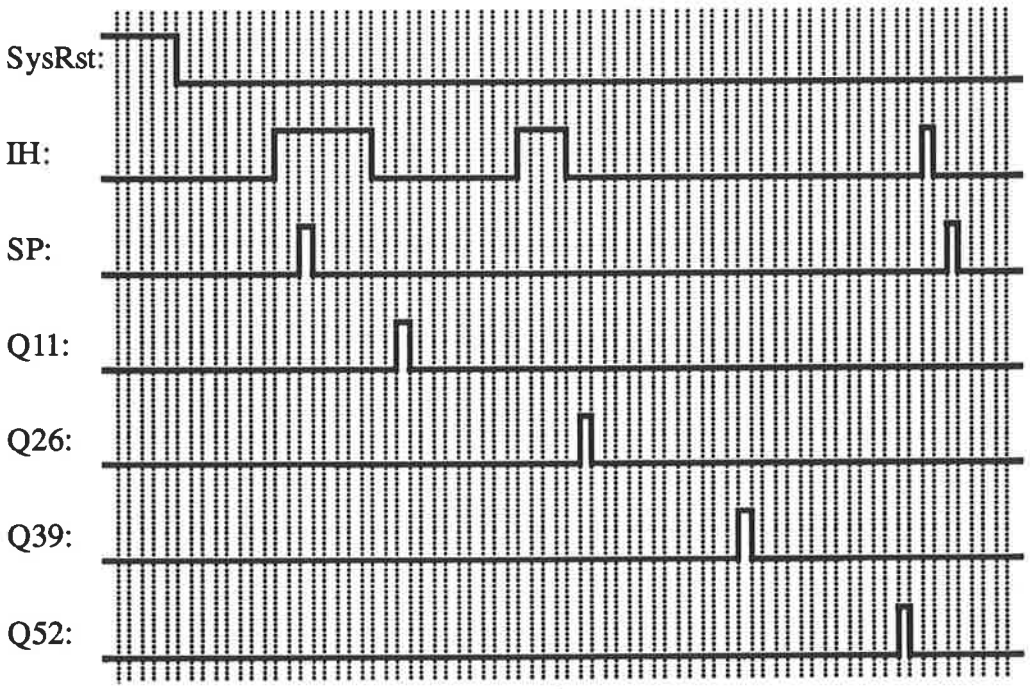
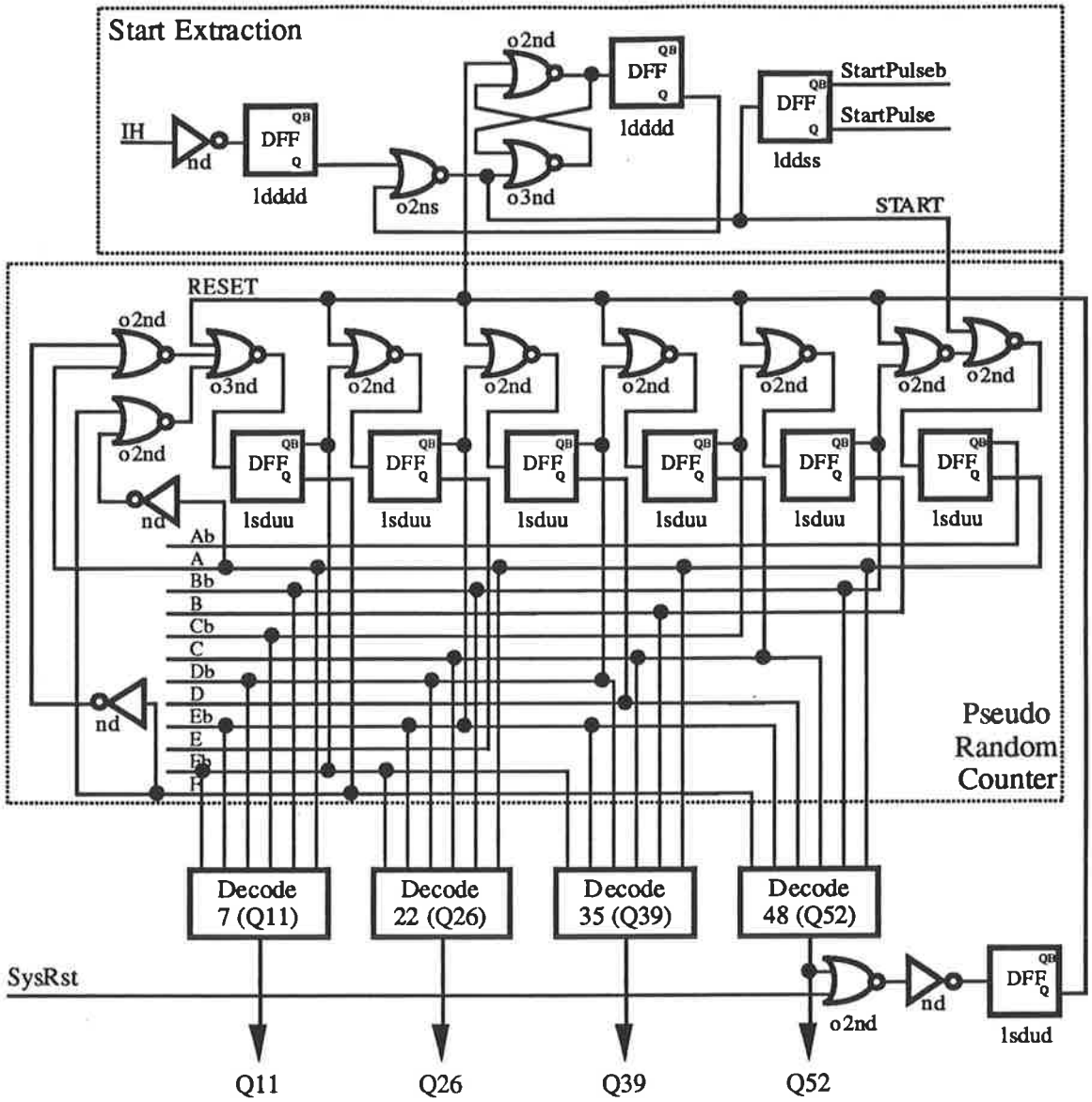


Figure 6.6: Timing diagram of the cell arrival extraction module



where the decoders are realised as:

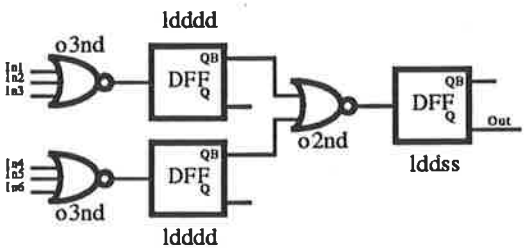


Figure 6.5: Schematic of the cell arrival extractor

Remarks	S5	S4	S3	S2	S1	S0	HEX
Stable State	0	0	0	0	0	1	01
Counting States	0	0	0	0	0	0	00
	1	0	0	0	0	1	21
	1	1	0	0	0	1	31
	1	1	1	0	0	1	39
	1	1	1	1	0	1	3D
	1	1	1	1	1	1	3F
	1	1	1	1	1	0	3E
	0	1	1	1	1	0	1E
	1	0	1	1	1	0	2E
	0	1	0	1	1	0	16
	1	0	1	0	1	0	2A
	0	1	0	1	0	0	14
	1	0	1	0	1	1	2B
	1	1	0	1	0	0	34
	0	1	1	0	1	1	1B
	0	0	1	1	0	0	0C
	1	0	0	1	1	1	27
	1	1	0	0	1	0	32
	0	1	1	0	0	0	18
	1	0	1	1	0	1	2D
	1	1	0	1	1	1	37
	1	1	1	0	1	0	3A
	0	1	1	1	0	0	1C
	1	0	1	1	1	1	2F
	1	1	0	1	1	0	36
	0	1	1	0	1	0	1A
	1	0	1	1	0	0	2C
	0	1	0	1	1	1	17
	0	0	1	0	1	0	0A
	1	0	0	1	0	0	24
	0	1	0	0	1	1	13
	0	0	1	0	0	0	08
	1	0	0	1	0	1	25
	1	1	0	0	1	1	33
	1	1	1	0	0	0	38
	0	1	1	1	0	1	1D
	0	0	1	1	1	1	0F
	0	0	0	1	1	0	06
	1	0	0	0	1	0	22
	0	1	0	0	0	0	10
	1	0	1	0	0	1	29
	1	1	0	1	0	1	35
	1	1	1	0	1	1	3B
	1	1	1	1	0	0	3C
	0	1	1	1	1	1	1F
	0	0	1	1	1	0	0E
	1	0	0	1	1	0	26
	0	1	0	0	1	0	12
	1	0	1	0	0	0	28
	0	1	0	1	0	1	15
	0	0	1	0	1	1	0B
	0	0	0	0	0	1	01

Table 6.3: State transition table for the pseudo random counter

6.1.2 The Block Requester

The block requester is realised as shown in Figure 6.7. It takes the internal header (**IH**) together with the complement of the Start Pulse (**SPb**) as inputs to extract the cell priority. The appropriate input block request is then sent out by an RS flip-flop which gets reset on the arrival of either the input block granted (**IBG**) or the input block not granted signal (**IBNG**) from the buffer manager. If the request is granted, the complement of the block write signal (**BWfb**) is sent out from the 2-stage latch after 11 clock cycles on the arrival of the cell and remains low until the cell is fully written to the DRAM. Otherwise, **BWfb** remains high. On the 15th clock cycle (**Q15**), the top 5-bit write address is latched out to generate the current write address whether the input request is granted or not. A timing diagram showing the operation of the block requester is shown in Figure 6.8.

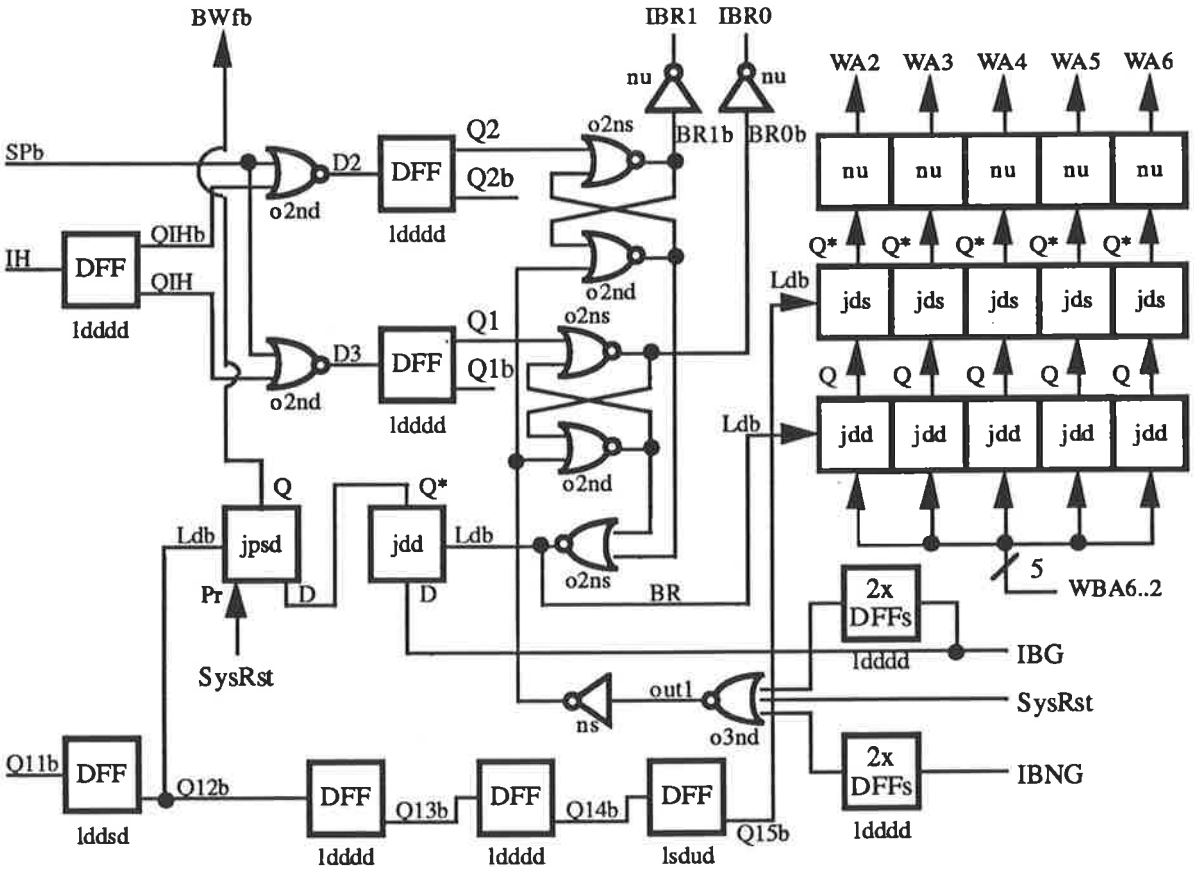


Figure 6.7: Schematic of the block requester

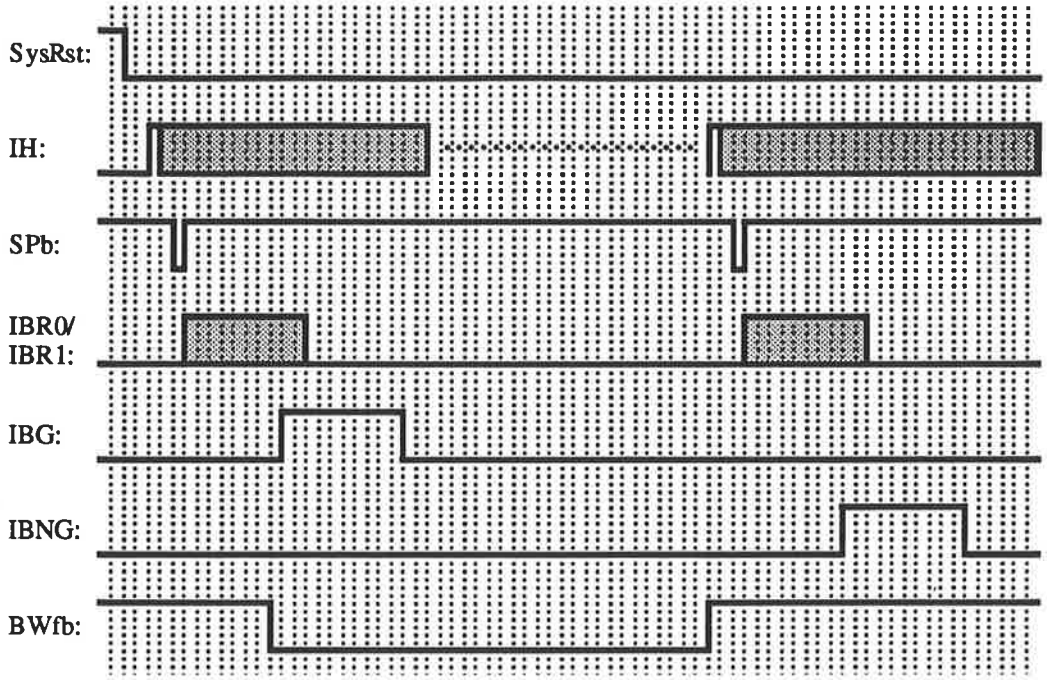


Figure 6.8: Timing diagram of the block requester

6.1.3 The Pulse Generation Module

The pulse generation module is simply a set of pipe-lined decoders. If an input block request is granted, the module takes the four timing pulses (**Q13**, **Q26**, **Q39** and **Q52**) from the cell arrival extraction module together with the complement of the block write signal (**BWfb**) from the block requester to generate four **Write** pulses to the memory control module as well as three count-up pulses (**Up**) and the clear signal (**Clear**) to the 2-bit counter to generate/update the lower 2-bit write address. On the other hand, if the input request is not granted, the same inputs to the 2-bit counter will be generated but no **Write** pulses are produced. The module can be implemented as shown in Figure 6.9 with the timing diagram shown in Figure 6.10.

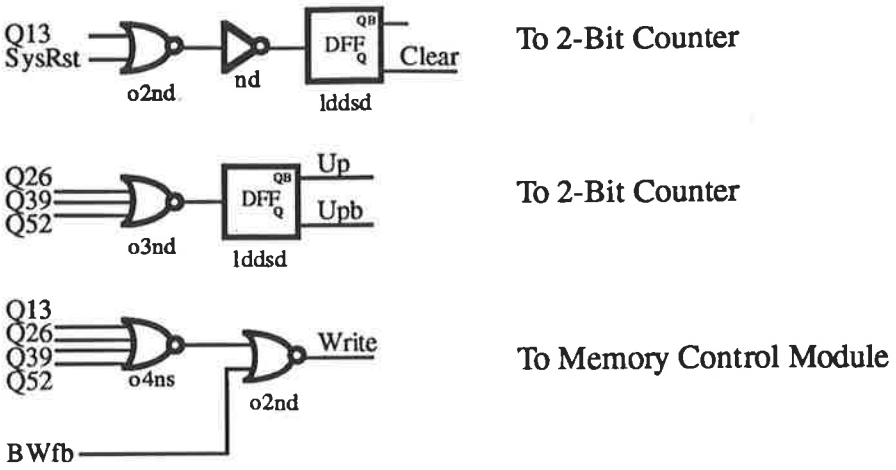


Figure 6.9: Schematic of the pulse generation module

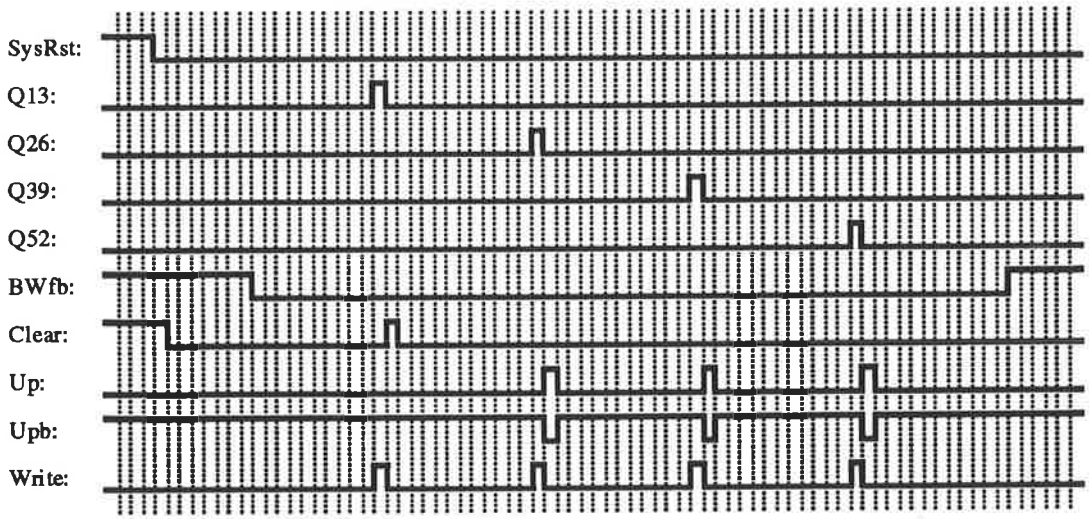


Figure 6.10: Timing diagram of the pulse generation module for a granted input block request

6.1.4 The Memory Control Module

The memory control module receives a **Write** pulse of one-clock period long from the block requester and sends out an input convert signal (**IC**) to the S/P converters after one clock period to load the parallel data out to the DRAM. It also generates a write enable signal (**WE**) of 11 clock cycles long two clock periods after **IC**. It is realised as shown in Figure 6.11 with its timing information shown in Figure 6.12.

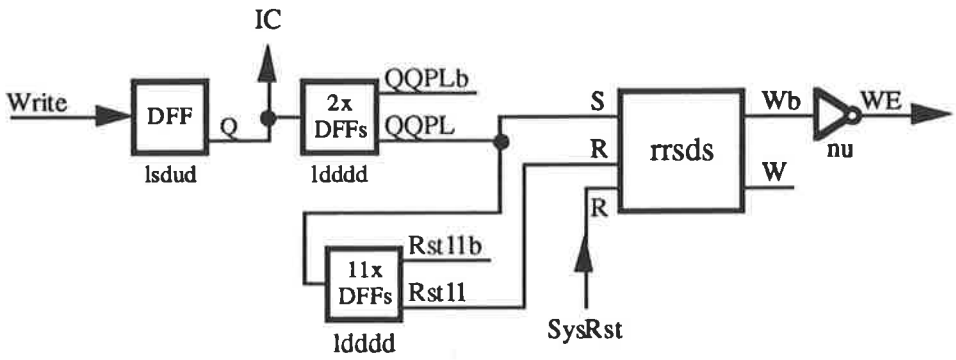


Figure 6.11: Schematic of the memory control module

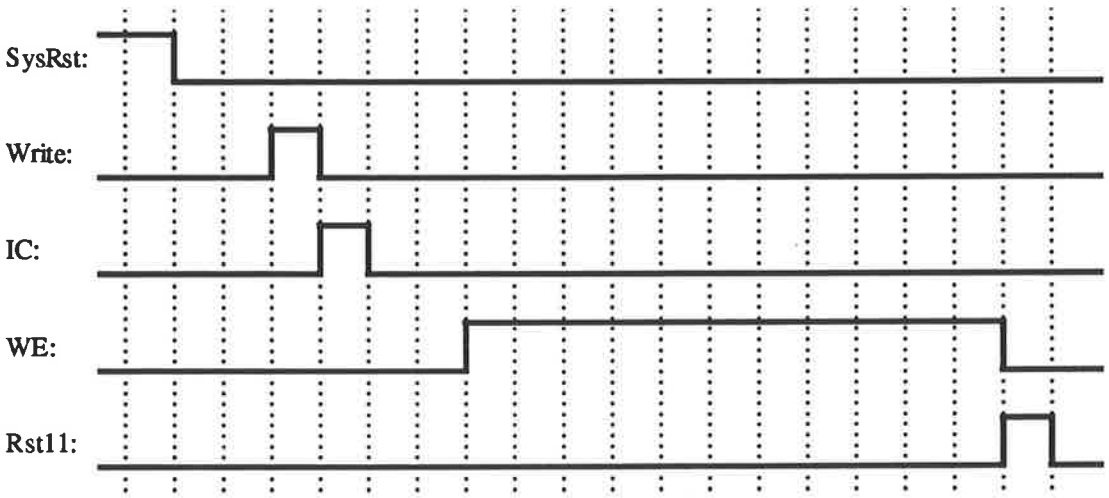


Figure 6.12: Timing diagram of the memory control module

6.1.5 The 2-Bit Counter

This counter is realised as a synchronous counter. However, due to the fact that it will be cleared before getting to state 00, the logic can be simplified by mapping the next state of 11 to XX. The circuit diagram of the counter is shown in Figure 6.13 and the resulting state transition table is shown in Table 6.4.

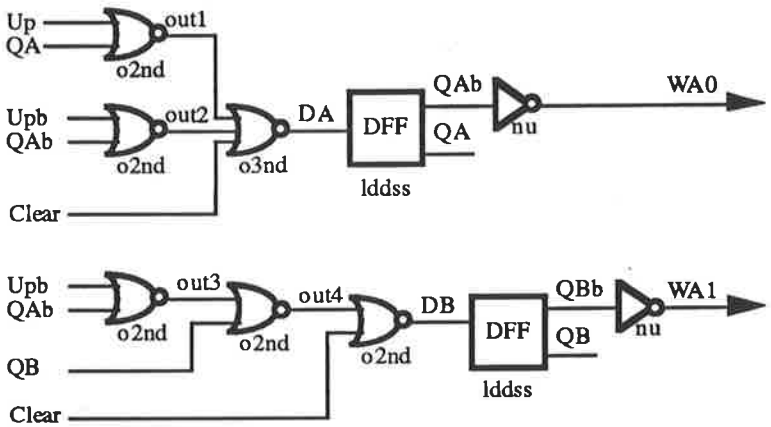


Figure 6.13: Circuit diagram of the 2-bit counter

Present State		Next State (Up=0)		Next State (Up=1)	
$WA1_t$	$WA0_t$	$WA1_{t+1}$	$WA0_{t+1}$	$WA1_{t+1}$	$WA0_{t+1}$
0	0	0	0	0	1
0	1	0	1	1	0
1	0	1	0	1	1
1	1	1	1	1	0

Table 6.4: State transition table of the simplified 2-bit counter

Figure 6.14 shows the layout of the entire input control circuit in **Magic** to verify the effectiveness of the layout style employed. In digital VLSI designs, it is customary to over-design a circuit to operate at a higher speed to account for any unforseen reasons that might render the circuit to operate slower. In this case, a 50% over-design factor is chosen due to the immaturity of the process employed. Hence, the entire input control is simulated at 900MHz. Furthermore, all output nodes are connected to a capacitive load of 1pF (where in reality it translates to a long interconnect line which is connected to other functional blocks) except for the input convert signal (**IC**) which has a capacitive load of 200fF and a 30μm wide EFET connected as external load (where in reality this signal is directly connected to an adjacent differential D flip-flop "fvv3v3v").

In the simulations, the first and the third cell are high priority cells whereas the second cell has a low priority. It can be seen that the first two block requests are granted and are assigned to a block address (**IBA**) and a block granted signal (**IBG**) by the buffer manager whereas the last block request is not granted (**IBNG** asserted). The **IRSIM** and **HSpice** simulation results are shown in Figure 6.15 and Figure 6.16 respectively.

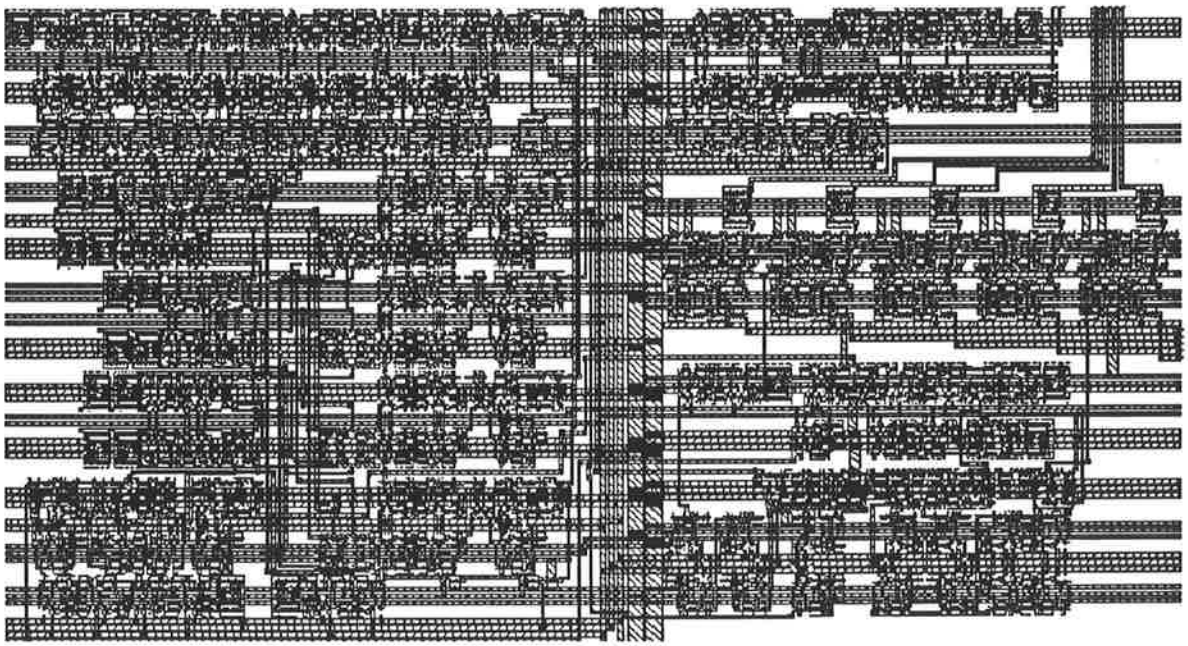


Figure 6.14: Layout of the input control in Magic

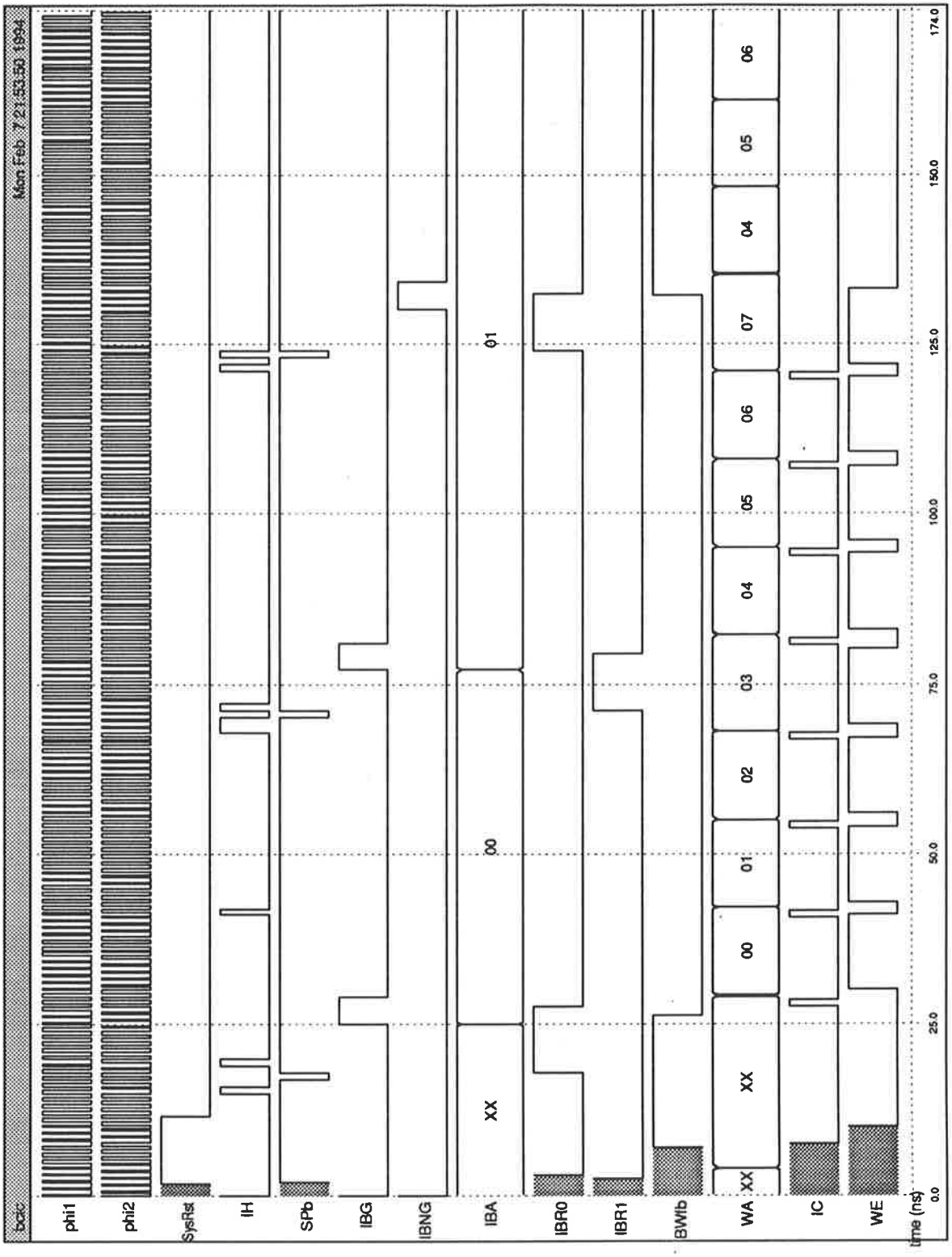


Figure 6.15: Simulation of the input control in IRSIM

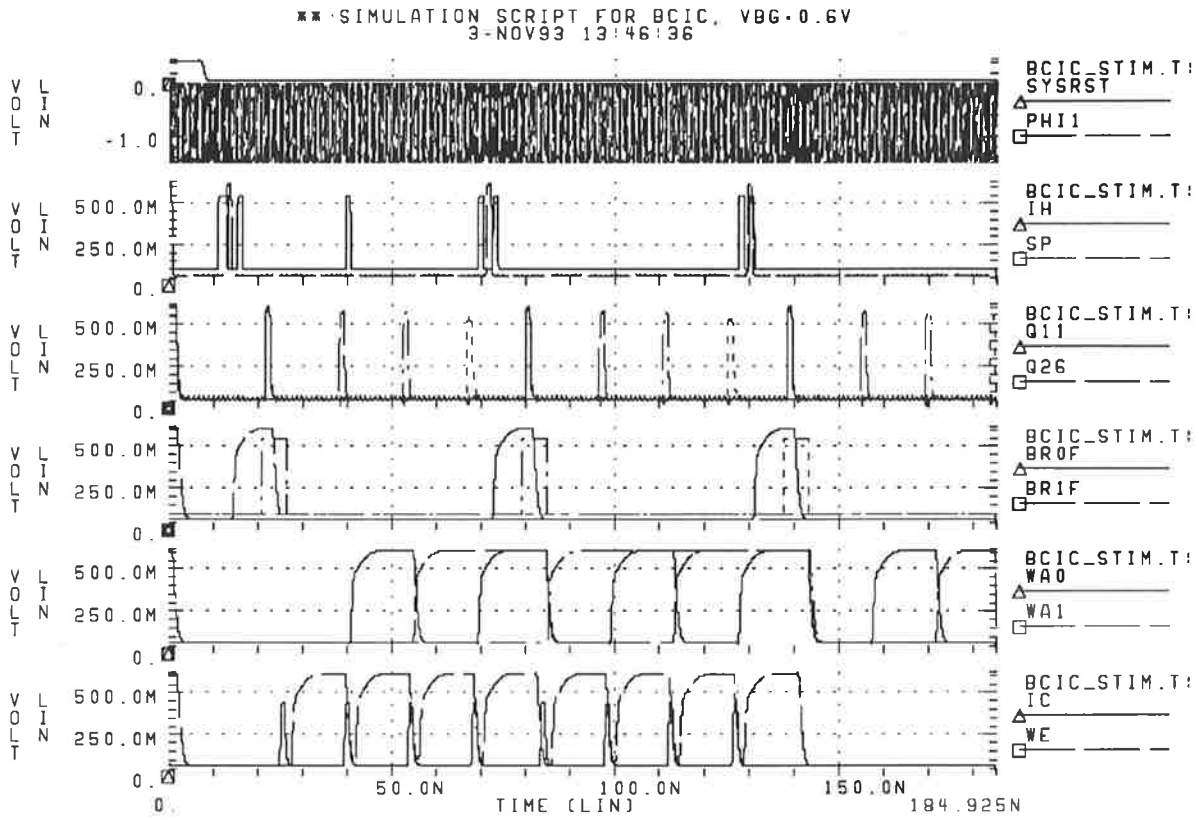


Figure 6.16(a): HSpice simulation of the input control at 900MHz

Note that all output nodes except IC are loaded with a 1pF capacitor

Key: Graph 1 shows the system reset signal (SYSRST) and the two phase clock (PHI1, PHI2)

Graph 2 shows the internal header (IH) and the start pulse (SP)

Graph 3 shows the four timing pulses Q11, Q26, Q39 and Q52 (Q39 and Q52 not shown in the legend)

Graph 4 shows the two input block request signals IBR0 and IBR1 which are denoted by BR0F and BR1F in the legend

Graph 5 shows the lowest 3-bit write address WA0, WA1 and WA2 (WA2 not shown in the legend)

Graph 6 shows the input convert (IC) and write enable (WE) signals

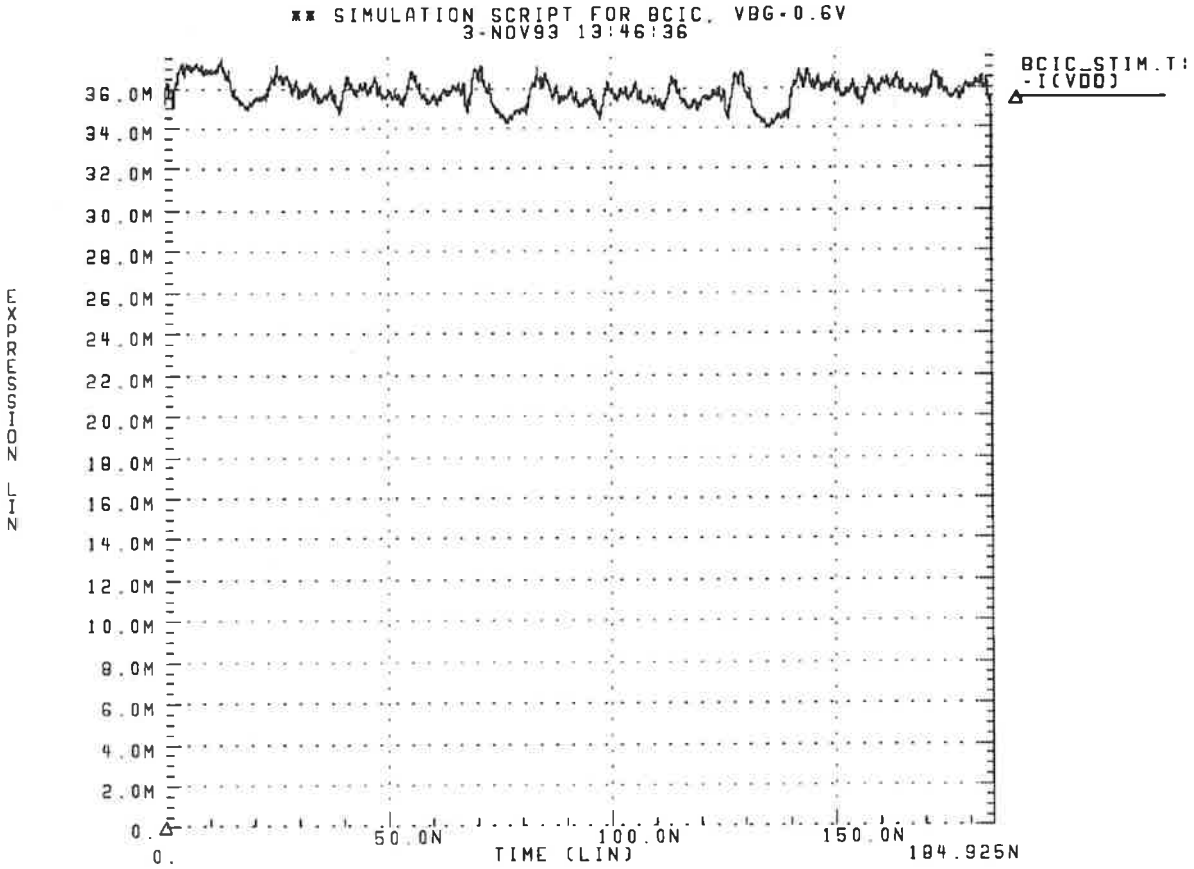


Figure 6.16(b): Current drawn by the input control at 900MHz for the simulation in figure 6.16(a)

It can be verified from Figure 6.16(b) that the logic families does in fact draw a fairly constant current from VDD. The mean deviation of current drawn by the input circuit is calculated to be 5.6%. This explains the fact that the low noise injected to the supply lines will not seriously affect the performance of the logic.

6.2 The Buffer Manager

The buffer manager processes input block requests (**IBR0**, **IBR1**) from the input control and output block requests (**OBR**) from the output control according to the status of the DRAM [CHU3]. High priority input block requests (**IBR0**) are always granted unless the DRAM is full. On the other hand, low priority input block requests (**IBR1**) are only granted when the DRAM is less than three-quarters (or equivalent to 24 cells) full. This ensures a lower cell loss probability is achieved for high priority cells. If an input request is granted, both the Start Pointer (**SP**) and the Queue Size Register (**QSR**) (which are responsible for generating the top 5-bit write address and keeping track of the number of stored cells in the DRAM respectively) will increment. Otherwise, both **SP** and **QSR** remain unchanged. Similarly, output block requests (**OBR**) are only granted if the DRAM is not empty. If an output block request is granted, the End Pointer (**EP**) (which generates the top 5-bit read address) increments while the Queue Size Register (**QSR**) decrements to indicate the DRAM has one more cell vacancy. A flow chart summarising the operation of the buffer manager is shown in Figure 6.17. Note that the buffer manager is designed to handle simultaneous arrival of both input and output block requests without getting into ambiguous states.

Figure 6.18 shows an interface of the buffer manager and Table 6.5 indicates the function of the interfacing signals. The buffer manager can be realised in eight main modules as shown in Figure 6.19. Table 6.6 describes the functionality of each module.

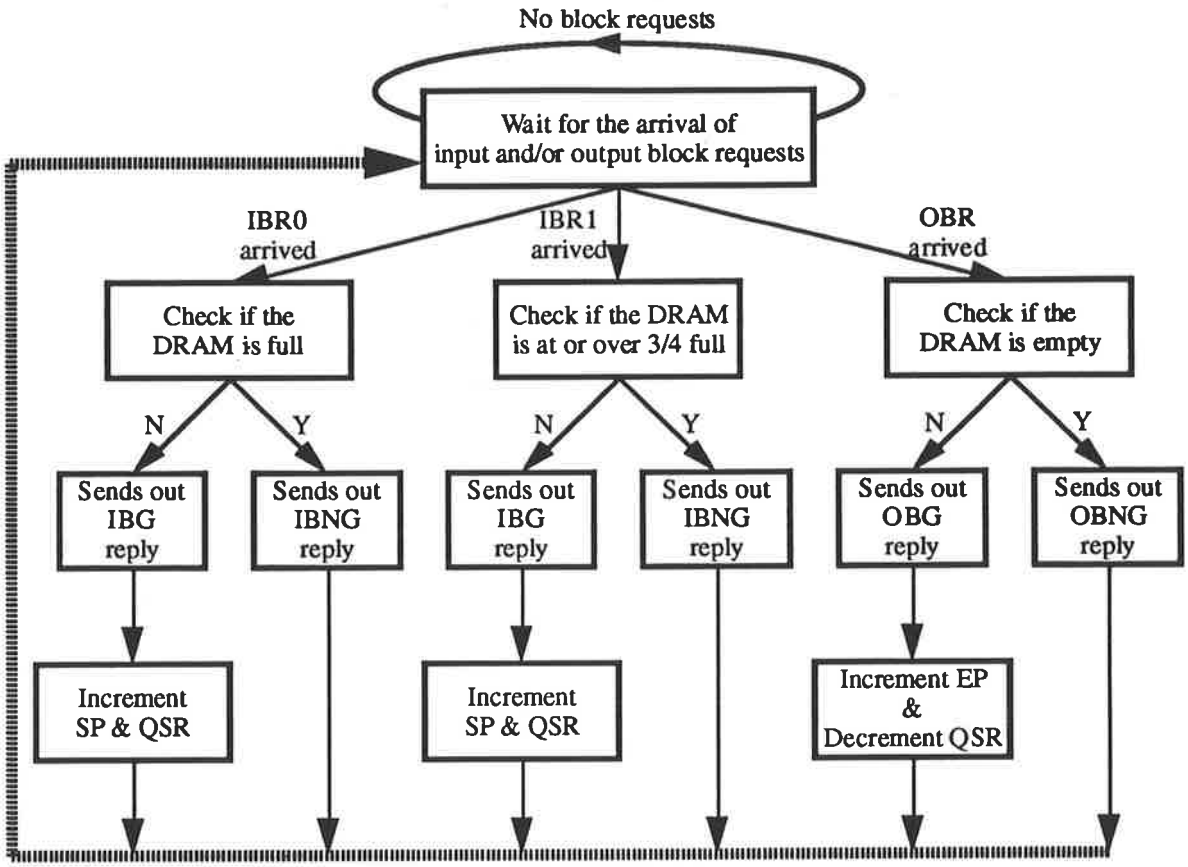


Figure 6.17: Flow chart summarising the functionality of the buffer manager

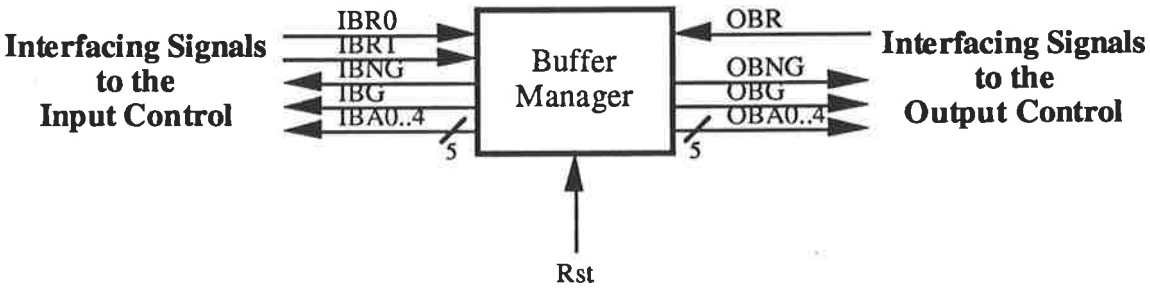


Figure 6.18: Interfacing the buffer manager

Signal Name	Description
Reset (Rst)	Resets the entire buffer manager module
Input Block Request 0 (IBR0)	Requests write permission for incoming high priority cells
Input Block Request 1 (IBR1)	Requests write permission for incoming low priority cells
Input Block Granted (IBG)	Grants incoming write block requests
Input Block Not Granted (IBNG)	Denies incoming write block requests
Input Block Address (IBA0..4)	Top 5-bit write address
Output Block Request (OBR)	Requests read permission
Output Block Granted (OBG)	Grants incoming read block requests
Output Block Not Granted (OBNG)	Denies incoming read block requests
Output Block Address (OBA0..4)	Top 5-bit read address

Table 6.5: Description of interfacing signals in the buffer manager

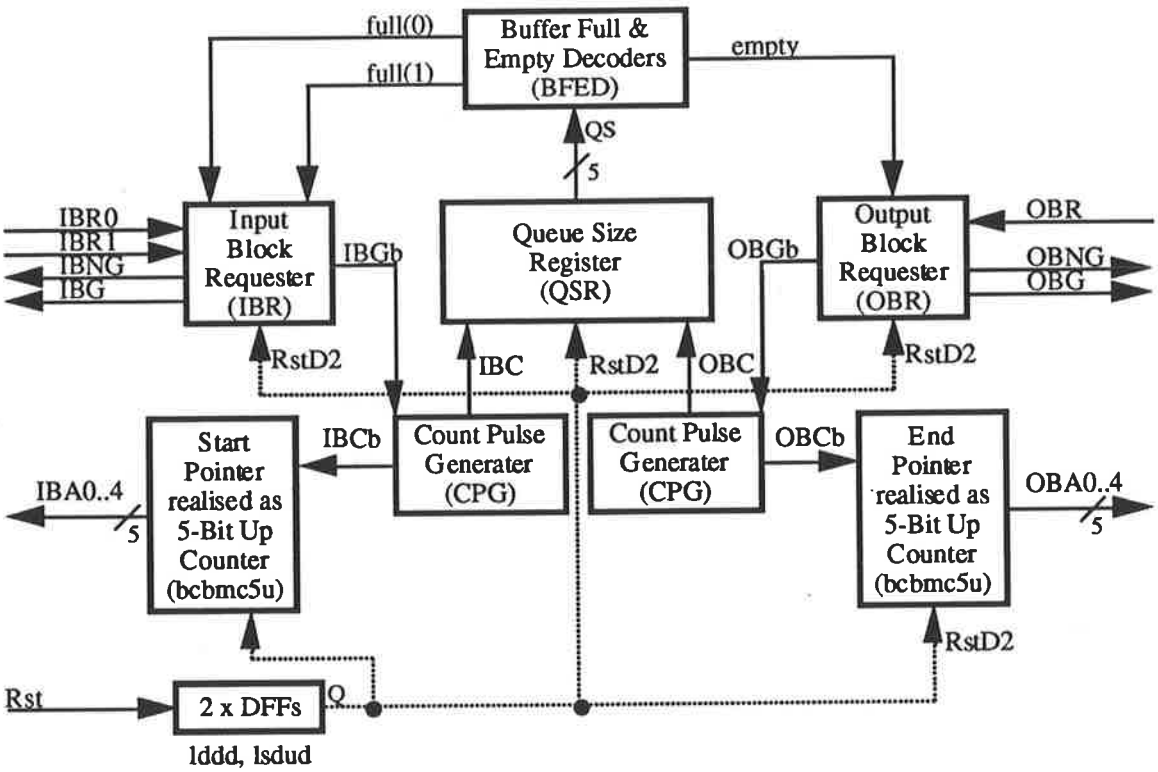


Figure 6.19: Block diagram representation of the buffer manager

Module Name	Functionality
Buffer Full & Empty Decoder (BFED)	Indicates whether the DRAM is full, 3/4 full or empty
Queue Size Register (QSR)	Keeps track of the status of the DRAM
Input Block Requester (IBR)	Processes input requests and sends out acknowledgments
Output Block Requester (OBR)	Processes output requests and sends out acknowledgments
Count Pulse Generator (CPG)	Generates count up/down pulses for the QSR
5-Bit Up Counter[10] (bcbmc5u)	Computes and sends out the top 5-bit input/output block address

Table 6.6: Module requirement of the buffer manager

The realisation of the six individual modules is discussed in the following sections.

10 Note that both the start and end pointers (SP, EP) are realised as 5-bit up counters.

6.2.1 The Queue Size Register

The core of the queue size register (**QSR**) is a synchronous 5-bit up/down counter where the schematic and the resulting state transition table are shown in Figure 6.20 and Table 6.7 respectively. The reason for a 5-bit instead of a 6-bit implementation for the **QSR** can be illustrated as follows: Assume there are 31 stored cells in the DRAM, a further granted incoming input block request will attempt to overwrite the new cell on the location where the first cell is stored. This is due to the first-in-first-out (FIFO) memory structure where the 5-bit input block address (**IBA0..4**) can overflow and recycle. As a remedy, only a 5-bit up/down counter is required to indicate the status of the DRAM.

Additional logic (module **QSRUD**) is included at the input of the counter to ensure stability when both the input block count (**IBC**) and output block count (**OBC**) pulses are high at the same time. This is done by not sending any increment (**UPB**) nor decrement (**DNB**) signals to the counter as the status of the **QSR** would remain unchanged.

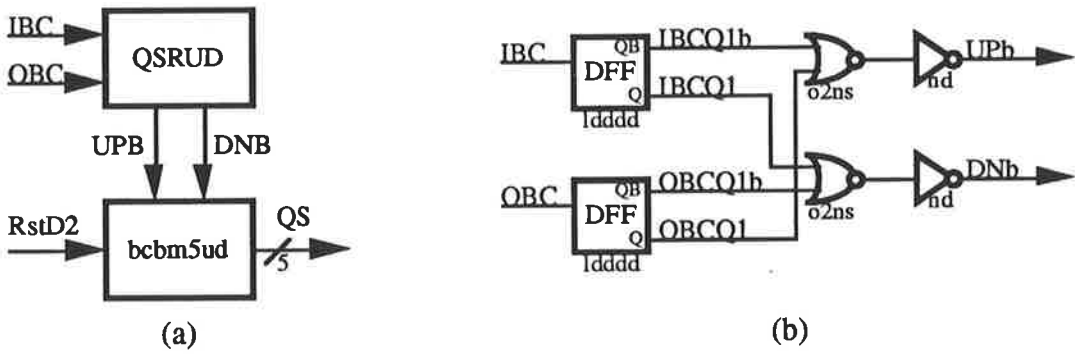


Figure 6.20(a): Schematic of the queue size register and

(b): Schematic of the module QSRUD

(c)

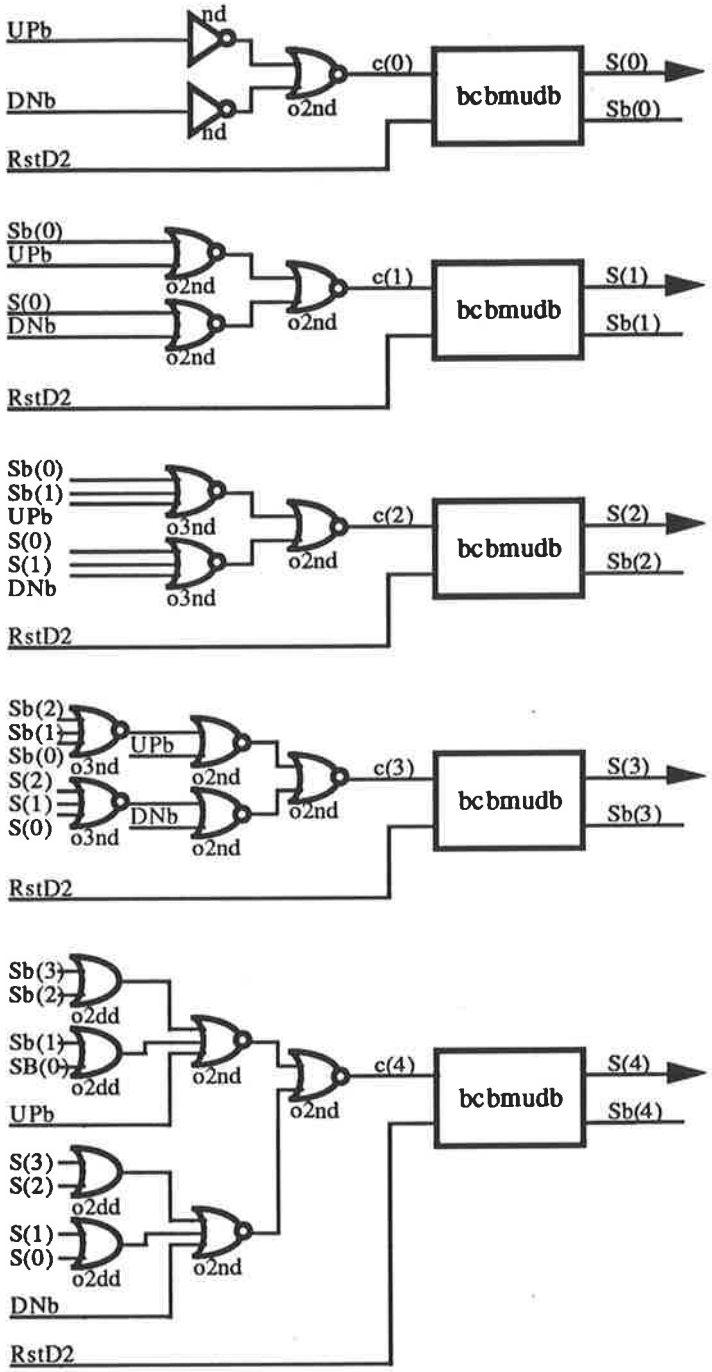


Figure 6.20(c): Schematic of the 5-bit up/down counter (bcbm5ud)

(d)

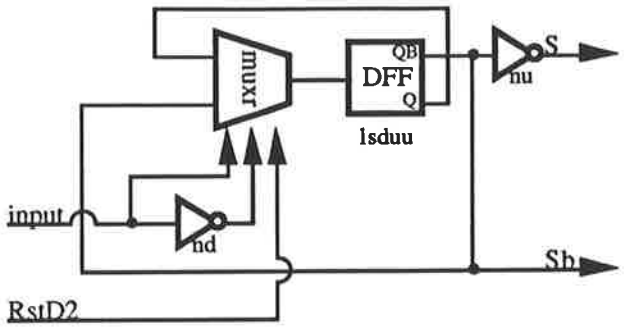


Figure 6.20(d): Schematic of the module bcbm5udb within the 5-bit up/down counter

Present State					Next State (UPb=0)					Next State (DNb=0)					Next State (UPb·DNb=1)				
QS4	QS3	QS2	QS1	QS0	QS4	QS3	QS2	QS1	QS0	QS4	QS3	QS2	QS1	QS0	QS4	QS3	QS2	QS1	QS0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0	0
0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1
0	0	1	1	0	0	0	1	1	1	0	0	1	0	1	0	0	1	1	0
0	0	1	1	1	0	1	0	0	0	0	0	1	1	0	0	0	1	1	1
0	1	0	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0	0
0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0	1	0	0	1
0	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	0	1	0
0	1	0	1	1	0	1	1	0	0	0	1	0	1	0	0	1	0	1	1
0	1	1	0	0	0	1	1	0	1	0	1	0	1	1	0	1	1	0	0
0	1	1	0	1	0	1	1	1	0	0	1	1	0	0	0	1	1	0	1
0	1	1	1	0	0	1	1	1	1	0	1	1	0	1	0	1	1	1	0
0	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	1	1	1	1
1	0	0	0	0	1	0	0	0	1	0	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	1	1	1	0	0	0	1	1	0	0	1	0
1	0	0	1	1	1	0	1	0	0	1	0	0	1	0	1	0	0	1	1
1	0	1	0	0	1	0	1	0	1	1	0	0	1	1	1	0	1	0	0
1	0	1	0	1	1	0	1	1	0	1	0	1	0	0	1	0	1	0	1
1	0	1	1	0	1	0	1	1	1	1	0	1	0	1	1	0	1	1	0
1	0	1	1	1	1	1	0	0	0	1	0	1	1	0	1	0	1	1	1
1	1	0	0	0	1	1	0	0	1	1	0	1	1	1	1	1	0	0	0
1	1	0	0	1	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1
1	1	0	1	0	1	1	0	1	1	1	1	0	0	1	1	1	0	1	0
1	1	0	1	1	1	1	1	0	0	1	1	0	1	0	1	1	0	1	1
1	1	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1	0	0
1	1	1	0	1	1	1	1	1	0	1	1	1	0	0	1	1	1	0	1
1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0
1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1

Table 6.7: State transition table of 5-bit up/down counter

6.2.2 The Buffer Full and Empty Decoders

The buffer full and empty decoders (**BFED**) can simply be realised as a set of decoders as shown in Figure 6.21 where the signals **QS4**, **QS3**, **QS2**, **QS1**, and **QS0** denotes the state vectors from the queue size register (**QSR**) while **QS4b**, **QS3b**, **QS2b**, **QS1b**, and **QS0b** denotes their complement. Table 6.8 shows the truth table of the module.

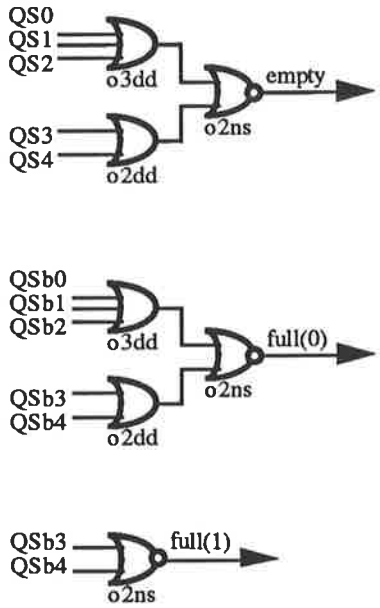


Figure 6.21: Schematic of BFED

Number of cells in DRAM	Equivalent QS Output					empty <Empty>	full(0) <Full>	full(1) <3/4 Full>
	QS4	QS3	QS2	QS1	QS0			
0	0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	0
2	0	0	0	1	0	0	0	0
3	0	0	0	1	1	0	0	0
4	0	0	1	0	0	0	0	0
5	0	0	1	0	1	0	0	0
6	0	0	1	1	0	0	0	0
7	0	0	1	1	1	0	0	0
8	0	1	0	0	0	0	0	0
9	0	1	0	0	1	0	0	0
10	0	1	0	1	0	0	0	0
11	0	1	0	1	1	0	0	0
12	0	1	1	0	0	0	0	0
13	0	1	1	0	1	0	0	0
14	0	1	1	1	0	0	0	0
15	0	1	1	1	1	0	0	0
16	1	0	0	0	0	0	0	0
17	1	0	0	0	1	0	0	0
18	1	0	0	1	0	0	0	0
19	1	0	0	1	1	0	0	0
20	1	0	1	0	0	0	0	0
21	1	0	1	0	1	0	0	0
22	1	0	1	1	0	0	0	0
23	1	0	1	1	1	0	0	0
24	1	1	0	0	0	0	0	1
25	1	1	0	0	1	0	0	1
26	1	1	0	1	0	0	0	1
27	1	1	0	1	1	0	0	1
28	1	1	1	0	0	0	0	1
29	1	1	1	0	1	0	0	1
30	1	1	1	1	0	0	0	1
31	1	1	1	1	1	0	1	1

Table 6.8: Truth table of the buffer full and empty decoders

6.2.3 The Input Block Requester

The input block requester (**IBR**) module receives information about the status of the DRAM (whether it is full, more than 3/4 full or less than 3/4 full) to process the input block requests (**IBR0** and **IBR1**). Figure 6.22 shows the schematic of the **IBR** module where the module **bcbmbr** within is realised as shown in Figure 6.23. One **bcbmbr** module is assigned for each cell priority. A timing diagram of the entire **IBR** module is given in Figure 6.24.

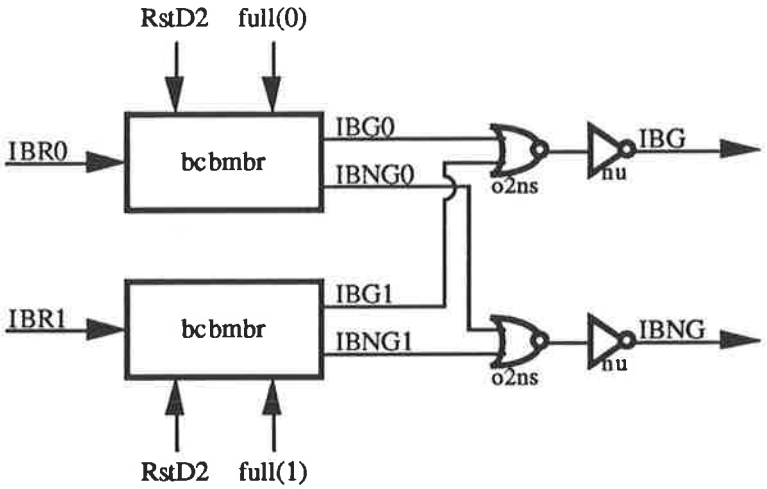


Figure 6.22: Schematic of the input block requester

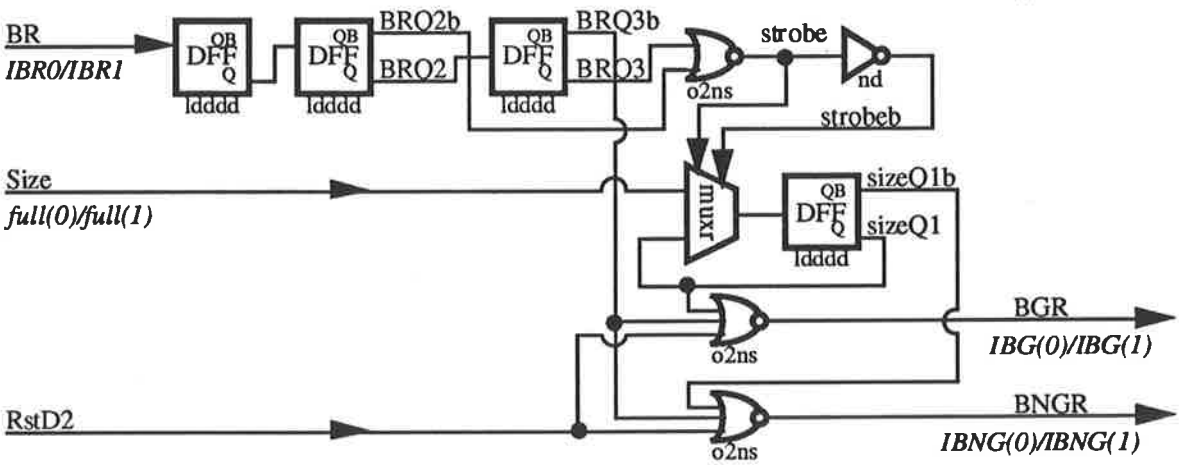


Figure 6.23: Schematic of the module bcbmbr

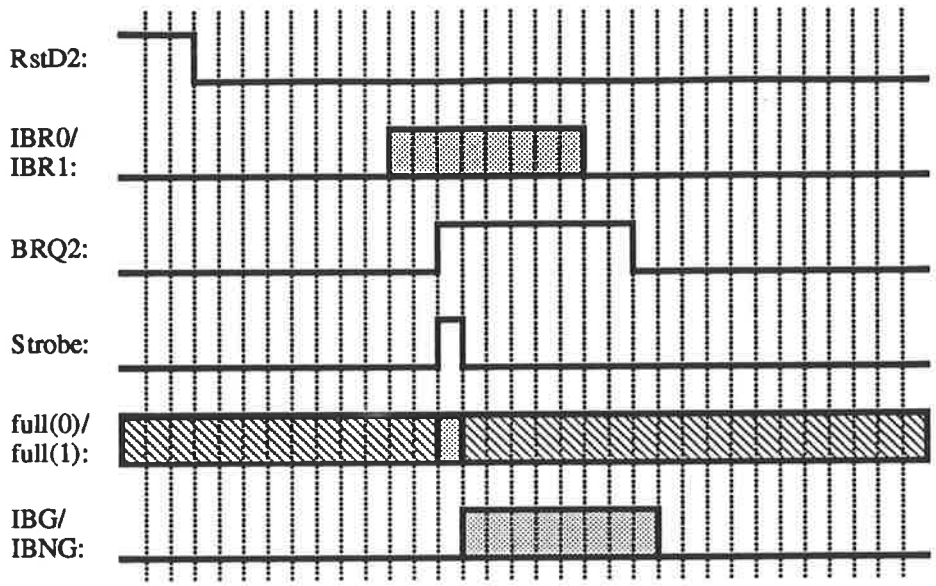


Figure 6.24: Timing diagram of the module bcbmbr

6.2.4 The Output Block Requester

The functionality of the output block requester (**OBR**) module is similar to that of the input block requester (**IBR**) module except only one **bcbmbr** module is required since all output block requests have the same priority. The **OBR** module can be realised as shown in Figure 6.25 and the corresponding timing diagram is shown in Figure 6.26.

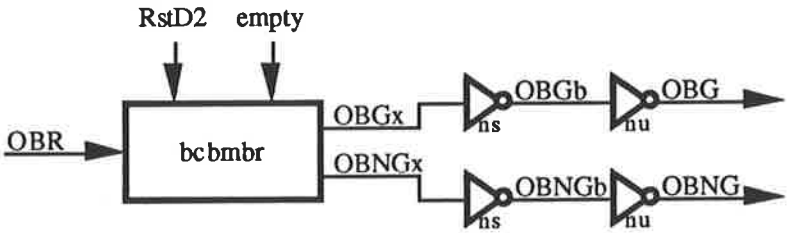


Figure 6.25: Schematic of the output block requester module

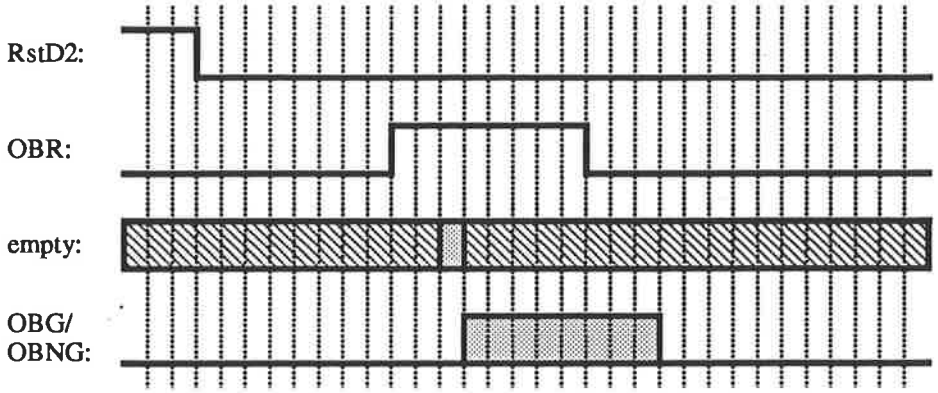


Figure 6.26: Timing diagram of the output block requester module

6.2.5 The Count Pulse Generater

The count pulse generater (**CPG**) produces a negative pulse of one clock period long on every positive edge of the input signal (**In**). It can be realised as shown in Figure 6.27 and the corresponding timing diagram is shown in Figure 6.28.

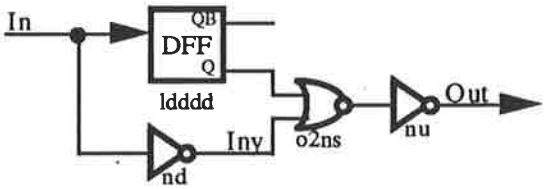


Figure 6.27: Schematic of the count pulse generater module

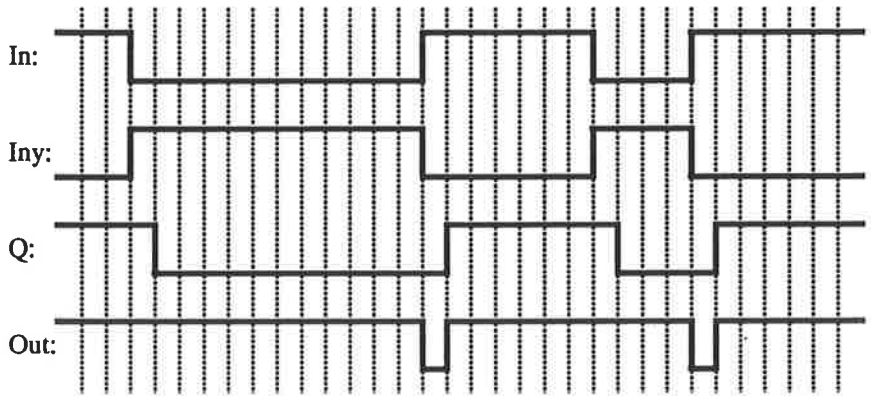


Figure 6.28: Timing diagram of the count pulse generater module

6.2.6 The 5-Bit Up Counters

The 5-bit up counters (**bcbmc5u**) in both the Start Pointer (**SP**) and the End Pointer (**EP**) are realised as synchronous counters. The schematic is shown in Figure 6.29 and the corresponding state transition diagram is shown in Table 6.9.

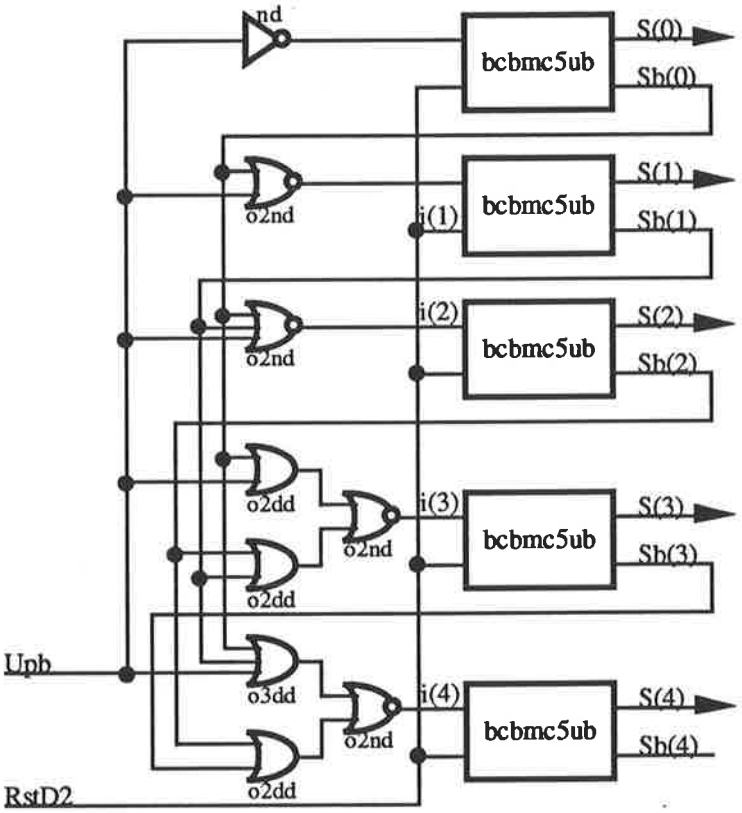


Figure 6.29(a): Schematic of the 5-bit up counter (bcbmc5u)

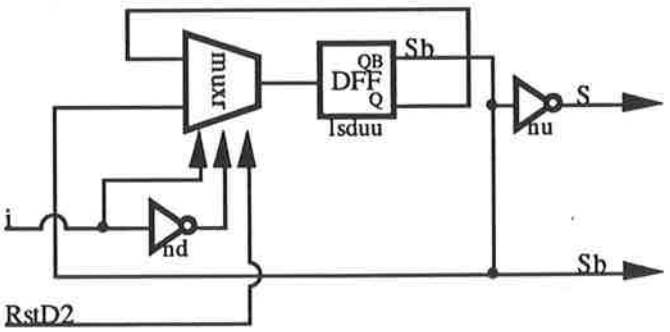


Figure 6.29(b): Schematic of the module bcbmc5ub

Present State					Next State (Up=0)					Next State (Up=1)				
S4	S3	S2	S1	S0	S4	S3	S2	S1	S0	S4	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	1	0	0	0	1	1	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	1
0	0	1	0	1	0	0	1	0	1	0	0	1	1	0
0	0	1	1	0	0	0	1	1	0	0	0	1	1	1
0	0	1	1	1	0	0	1	1	1	0	1	0	0	0
0	1	0	0	0	0	1	0	0	0	0	1	0	0	1
0	1	0	0	1	0	1	0	0	1	0	1	0	1	0
0	1	0	1	0	0	1	0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	1	0	0	0	1	1	0	1
0	1	1	0	1	0	1	1	0	1	0	1	1	1	0
0	1	1	1	0	0	1	1	1	0	0	1	1	1	1
0	1	1	1	1	0	1	1	1	1	1	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
1	0	0	0	1	1	0	0	0	1	1	0	0	1	0
1	0	0	1	0	1	0	0	1	0	1	0	0	1	1
1	0	0	1	1	1	0	0	1	1	1	0	1	0	0
1	0	1	0	0	1	0	1	0	0	1	0	1	0	1
1	0	1	0	1	1	0	1	0	1	1	0	1	1	0
1	0	1	1	0	1	0	1	1	0	1	0	1	1	1
1	0	1	1	1	1	0	1	1	1	1	1	0	0	0
1	1	0	0	0	1	1	0	0	0	1	1	0	0	1
1	1	0	0	1	1	1	0	0	1	1	1	0	1	0
1	1	0	1	0	1	1	0	1	0	1	1	0	1	1
1	1	0	1	1	1	1	0	1	1	1	1	1	0	0
1	1	1	0	0	1	1	1	0	0	1	1	1	0	1
1	1	1	0	1	1	1	1	0	1	1	1	1	1	0
1	1	1	1	0	1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0

Table 6.9: State transition table for the 5-bit up counter

Figure 6.30 shows the layout of the entire buffer manager circuit in **Magic** to verify the effectiveness of the layout style employed. The simulation result in **IRSIM** is shown in Figure 6.31 where the DRAM is initially empty and is gradually filled up by incoming cells while the output block request signal (**OBR**) requests a block once every 53 clock periods. It can be verified that when the DRAM is at or above 3/4 full (from the **BHF** signal), only high priority block requests (**IBR0**) are granted. When the DRAM is completely full, all block requests (**IBR0**, **IBR1**) are rejected. Furthermore, no output block request (**OBR**) will be granted unless the DRAM is filled with at least one cell. Due to the long simulation time for **HSpice**, only a 3-cell long simulation was carried out. Initially, an output block request (**OBR**) arrives and since the DRAM is empty, it is rejected. A high priority input block request (**IBR0**) then arrives with an output block request (**OBR**). Since the DRAM is still empty, **OBR** is rejected while **IBR0** is granted. The status of the DRAM then becomes filled and the buffer empty signal (**BE**) deserts. A low priority input block request (**IBR1**) then arrives and gets granted. Finally, a high priority input block request (**IBR0**) arrives with an output block request (**OBR**). As the DRAM is neither empty nor completely filled, both requests are granted. By the same argument as Section 6.1, the simulation is performed at a 50% over-design factor (ie. 450MHz) to ensure the targeted operational speed can be achieved. The simulation result is shown in Figure 6.32 with all output nodes connected to a capacitive load of 500fF to model the effect of the interconnect wire.

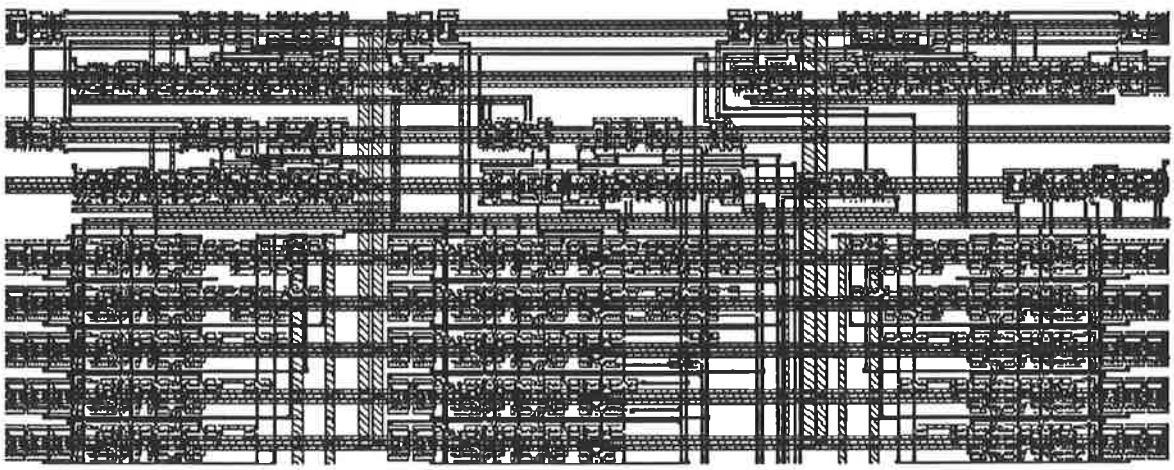


Figure 6.30: Layout of the buffer manager in Magic

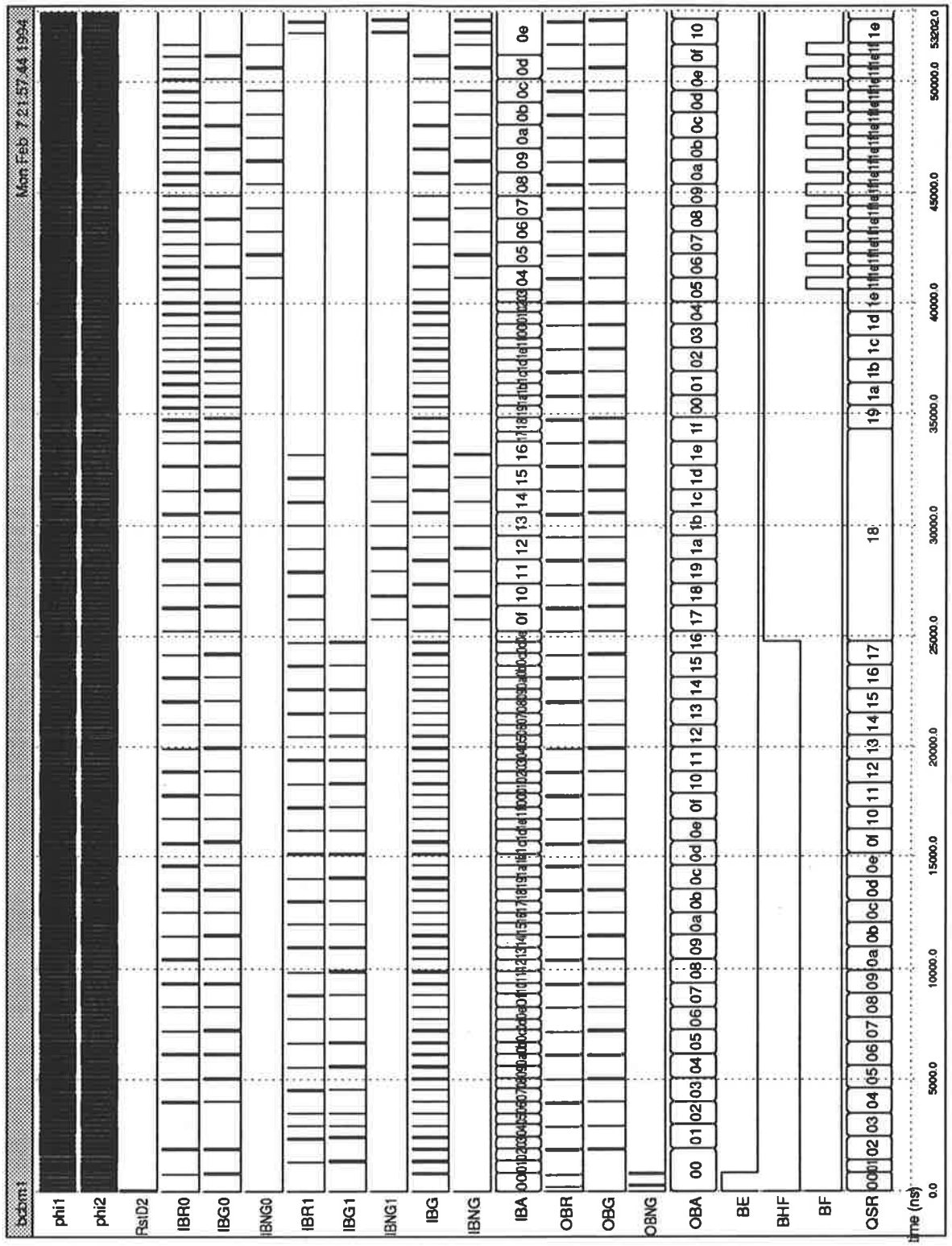


Figure 6.31: Simulation of the buffer manager in IRSIM

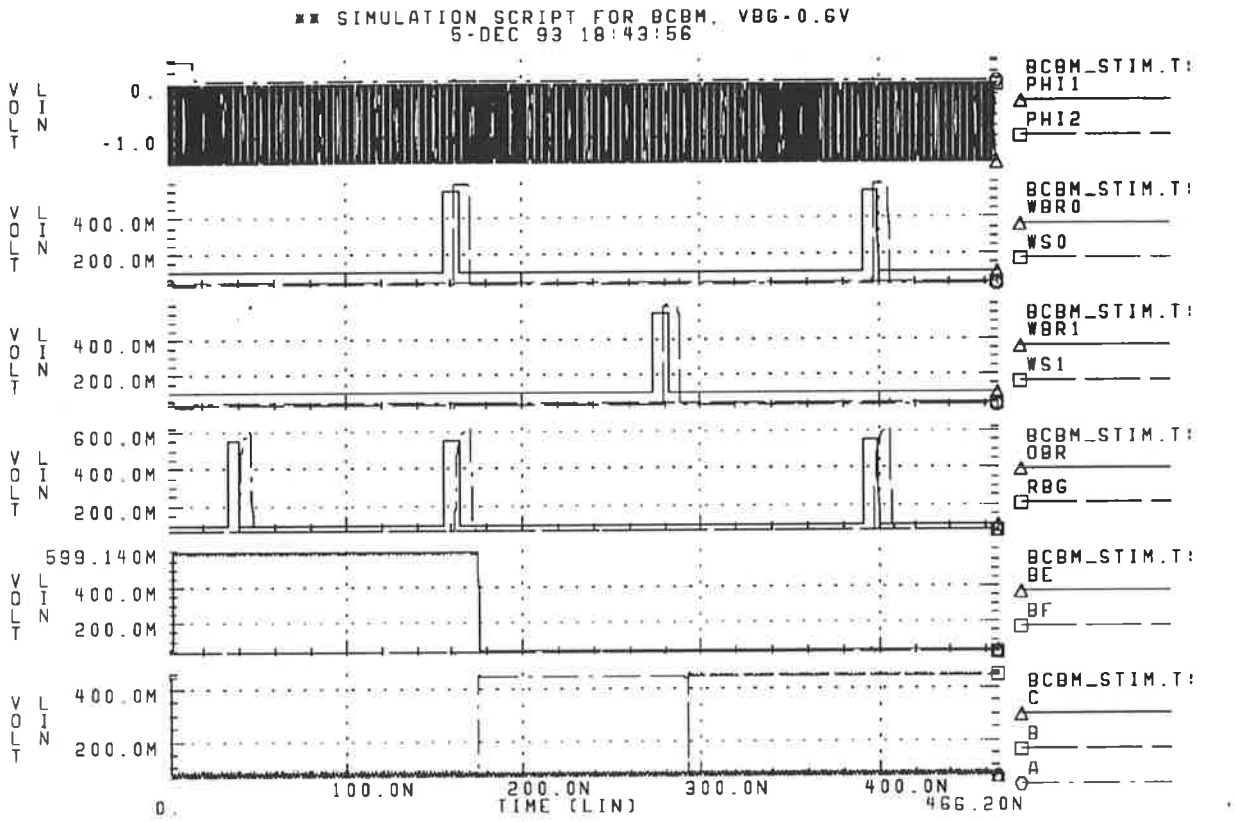


Figure 6.32(a): HSpice simulation of the buffer manager at 450MHz

Note that all output nodes are loaded with a 500fF capacitor

Key: Graph 1 shows the reset signal (RST) (not shown in legend) and the two phase clock (PHI1, PHI2)

Graph 2 shows the high priority input block request (IBR0), the high priority input block granted (IBG0) and the high priority input block not granted (IBNG0) signals which are denoted by WBR0, WS0 and WNS0 (not shown) in the legend

Graph 3 shows the low priority input block request (IBR1), the low priority input block granted (IBG1) and the low priority input block not granted (IBNG1) signals which are denoted by WBR1, WS1 and WNS1 (not shown) in the legend

Graph 4 shows the output block request (OBR), the output block granted (OBG) (denoted by RBG in the legend) and the output block not granted (OBNG) (not shown in the legend) signals

Graph 5 shows the buffer empty (BE) and the buffer full (BF) signals

Graph 6 shows the lowest 3-bit state vector in the queue size register (C, B and A)

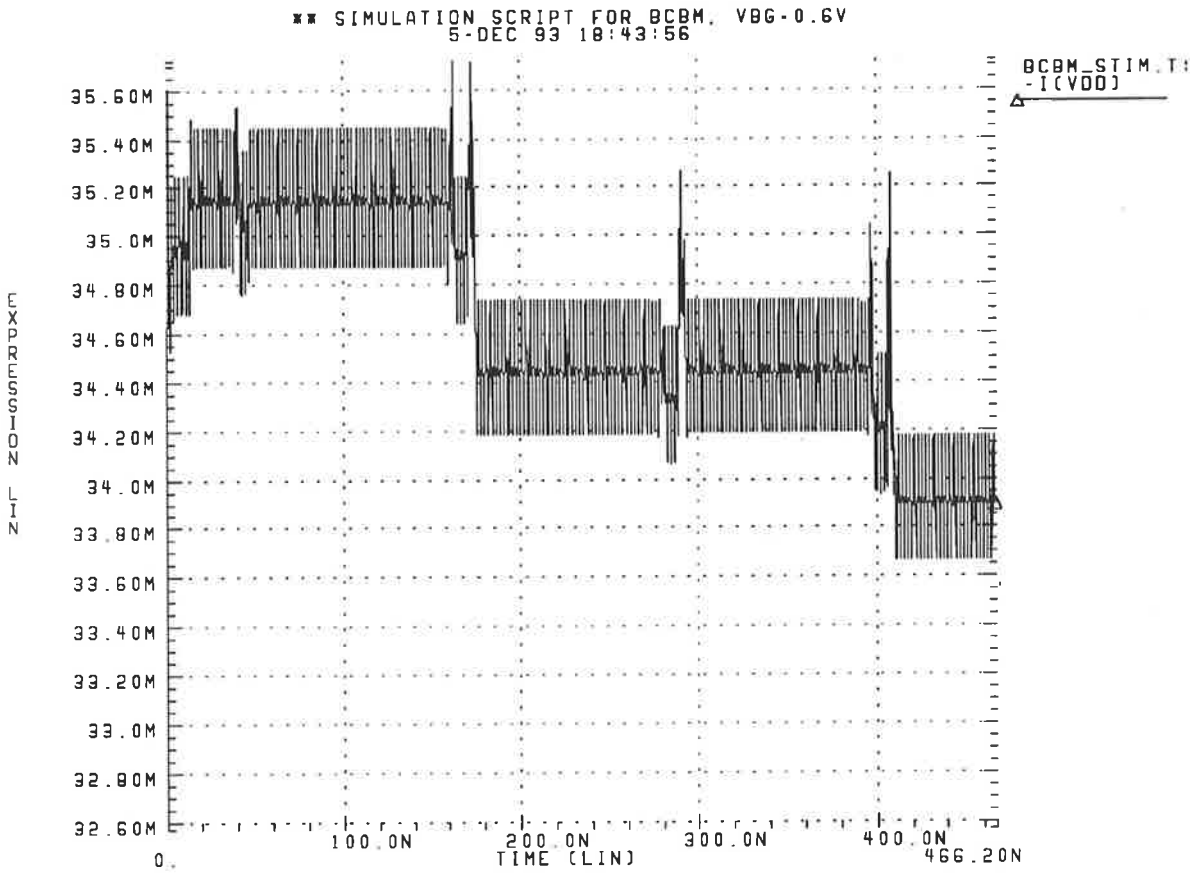


Figure 6.32(b): Current drawn by the buffer manager at 450MHz for the simulation in figure 6.32(a)

It can be verified from Figure 6.32(b) that the logic families does in fact draw a fairly constant current from VDD. The mean deviation of current drawn by the buffer manager is calculated to be 5.2%. This again verifies the fact that the choice of logic families does have a significant impact on the amount of noise injected to the supply lines which in turn affect the performance of the logic themselves.

6.3 The Output Control

The functionality of output control closely resembles that of the input control except the former sends out an output block request (**OBR**) to the buffer manager once every 53 clock cycles whereas the latter sends out input block requests (**IBR0** or **IBR1**) only when cells arrive [CHU4]. Depending upon the decision of the buffer manager, the output block request will either be granted (which will be accompanied by a 5-bit block address) or not granted. If the request is granted, the output control will generate the full 7-bit read address (**RA0..6**) and a read enable (**RE**) signal to the **DRAM** as well as an output convert (**OC**) signal to the output parallel-to-serial (P/S) converters at the end of each subcells. Otherwise, only **OC** signals are generated. A flow chart summarising the functionality of the output control is shown in Figure 6.33. Note that the current granted cell is read on the next 53 clock cycles. This is due to the fact that the queue size register (**QSR**) in the buffer manager indicates a cell is already in the DRAM before it is completely stored and the output control reads cells out from the DRAM at half the speed they were written in by the input control. Consequently, granted output block requests should only start reading cells out from the DRAM in the next 53 clock cycles to ensure they are complete.

An interface of the output control and the corresponding signal descriptions are shown in Figure 6.34 and Table 6.10 respectively.

The output control can be realised in six main modules as shown in Figure 6.35. A functional description of the modules is given in Table 6.11.

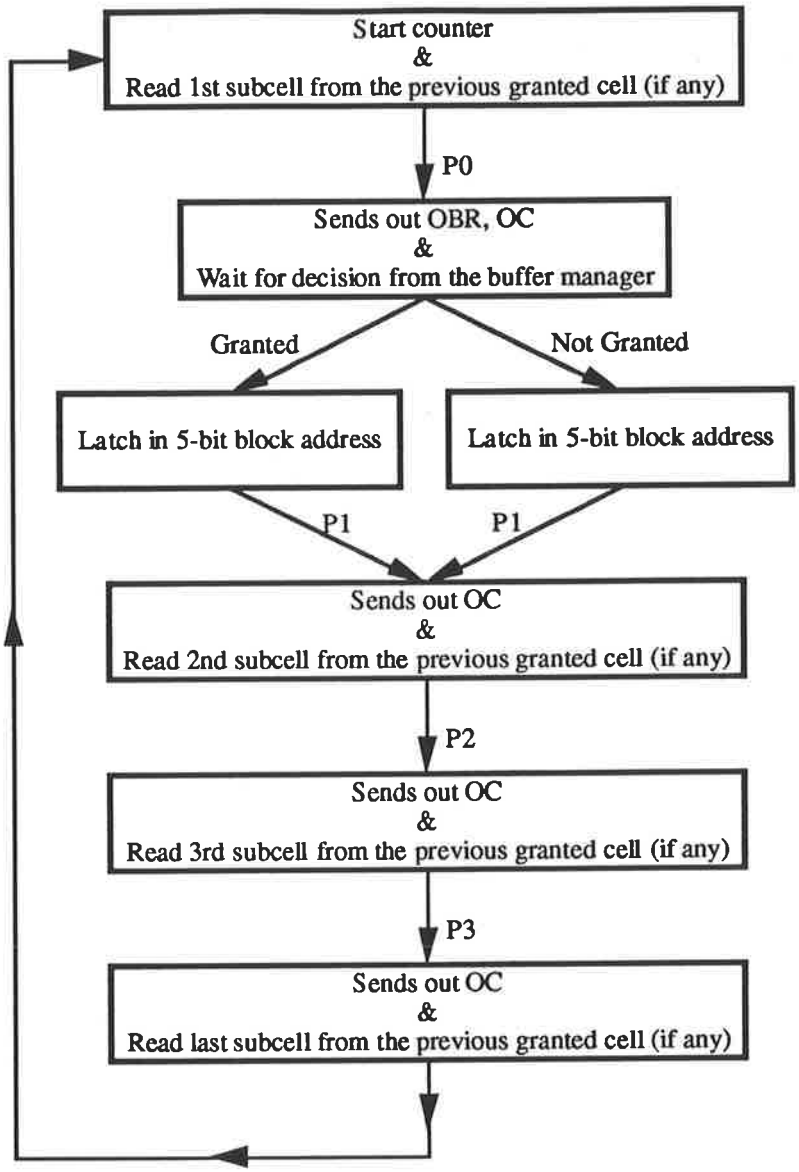


Figure 6.33: Flow chart summarising the functionality of the output control

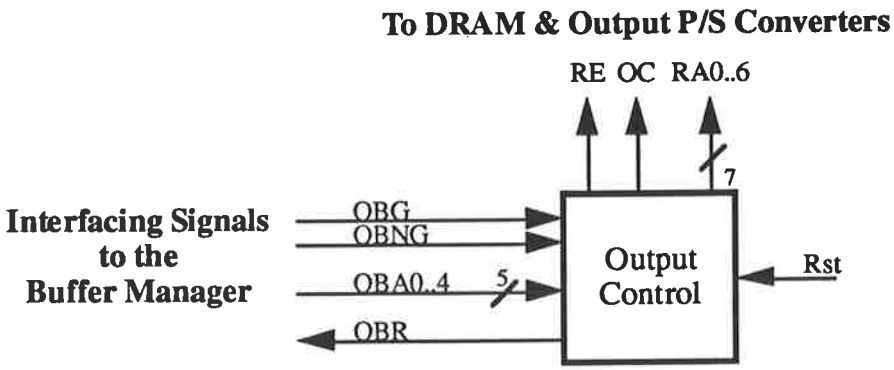


Figure 6.34: Interfacing the output control

Signal Name	Description
Reset (Rst)	Resets the entire output control module
Output Block Request (OBR)	Requests read permission
Output Block Not Granted (OBNG)	Denies incoming read block request
Output Block Granted (OBG)	Grants incoming read block request
Output Block Address 5-bit (OBA0..4)	Top 5-bit read address
Read Address 7-bit (RA0..6)	Full 7-bit read address
Read Enable (RE)	Enable memory read sequence
Output Convert (OC)	Instructs the output P/S converters to start conversion

Table 6.10: Description of interfacing signals in the output control

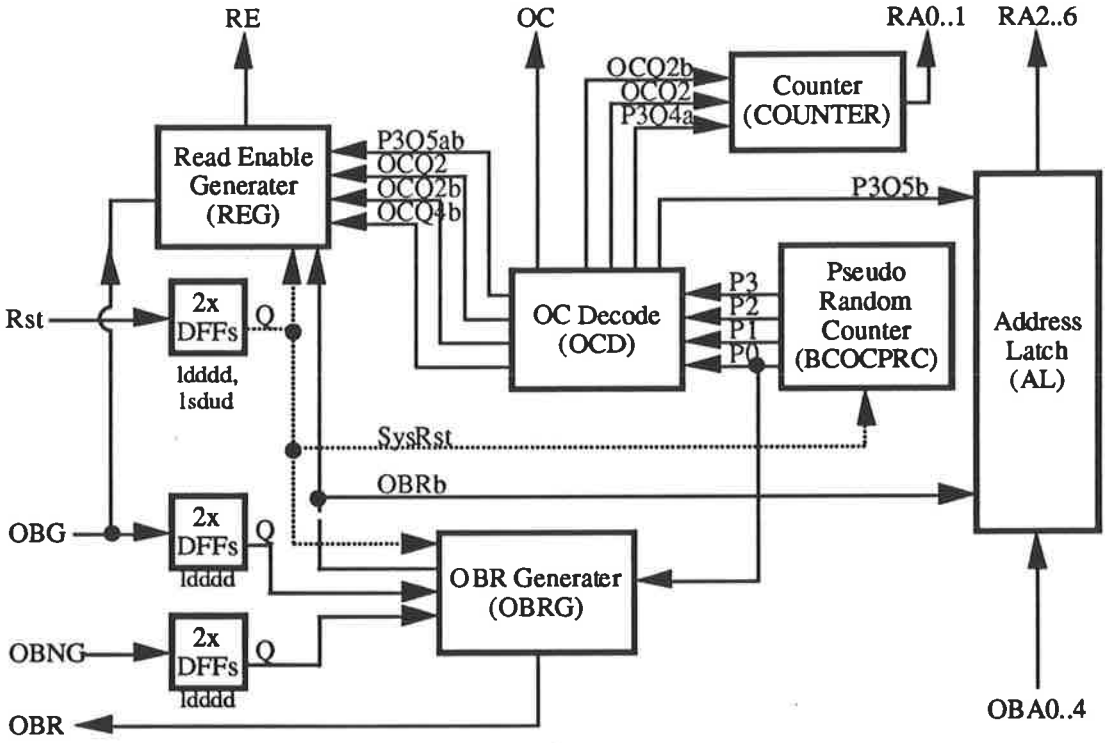


Figure 6.35: Block diagram representation of the output control

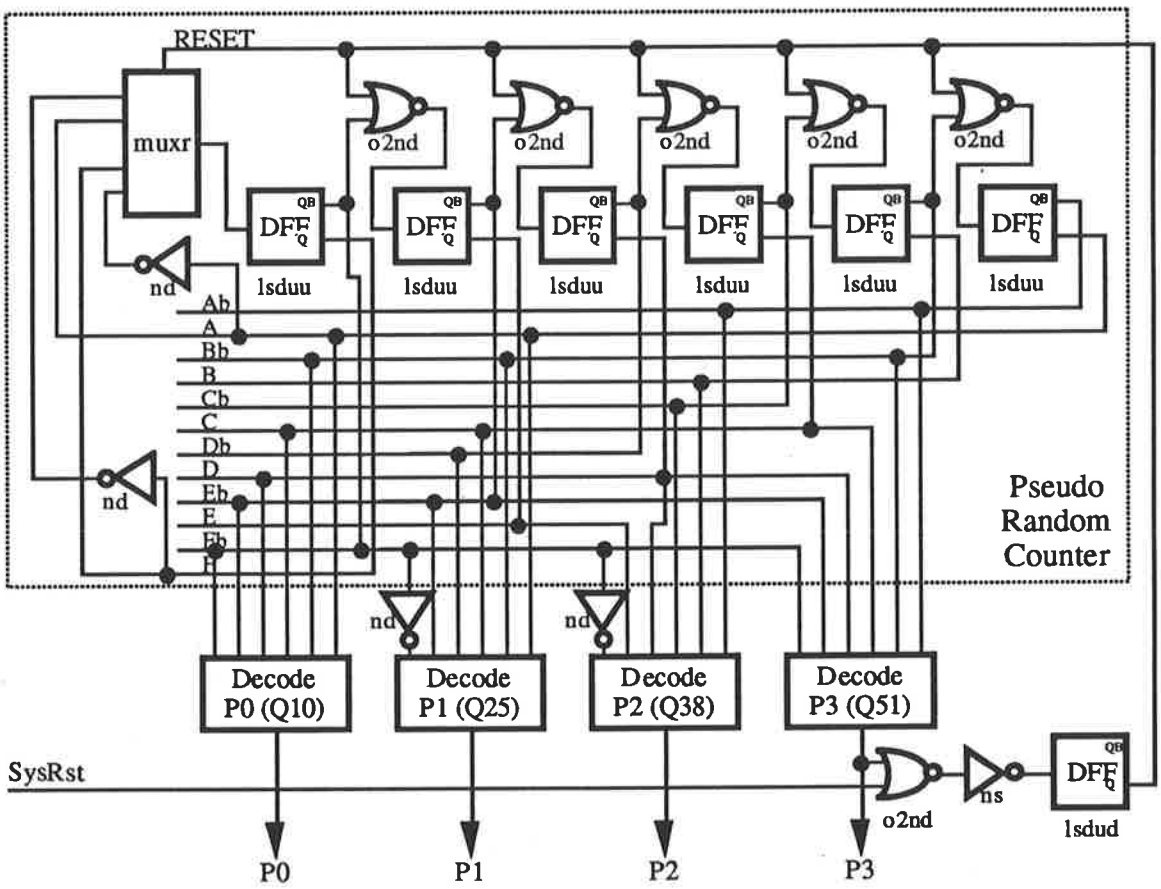
Signal Name	Description
Pseudo Random Counter (BCOCPRC)	Generates four timing pulses to indicate the end of subcells
Address Latches (AL)	Latches out the top 5 bit read address
2-Bit Counter (BCOCC)	Generates the lower 2-bit read address
Read Enable Generater (REG)	Generates read enable signals for the DRAM
Output Block Requester (OBRG)	Generates a read block request signal once every 53 clock periods and sends it to the buffer manager
Output Convert Decoder (OCD)	Generates an output convert signal for the output P/S converters

Table 6.11: Module requirement of the output control

The realisation of the six individual blocks is discussed in the following sections.

6.3.1 The Pseudo Random Counter

The pseudo random counter (**bcocprc**) is similar to the one used in the input control except that it counts at all times whereas the latter counts only when a new cell arrives. As a result, the start extraction module can be removed. The resulting circuit diagram is shown in Figure 6.36 and the corresponding state transition table is shown in Table 6.12.



where the decoders (**bcocprcd**) are realised as:

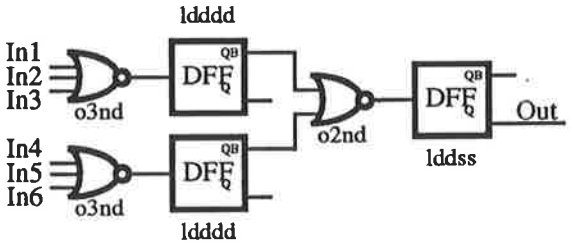


Figure 6.36: Schematic of the pseudo random counter (bcocprc)

Remarks	S5	S4	S3	S2	S1	S0	HEX
Reset State	0	0	0	0	0	0	00
Counting States	1	0	0	0	0	0	20
	0	1	0	0	0	0	10
	1	0	1	0	0	0	28
	0	1	0	1	0	0	14
	1	0	1	0	1	0	2A
	0	1	0	1	0	1	15
	0	0	1	0	1	0	0A
	1	0	0	1	0	1	25
	1	1	0	0	1	0	32
	0	1	1	0	0	1	19
	0	0	1	1	0	0	0C
	1	0	0	1	1	0	26
	0	1	0	0	1	1	13
	0	0	1	0	0	1	09
	0	0	0	1	0	0	04
	1	0	0	0	1	0	22
	0	1	0	0	0	1	11
	0	0	1	0	0	0	08
	1	0	0	1	0	0	24
	0	1	0	0	1	0	12
	1	0	1	0	0	1	29
	1	1	0	1	0	0	34
	0	1	1	0	1	0	1A
	1	0	1	1	0	1	2D
	1	1	0	1	1	0	36
	0	1	1	0	1	1	1B
	0	0	1	1	0	1	0D
	0	0	0	1	1	0	06
	1	0	0	0	1	1	23
	1	1	0	0	0	1	31
	1	1	1	0	0	0	38
	0	1	1	1	0	0	1C
	1	0	1	1	1	0	2E
	0	1	0	1	1	1	17
	0	0	1	0	1	1	0B
	0	0	0	1	0	1	05
	0	0	0	0	1	0	02
	1	0	0	0	0	1	21
	1	1	0	0	0	0	30
	0	1	1	0	0	0	18
	1	0	1	1	0	0	2C
	0	1	0	1	1	0	16
	1	0	1	0	1	1	2B
	1	1	0	1	0	1	35
	1	1	1	0	1	0	3A
	0	1	1	1	0	1	1D
	0	0	1	1	1	0	0E
	1	0	0	1	1	1	27
	1	1	0	0	1	1	33
	1	1	1	0	0	1	39
	1	1	1	1	0	0	3C
	0	1	1	1	1	0	1E
	0	0	0	0	0	0	00

Table 6.12: State transition table for the pseudo random counter (bcocprc)

6.3.2 The Address Latch

The address latch (**AL**) is realised as five lots of two-stage data latches. When an output block request (**OBR**) is sent out, it latches out the current Output Block Address (**OBA**) from the first stage. Five clock periods after the signalling pulse **P3**, this address is latched out from the second stage to the DRAM. Note that this address is latched out irrespective of the status of the output block granted (**OBG**) or the output block not granted (**OBNG**) signals as the validity of the address is only determined by the read enable (**RE**) signal. The schematic and the timing diagram of this module are shown in Figure 6.37 and Figure 6.38 respectively.

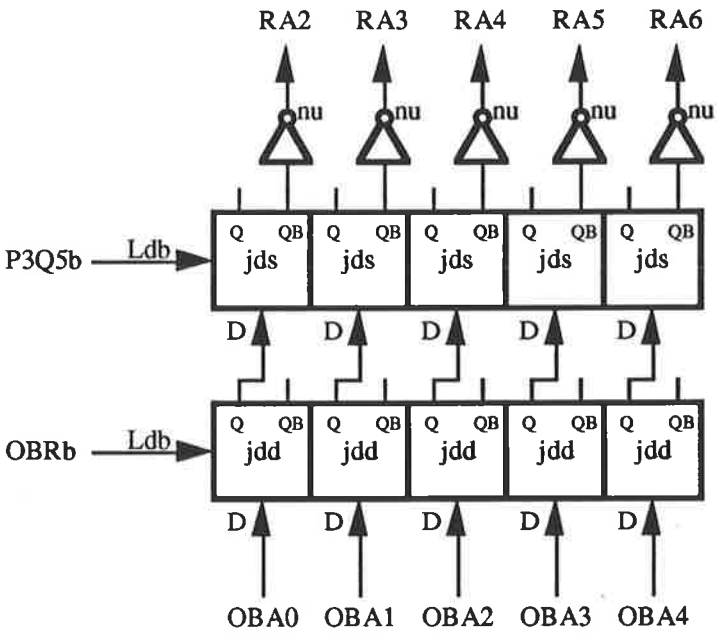


Figure 6.37: Schematic of address latch (AL) module

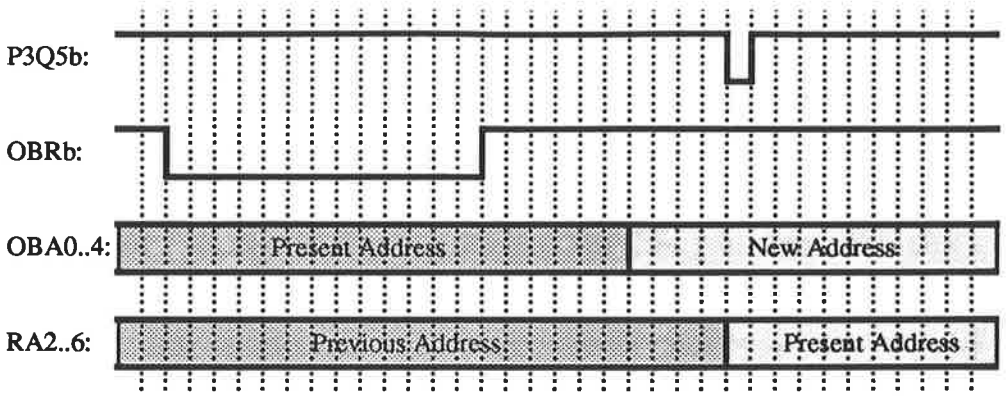


Figure 6.38: Timing diagram of the address latch (AL) module

6.3.3 The 2-Bit Counter

The 2-bit counter (**bcocc**) is identical to the one used in the input control. The circuit diagram and state transition table are shown in Figure 6.39 and Table 6.13 respectively.

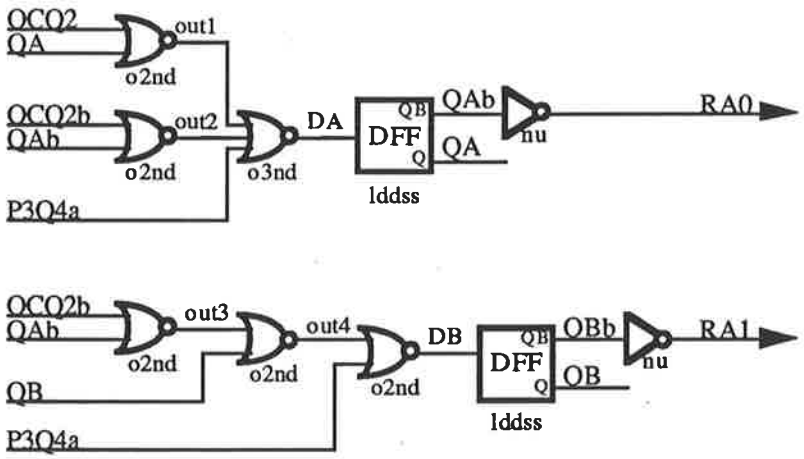


Figure 6.39: Schematic of the 2-bit counter

Present State		Next State (OCQ2=0)		Next State (OCQ2=1)	
$RA1_t$	$RA0_t$	$RA1_{t+1}$	$RA0_{t+1}$	$RA1_{t+1}$	$RA0_{t+1}$
0	0	0	0	0	1
0	1	0	1	1	0
1	0	1	0	1	1
1	1	1	1	1	0

Table 6.13: State transition table for the 2-bit counter

6.3.4 The Read Enable Generater

The read enable generater module (**REG**) generates read enable signals (**RE**) to the DRAM only if an output block request (**OBR**) is granted. It was realised as shown in Figure 6.40. The same 2-stage latch as in the address latch (**AL**) module is used in its construction to latch in the output block granted (**OBG**) signal. A timing diagram is shown in Figure 6.41 where it can be seen that there are three clock cycles between adjacent read enables (**RE**) to ensure the read address is stable before commence reading.

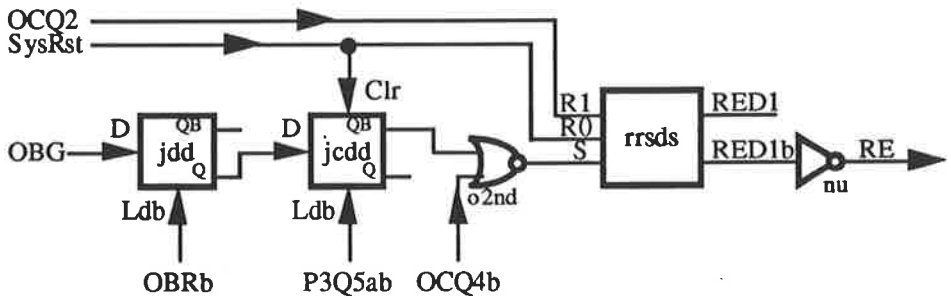


Figure 6.40: Schematic of the read enable generater

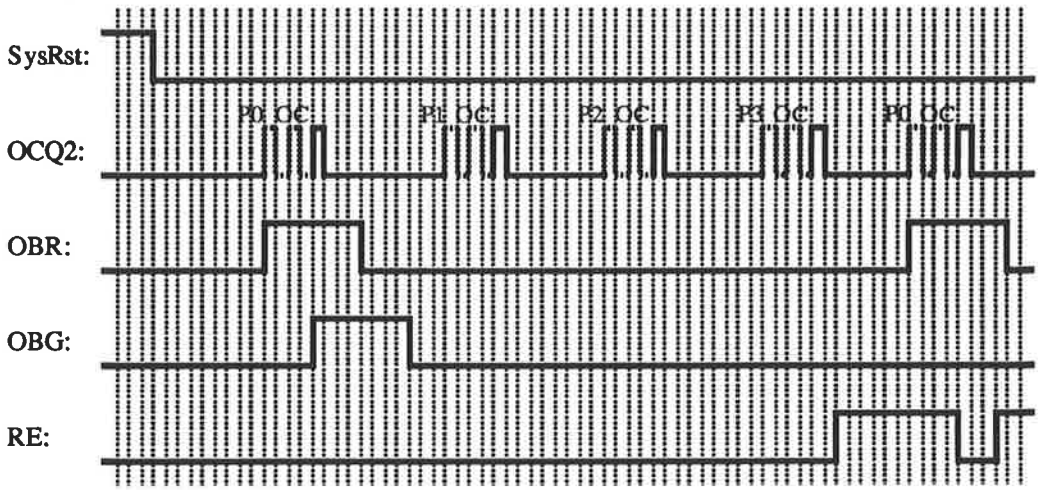


Figure 6.41: Timing diagram of the read enable generater

6.3.5 The Output Block Request Generater

The output block request generater (**OBRG**) sends an output block request (**OBR**) to the buffer manager once every 53 clock periods and awaits its decision. It can be realised as shown in Figure 6.42 with the timing diagram shown in Figure 6.43.

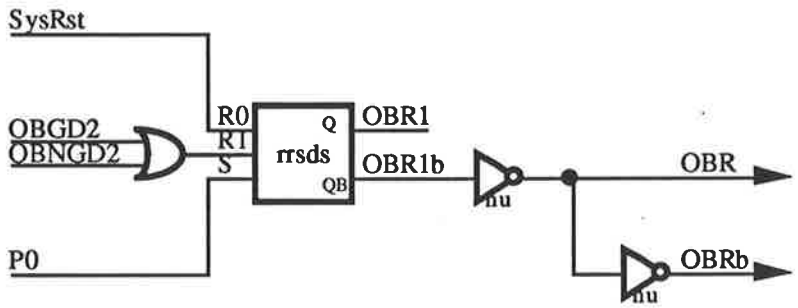


Figure 6.42: Schematic of the output block request generater

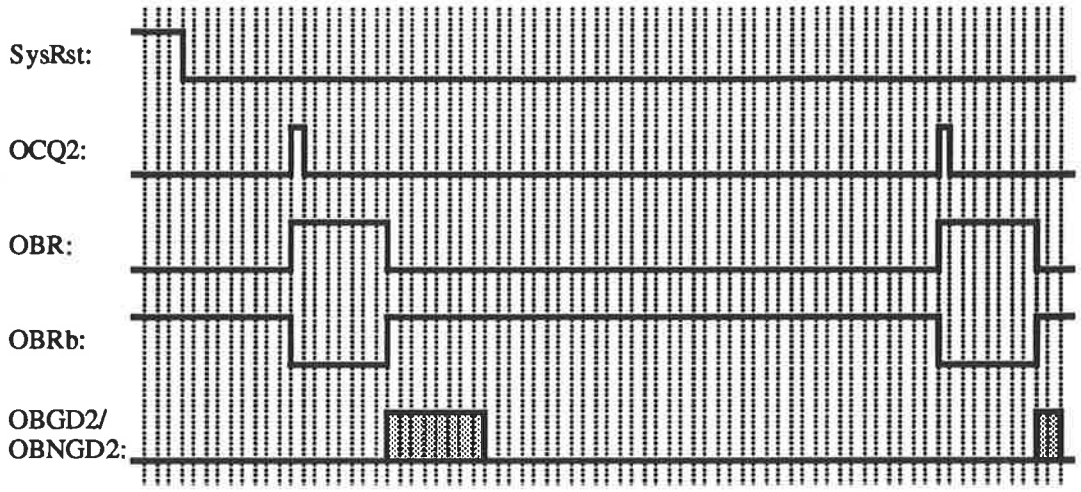


Figure 6.43: Timing diagram of the output block request generater

6.3.6 The Output Convert Decoder

The output convert decoder (**OCD**) generates output convert (**OC**) signals as well as delaying the signalling pulse **P3** for other modules. It can be realised as shown in Figure 6.44 with the timing diagram shown in Figure 6.45.

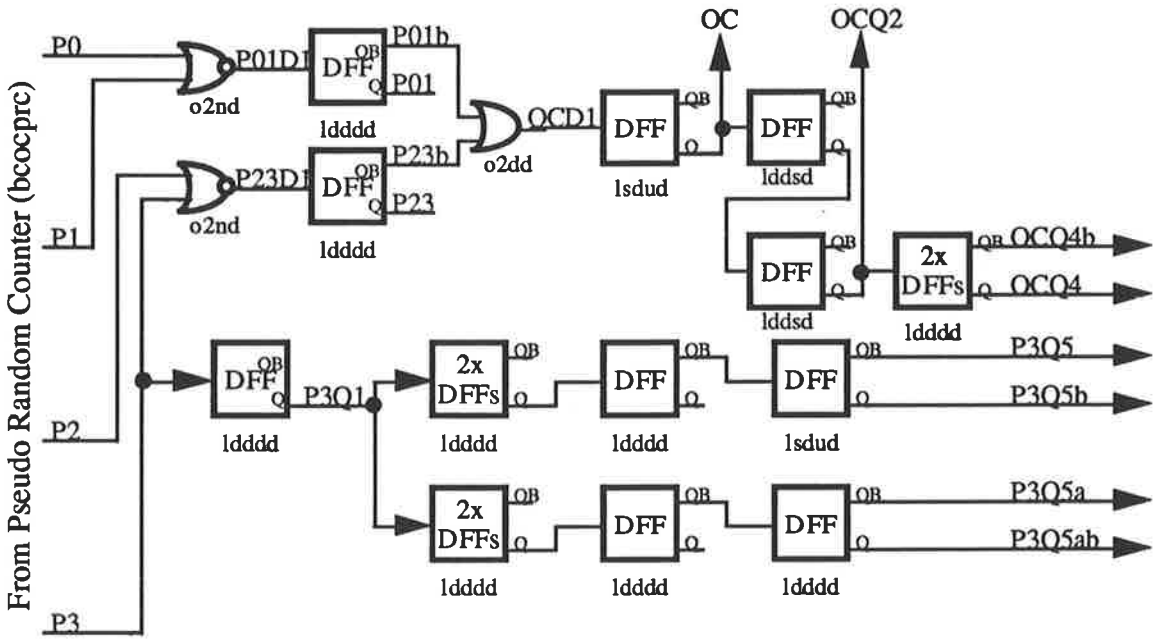


Figure 6.44: Schematic of the output convert decoder

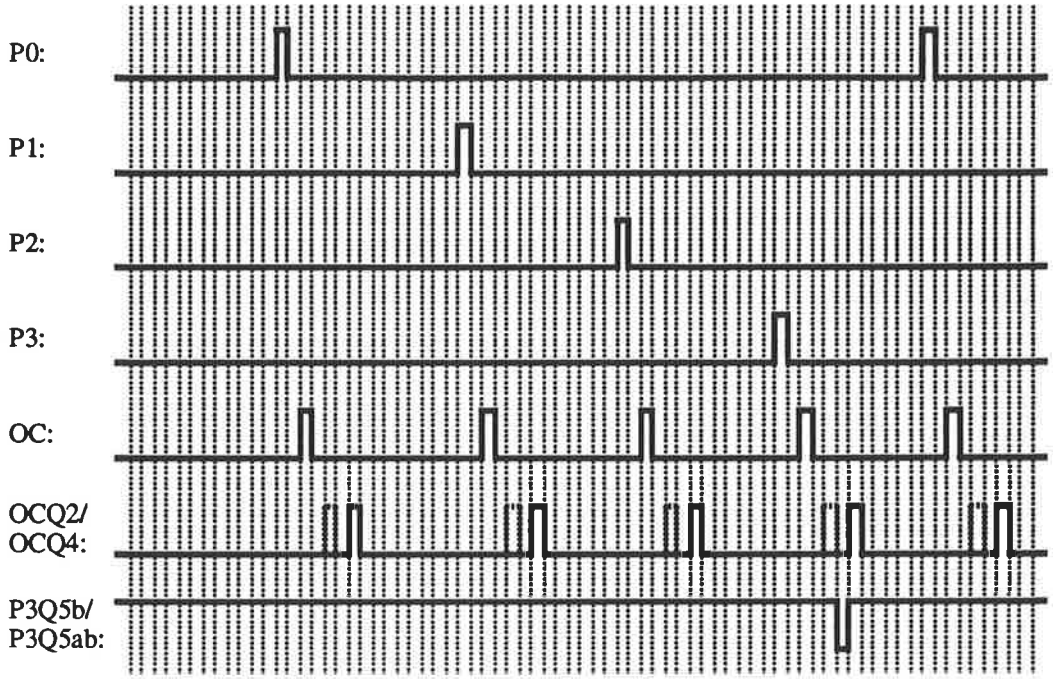


Figure 6.45: Timing diagram of the output convert decoder

Figure 4.46 shows the layout of the entire output control circuit in **Magic** to verify the effectiveness of the layout style employed. The simulation result in **IRSIM** is shown in Figure 6.47 where the DRAM is initially empty and only alternating output block request (**OBR**) gets granted. By the same argument as Section 6.1, the **HSpice** simulation is performed at a 50% over-design factor (ie. 450MHz) to ensure the targeted operational speed of 300MHz can be achieved. The simulation result in **HSpice** is shown in Figure 6.48 where all output nodes are connected to a capacitive load of 1pF (where in reality it translates to a long interconnect line which is connected to other functional blocks) except for the output convert signal (**OC**) which has a capacitive load of 200fF and a 30μm wide EFET connected as external load (where in reality this signal is directly connected to an adjacent differential D flip-flop "fvv3v3v").

It can be verified from the last panel in Figure 6.48 that the logic families do in fact draw a fairly constant current from VDD. The mean deviation of the current drawn by the output control is calculated to be 6.1%. This again justifies the fact that the chosen logic families do have a minimal noise injection to the supply lines.

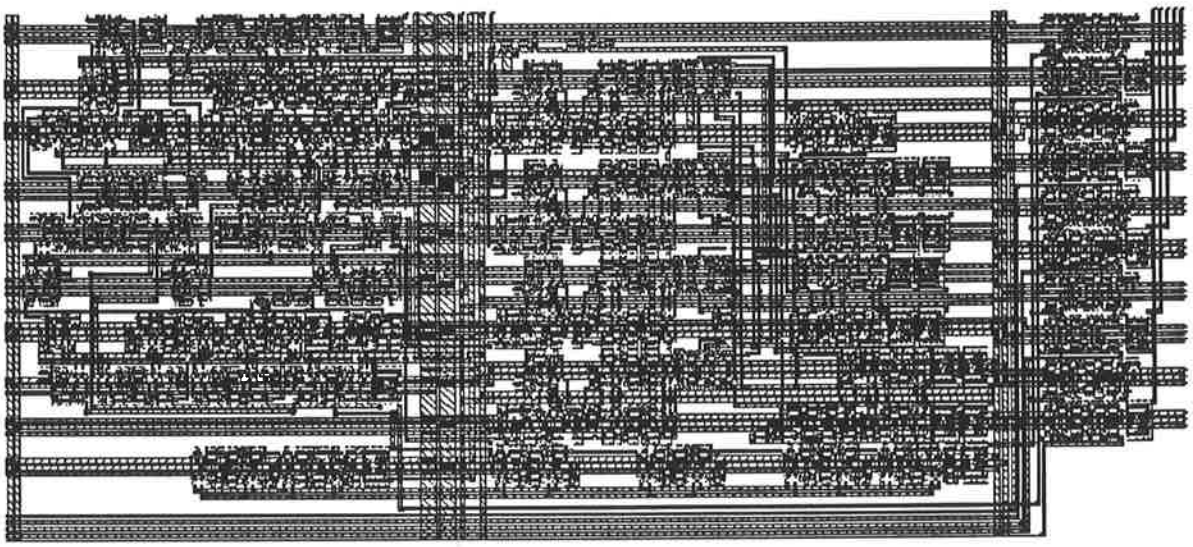


Figure 6.46: Layout of the output control in Magic

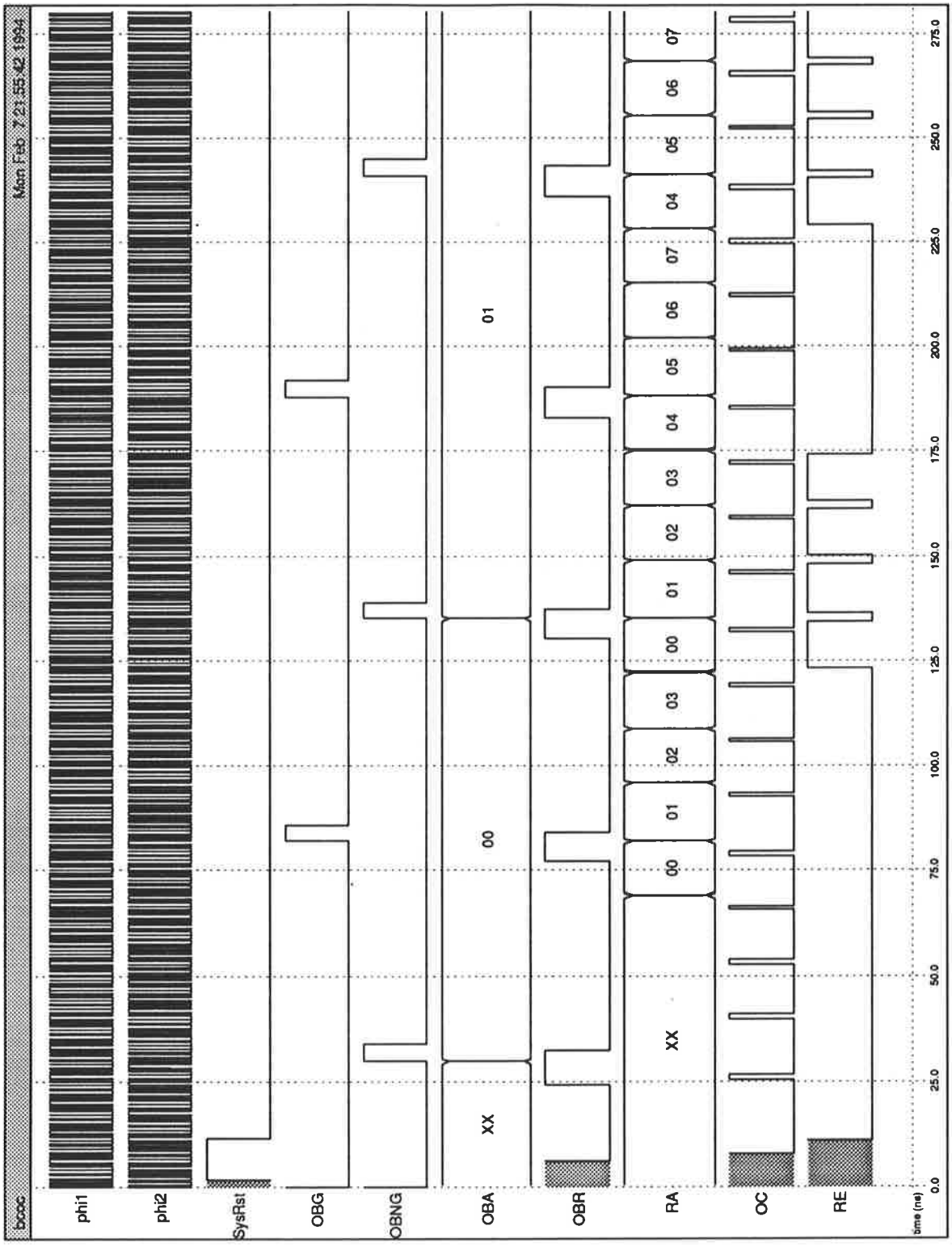


Figure 6.47: Simulation of the output control in IRSIM

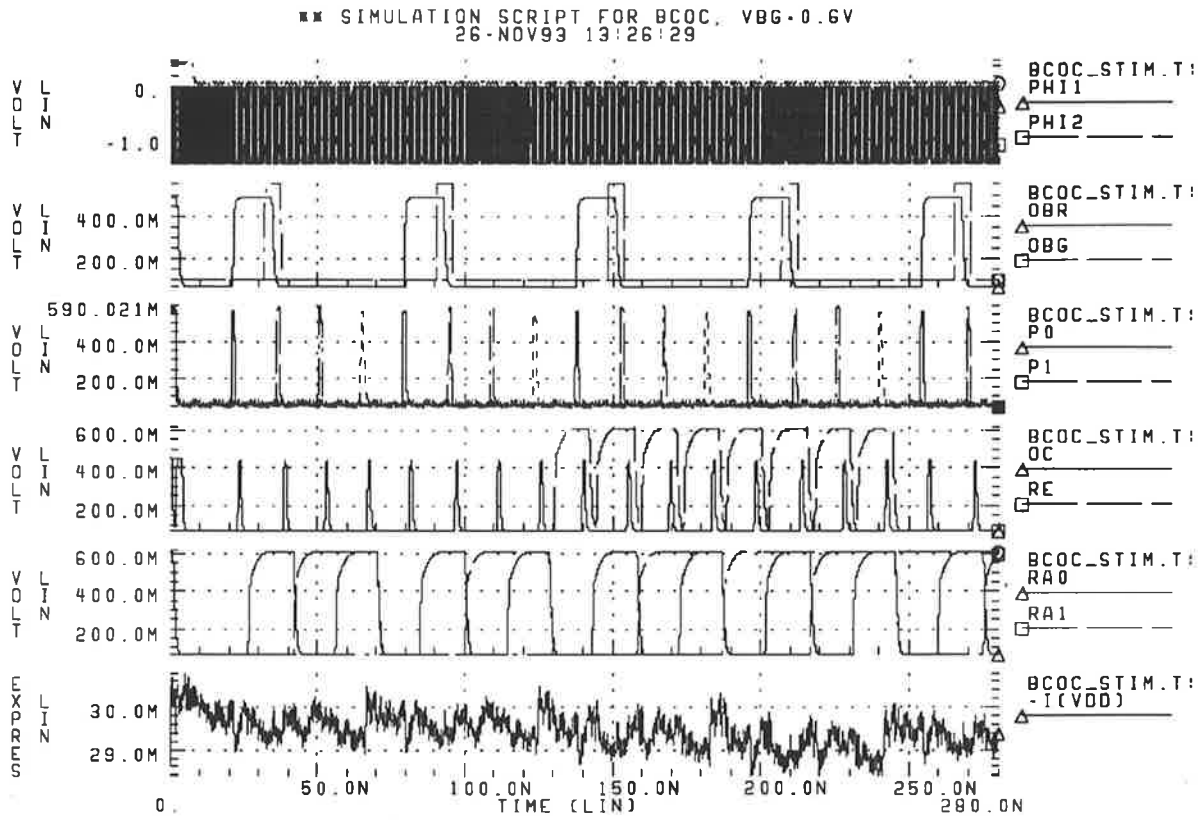


Figure 6.48: HSpice simulation of the output control at 450MHz

Note that all output nodes except OC are loaded with a 1pF capacitor

Key: Graph 1 shows the system reset signal (SYSRST) (not shown in legend) and the two phase clock (PHI1, PHI2)

Graph 2 shows the output block request (OBR), output block granted (OBG) and the output block not granted (OBNG) (not shown in legend) signals

Graph 3 shows the four timing pulses P0, P1, P2 and P3 (P2 and P3 not shown in legend)

Graph 4 shows the output convert (OC) and the read enable (RE) signals

Graph 5 shows the lowest 3-bit read address RA0, RA1 and RA2 (RA2 not shown in legend)

Graph 6 shows the current drawn from the VDD bus

Chapter 7. Conclusion and Future Work

A conclusion of this research is presented in section 7.1 and some possible future work on this project is discussed in section 7.2.

7.1 Conclusion

A survey of GaAs logic families was carried out. Amongst all candidates, Direct Coupled FET Logic (DCFL), Source-follower Direct Coupled FET Logic (SDCFL), Super Buffer FET Logic (SBFL) and Ultra Buffer FET Logic (UBFL) are chosen due to their simplicity, compatibility in signals levels, power consumption and speed requirements. They are optimised at 125°C to operate over military specifications with a 1.5V power supply. After a performance comparison, a "mixed" logic design approach using pure NOR structures is presented in Chapter 2 and is used extensively throughout the design of the control logic within a buffer chip in which DCFL is used extensively for local logic operations where both the fan-ins and output loads are small due to its simplicity and superb power-delay product. SDCFL is used in critical paths and/or higher fan-in applications. SBFL is used only for buffering global signals (such as clocks and resets) due to high power consumption and noise injection to the power busses. UBFL is best suited for buffering parallel busses (such as address and data bus) since the power dissipation is low and the injection of noise into the power busses is the major concern.

The control logic of the buffer chip is realised in three main modules: an input control module, a buffer manager module and an output control module. The details of these modules are discussed in Chapter 6. They are designed, laid out using the layout tool "MAGIC" and simulated using different simulation tools such as IRSIM and HSpice (see Chapter 4). A performance summary of the three modules is shown in Table 7.1.

	Module Name		
	Input Control	Buffer Manager	Output Control
Number of Transistors	1336	1232	1104
Dimensions (length x width)	882μm x 481μm	1069μm x 421μm	912μm x 416μm
Targeted Operational Speed	600MHz	300MHz	300MHz
Simulation Speed	900MHz	450MHz	450MHz
Simulated Power Consumption at Maximum Simulation Speed	less than 56mW	less than 54mW	less than 47mW

Table 7.1: Performance summary of the three modules constituting the control circuit within the buffer chip

In addition, a design methodology for high complexity VLSI circuits was presented. It is based on different levels of simulation which ranges from switch level to full spice simulations. The relative merits and disadvantages of various simulation tools used throughout this project are discussed in Chapter 4.

The layout of the entire buffer chip is shown in Figure 7.1 where the input control (denoted by bcic), the buffer manager (denoted by bcbm) and the output control (denoted by bcoc) are the author's contribution on this chip design. Other components of the chip, that is, the dynamic memory (DRAM) which constitutes some 46,000 transistors, the input serial-to-parallel converters (S/P converters), the output parallel-to-serial converters (P/S converters), the clock drivers and the pads were designed by Mr. Jens Jakobsen at Jydsk Telefon, Denmark. The buffer chip has a total die area of 26.1mm^2 and contains over 70,000 transistors. HSpice simulation shows this chip has a power dissipation of less than 1W at a supply voltage of 1.5V while operating at a 125°C environment. This chip was sent to fabrication on November, 1993 and is expected to be delivered on April, 1994.

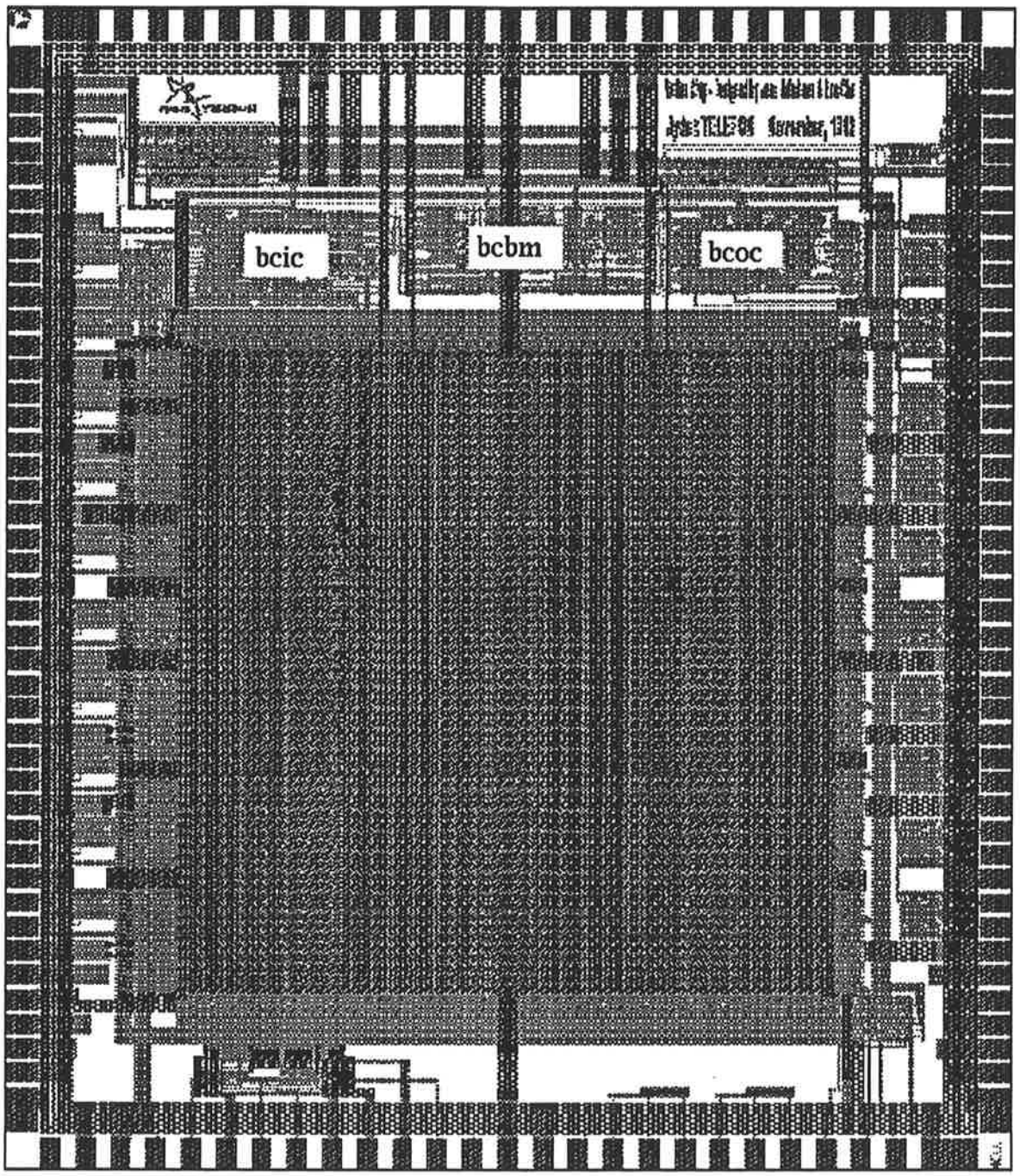


Figure 7.1: Layout of the entire buffer chip

7.2 Future Work

• The most important future work would be to test the chip after delivery and compare the simulated results with the actual measurements to verify the accuracy of the device models and the design methodology.

• Finish the design of the multiplexer chip and construct a *2 x 2* switch so that further properties of the switching elements can be characterised.

• Investigate the systems level specifications such as printed circuit board (PCB) layout, chip-to-chip communications and cooling mechanisms.

• Continue the search for better logic families to enable higher operational speed while preserving low power consumption and low noise injection to the power busses.

• Explore the idea of a hybrid approach to the implementation of the ATM switch with GaAs and CMOS technologies. For instance GaAs technology may be used as an interface to multiplex/demultiplex high data rate cell streams so that the internal operation speed can be reduced to a frequency at which CMOS logic circuits can operate.

Appendix A. VHDL Structural Description of the Input Control

The VHDL structural description of the entire input control is shown below:

```
--------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcic.vst
-- Function:    Buffer chip---Input Circuit
-- Author:      Eric Chu
-- Date :       11th October, 1993
--------------------------------------------------------------------------------
-- Entity Declaration

ENTITY bcic IS
 port (
 vss,vdd: bit;                    -- Power supply
 phi1, phi2: bit;                 -- Clock
 Rst: bit;                        -- Reset
 IDIH: bit;                       -- Input data internal header
 IBA: bit_vector (4 downto 0);    -- Input block address
 IBG: bit;                        -- Input block granted
 IBNG: bit;                       -- Input Block Not Granted
 IC: inout bit;                   -- Input Conversion
 IBR: out bit_vector (1 downto 0); -- Input Block Request
 WE: out bit;                     -- Write Enable
 WA: out bit_vector (6 downto 0)  -- Write Address
 );
END bcic;

-- Architecture Declaration

ARCHITECTURE structural_view OF bcic IS

 COMPONENT icmc
  port (vss,vdd,phi1,phi2,SysRst,Write: BIT; PL: inout BIT; WriteEnable: out BIT);
 END COMPONENT;

 COMPONENT icpg
  port (vss,vdd,phi1,phi2,SysRst,Q13,Q26,Q39,Q52,BWfb: BIT; Write: out BIT; Up,
Upb, ClrCount: inout BIT);
 END COMPONENT;

 COMPONENT iccae
  port (vss,vdd,phi1,phi2,SysRst,IH: BIT;
StartPulse,StartPulseb,Q11,Q11b,Q26,Q39,Q52: inout BIT);
 END COMPONENT;

 COMPONENT icbr
  port (vss,vdd,phi1,phi2,SysRst,SPb,IH,BG,BNG,Q11b,A6,A5,A4,A3,A2: BIT;
BR0f,BR1f: out BIT; BWfb: inout BIT; WA6,WA5,WA4,WA3,WA2: out BIT);
 END COMPONENT;
```

```
 COMPONENT count
   port (vss,vdd,phi1,phi2,Clear,up,upb: BIT; WA1,WA0: out BIT);
 END COMPONENT;

 COMPONENT ldddd
   port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lddsd
   port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lsdud
   port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;


-- SIGNAL DECLARATION

  SIGNAL  StartPulse,StartPulseb,Q11,Q11b,Q26,Q39,Q52: BIT;

  SIGNAL  BWfb,Write,Up,Upb,ClrCount: BIT;

  SIGNAL  Q12,Q12b,Q13,Q13b: BIT;

  SIGNAL  QRst,QRstb,SysRst,SysRstb: BIT;


-- Structural Description

BEGIN

   cae:  iccae PORT MAP
(vss,vdd,phi1,phi2,SysRst,IDIH,StartPulse,StartPulseb,Q11,Q11b,
Q26,Q39,Q52);

   pg:    icpg PORT MAP
(vss,vdd,phi1,phi2,SysRst,Q13,Q26,Q39,Q52,BWfb,Write,Up,Upb,
ClrCount);

   mc:    icmc PORT MAP (vss,vdd,phi1,phi2,SysRst,Write,IC,WE);

   up2:   count PORT MAP (vss,vdd,phi1,phi2,ClrCount,Up,Upb,WA(1),WA(0));

   br:    icbr PORT MAP (vss,vdd,phi1,phi2,SysRst,StartPulseb,IDIH,IBG,IBNG,Q11b,
IBA(4),IBA(3),IBA(2),IBA(1),IBA(0),IBR(0),IBR(1),BWfb,WA(6),WA(5),WA(4),
WA(3),WA(2));

   dffd1: ldddd PORT MAP (vss,vdd,phi1,phi2,Q11,Q12,Q12b);
   dffs1: lddsd PORT MAP (vss,vdd,phi1,phi2,Q12,Q13,Q13b);

   dffd2: ldddd PORT MAP (vss,vdd,phi1,phi2,Rst,QRst,QRstb);
   dffu1: lsdud PORT MAP (vss,vdd,phi1,phi2,QRst,SysRst,SysRstb);

end structural_view;
```

Structural Description of the Cell Arrival Extraction module within the input control:

```
--------------------------------------------------------------------------------
-- VHDL structorial description of module
-- File name:   iccae.vst
-- Function:    Buffer Chip---Input Control Cell Arrival Extraction
-- Author:      Eric Chu
-- Date :       11th October, 1993
--------------------------------------------------------------------------------


-- Entity Declaration

ENTITY iccae IS
PORT (vss,vdd,phi1,phi2,SysRst,IH: BIT;
StartPulse,StartPulseb,Q11,Q11b,Q26,Q39,Q52: inout BIT);
END iccae;


-- Architecture Declaration

ARCHITECTURE structural_view OF iccae IS

 COMPONENT nd
   port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o2nd
   port (vss,vdd,i0,i1: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o2ns
   port (vss,vdd,i0,i1: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o3nd
   port (vss,vdd,i0,i1,i2: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT ldddd
   port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lddss
   port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lsdud
   port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lsduu
   port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;
```

```
-- Signal Declaration

 SIGNAL   S0,S1,S2,S3,S4,S5: BIT;
 SIGNAL   S0b,S1b,S2b,S3b,S4b,S5b: BIT;
 SIGNAL  Q26b,Q39b,Q52b: BIT;
 SIGNAL  IHb: BIT;
 SIGNAL  IH_DISABLE: BIT;
 SIGNAL  IH_DISABLEb: BIT;
 SIGNAL  START: BIT;
 SIGNAL  DIH: BIT;
 SIGNAL  DIHb: BIT;
 SIGNAL  RESET: BIT;
 SIGNAL  RESETb: BIT;
 SIGNAL  D1: BIT;
 SIGNAL  D1b: BIT;
 SIGNAL  out1: BIT;
 SIGNAL  out2: BIT;
 SIGNAL  out3: BIT;
 SIGNAL  out7: BIT;
 SIGNAL  out7b: BIT;
 SIGNAL  DS5: BIT;
 SIGNAL  DS4: BIT;
 SIGNAL  DS3: BIT;
 SIGNAL  DS2: BIT;
 SIGNAL  DS1: BIT;
 SIGNAL  DS0: BIT;
 SIGNAL   S0bx,S5bx: BIT;

-- Additional internal signals for decoding logics:
-- Decode 12

 SIGNAL  out8: BIT;
 SIGNAL  out9: BIT;
 SIGNAL  out10: BIT;
 SIGNAL  Qout8,Qout8b,Qout9,Qout9b: BIT;

-- Decode 26

 SIGNAL  out11: BIT;
 SIGNAL  out12: BIT;
 SIGNAL  out13: BIT;
 SIGNAL  Qout11,Qout11b,Qout12,Qout12b: BIT;

-- Decode 39

 SIGNAL  out14: BIT;
 SIGNAL  out15: BIT;
 SIGNAL  out16: BIT;
 SIGNAL  Qout14,Qout14b,Qout15,Qout15b: BIT;

-- Decode 52

 SIGNAL  out17: BIT;
 SIGNAL  out18: BIT;
 SIGNAL  out19: BIT;
 SIGNAL  Qout17,Qout17b,Qout18,Qout18b: BIT;
 SIGNAL  S5by: BIT;
```

```
-- Structural Description

BEGIN

  invd1:    nd  PORT MAP (vss,vdd,IH,IHb);
  dffd0:  ldddd  PORT MAP (vss,vdd,phi1,phi2,IHb,DIHb,DIH);
  nor2s1:  o2ns  PORT MAP (vss,vdd,DIHb,IH_DISABLE,START);
  nor2d1:  o2nd  PORT MAP (vss,vdd,RESET,D1b,D1);
  nor2d2:  o2nd  PORT MAP (vss,vdd,D1,START,D1b);
  dffd1:  ldddd  PORT MAP (vss,vdd,phi1,phi2,D1,IH_DISABLE,IH_DISABLEb);
  dffs1:  lddss  PORT MAP (vss,vdd,phi1,phi2,START,StartPulse,StartPulseb);

-- Pseudo Random Counter

-- 6th Bit
  invx1:    nd  PORT MAP (vss,vdd,S0,S0bx);
  invx2:    nd  PORT MAP (vss,vdd,S5,S5bx);
  nor2d3:  o2nd  PORT MAP (vss,vdd,S0,S5bx,out1);
  nor2d4:  o2nd  PORT MAP (vss,vdd,S5,S0bx,out2);
  nor3d1:  o3nd  PORT MAP (vss,vdd,RESET,out1,out2,DS5);
  dffu1:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS5,S5,S5b);

-- 5th Bit
  nor2d5:  o2nd  PORT MAP (vss,vdd,RESET,S5b,DS4);
  dffu2:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS4,S4,S4b);

-- 4th Bit
  nor2d6:  o2nd  PORT MAP (vss,vdd,RESET,S4b,DS3);
  dffu3:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS3,S3,S3b);

-- 3th Bit
  nor2d7:  o2nd  PORT MAP (vss,vdd,RESET,S3b,DS2);
  dffu4:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS2,S2,S2b);

-- 2nd Bit
  nor2d8:  o2nd  PORT MAP (vss,vdd,RESET,S2b,DS1);
  dffu5:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS1,S1,S1b);

-- 1st Bit
  nor2d9:  o2nd  PORT MAP (vss,vdd,RESET,S1b,out3);
  nor2d10:  o2nd  PORT MAP (vss,vdd,START,out3,DS0);
  dffu6:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS0,S0,S0b);

-- End of Pseudo Random Counter

-- End of Cell Decode

  nor2d12:  o2nd  PORT MAP (vss,vdd,Q52,SysRst,out7);
  invd2:    nd  PORT MAP (vss,vdd,out7,out7b);
  dffu0:  lsdud  PORT MAP (vss,vdd,phi1,phi2,out7b,RESET,RESETb);

-- Decode 11 (Actually 7)

  nor3d4:  o3nd  PORT MAP (vss,vdd,S0,S1b,S2b,out8);
  dffd4:  ldddd  PORT MAP (vss,vdd,phi1,phi2,out8,Qout8,Qout8b);
  nor3d5:  o3nd  PORT MAP (vss,vdd,S3b,S4b,S5b,out9);
  dffd5:  ldddd  PORT MAP (vss,vdd,phi1,phi2,out9,Qout9,Qout9b);
```

```
  nor2d13: o2nd  PORT MAP (vss,vdd,Qout8b,Qout9b,out10);
  dffs3:  lddss  PORT MAP (vss,vdd,phi1,phi2,out10,Q11,Q11b);

-- Decode 26 (Actually 22)

  nor3d6:  o3nd  PORT MAP (vss,vdd,S0,S1b,S2,out11);
  dffd6:  ldddd  PORT MAP (vss,vdd,phi1,phi2,out11,Qout11,Qout11b);
  nor3d7:  o3nd  PORT MAP (vss,vdd,S3b,S4b,S5b,out12);
  dffd7:  ldddd  PORT MAP (vss,vdd,phi1,phi2,out12,Qout12,Qout12b);
  nor2d14: o2nd  PORT MAP (vss,vdd,Qout11b,Qout12b,out13);
  dffs4:  lddss  PORT MAP (vss,vdd,phi1,phi2,out13,Q26,Q26b);

-- Decode 39 (Actually 35)

  nor3d8:  o3nd  PORT MAP (vss,vdd,S0,S1,S2,out14);
  dffd8:  ldddd  PORT MAP (vss,vdd,phi1,phi2,out14,Qout14,Qout14b);
  nor3d9:  o3nd  PORT MAP (vss,vdd,S3b,S4b,S5b,out15);
  dffd9:  ldddd  PORT MAP (vss,vdd,phi1,phi2,out15,Qout15,Qout15b);
  nor2d15: o2nd  PORT MAP (vss,vdd,Qout14b,Qout15b,out16);
  dffs5:  lddss  PORT MAP (vss,vdd,phi1,phi2,out16,Q39,Q39b);

-- Decode 52 (Actually 48)

  invd3:    nd  PORT MAP (vss,vdd,S5b,S5by);
  nor3d10: o3nd  PORT MAP (vss,vdd,S0,S1b,S2,out17);
  dffd10: ldddd  PORT MAP (vss,vdd,phi1,phi2,out17,Qout17,Qout17b);
  nor3d11: o3nd  PORT MAP (vss,vdd,S3,S4b,S5by,out18);
  dffd11: ldddd  PORT MAP (vss,vdd,phi1,phi2,out18,Qout18,Qout18b);
  nor2d16: o2nd  PORT MAP (vss,vdd,Qout17b,Qout18b,out19);
  dffs6:  lddss  PORT MAP (vss,vdd,phi1,phi2,out19,Q52,Q52b);

end structural_view;
```

Structural Description of the Block Requester module within the input control:

```
--------------------------------------------------------------------------------
-- VHDL structorial description of module
-- File name:  icbr.vst
-- Function:   Buffer chip---Input Control Block Requester
-- Author:     Eric Chu
-- Date :      8th October, 1993
--------------------------------------------------------------------------------


-- Entity Declaration

ENTITY icbr IS
   PORT (vss,vdd,phi1,phi2,SysRst,SPb,IH,BG,BNG,Q11b,A6,A5,A4,A3,A2: BIT;
BR0f,BR1f: out BIT; BWfb: inout BIT; WA6,WA5,WA4,WA3,WA2: out BIT);
END icbr;
```

```
-- Architecture Declaration

ARCHITECTURE structural_view OF icbr IS

 COMPONENT ns
  port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT nu
  port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o2nd
  port (vss,vdd,i0,i1: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o2ns
  port (vss,vdd,i0,i1: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o3nd
  port (vss,vdd,i0,i1,i2: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT jdd
  port (vss,vdd,Ldb,D: BIT; Q,Qb: out BIT);
 END COMPONENT;

 COMPONENT jds
  port (vss,vdd,Ldb,D: BIT; Q,Qb: out BIT);
 END COMPONENT;

 COMPONENT jpsd
  port (vss,vdd,Set,Ldb,D: BIT; Q,Qb: inout BIT);
 END COMPONENT;

 COMPONENT ldddd
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lddsd
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lsdud
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT sink
  port (vss,vdd,i: BIT);
 END COMPONENT;


-- Signal Declaration

 SIGNAL  QIH,QIHb: BIT;
 SIGNAL  D2,Q2,Q2b,D3,Q3,Q3b: BIT;
```

```
  SIGNAL  br1,br1b,br0,br0b,br: BIT;
  SIGNAL  out1,out2: BIT;
  SIGNAL  QBG,QBGb,QQBG,QQBGb,QBNG,QBNGb,QQBNG,QQBNGb: BIT;
  SIGNAL  BG1,BG1b,BWf: BIT;
  SIGNAL  QA2,QA3,QA4,QA5,QA6,QA2b,QA3b,QA4b,QA5b,QA6b: BIT;
  SIGNAL  QQA2,QQA3,QQA4,QQA5,QQA6,QQA2b,QQA3b,QQA4b,QQA5b,QQA6b:
BIT;
  SIGNAL  Q12,Q12b,Q13,Q13b,Q14,Q14b,Q15,Q15b: BIT;

-- Structural Description

BEGIN

   dffd1:   ldddd  PORT MAP (vss,vdd,phi1,phi2,IH,QIH,QIHb);

   nor2d1:  o2nd  PORT MAP (vss,vdd,QIHb,SPb,D2);
   nor2d2:  o2nd  PORT MAP (vss,vdd,QIH,SPb,D3);

   dffd2:   ldddd  PORT MAP (vss,vdd,phi1,phi2,D2,Q2,Q2b);
   dffd3:   ldddd  PORT MAP (vss,vdd,phi1,phi2,D3,Q3,Q3b);

   nor2s1:  o2ns  PORT MAP (vss,vdd,br1,Q2,br1b);
   invu1:    nu  PORT MAP (vss,vdd,br1b,BR1f);
   nor2d3:  o2nd  PORT MAP (vss,vdd,br1b,out2,br1);

   nor2s2:   o2ns PORT MAP (vss,vdd,br0,Q3,br0b);
   invu2:    nu   PORT MAP (vss,vdd,br0b,BR0f);
   nor2d4:   o2nd PORT MAP (vss,vdd,br0b,out2,br0);

   nor3d1:   o3nd PORT MAP (vss,vdd,QQBNG,SysRst,QQBG,out1);
   invs1:    ns   PORT MAP (vss,vdd,out1,out2);

   nor2s3:   o2ns PORT MAP (vss,vdd,br0,br1,br);

-- Getting Q11b's
   dffs1:   lddsd PORT MAP (vss,vdd,phi1,phi2,Q11b,Q12b,Q12);
   dffd4:   lddd PORT MAP (vss,vdd,phi1,phi2,Q12b,Q13b,Q13);
   dffd5:   lddd PORT MAP (vss,vdd,phi1,phi2,Q13b,Q14b,Q14);
   dffu1:   lsdud PORT MAP (vss,vdd,phi1,phi2,Q14b,Q15b,Q15);

-- Extra Logics for Delaying Block Write Signal
   latch1:    jdd PORT MAP (vss,vdd,br,BG,BG1,BG1b);
   sink1:    sink PORT MAP (vss,vdd,BG1);
   latch2:   jpsd PORT MAP (vss,vdd,SysRst,Q12b,BG1b,BWfb,BWf);
   sink2:    sink PORT MAP (vss,vdd,BWfb);

--- Latching Addresses
   latch3:    jdd PORT MAP (vss,vdd,br,A2,QA2,QA2b);
   latch4:    jdd PORT MAP (vss,vdd,br,A3,QA3,QA3b);
   latch5:    jdd PORT MAP (vss,vdd,br,A4,QA4,QA4b);
   latch6:    jdd PORT MAP (vss,vdd,br,A5,QA5,QA5b);
   latch7:    jdd PORT MAP (vss,vdd,br,A6,QA6,QA6b);
   sink3:    sink PORT MAP (vss,vdd,QA2b);
   sink4:    sink PORT MAP (vss,vdd,QA3b);
   sink5:    sink PORT MAP (vss,vdd,QA4b);
   sink6:    sink PORT MAP (vss,vdd,QA5b);
   sink7:    sink PORT MAP (vss,vdd,QA6b);
```

```
  latch8:    jds PORT MAP (vss,vdd,Q15b,QA2,QQA2,QQA2b);
  latch9:    jds PORT MAP (vss,vdd,Q15b,QA3,QQA3,QQA3b);
  latch10:   jds PORT MAP (vss,vdd,Q15b,QA4,QQA4,QQA4b);
  latch11:   jds PORT MAP (vss,vdd,Q15b,QA5,QQA5,QQA5b);
  latch12:   jds PORT MAP (vss,vdd,Q15b,QA6,QQA6,QQA6b);
  sink8:    sink PORT MAP (vss,vdd,QQA2);
  sink9:    sink PORT MAP (vss,vdd,QQA3);
  sink10:   sink PORT MAP (vss,vdd,QQA4);
  sink11:   sink PORT MAP (vss,vdd,QQA5);
  sink12:   sink PORT MAP (vss,vdd,QQA6);

  invu3:      nu PORT MAP (vss,vdd,QQA2b,WA2);
  invu4:      nu PORT MAP (vss,vdd,QQA3b,WA3);
  invu5:      nu PORT MAP (vss,vdd,QQA4b,WA4);
  invu6:      nu PORT MAP (vss,vdd,QQA5b,WA5);
  invu7:      nu PORT MAP (vss,vdd,QQA6b,WA6);

--- Input Dff's for Meta-Stability
  dffd6:   ldddd PORT MAP (vss,vdd,phi1,phi2,BG,QBG,QBGb);
  dffd7:   ldddd PORT MAP (vss,vdd,phi1,phi2,QBG,QQBG,QQBGb);
  dffd8:   ldddd PORT MAP (vss,vdd,phi1,phi2,BNG,QBNG,QBNGb);
  dffd9:   ldddd PORT MAP (vss,vdd,phi1,phi2,QBNG,QQBNG,QQBNGb);

end structural_view;
```

Structural Description of the Pulse Generation module within the input control:

```
-------------------------------------------------------------------------------
-- VHDL structorial description of module
-- File name:   icpg.vst
-- Function:    Buffer chip---Input Control---Pulse Generation
-- Author:      Eric Chu
-- Date :       11th October, 1993
-------------------------------------------------------------------------------


-- Entity Declaration

ENTITY icpg IS
   PORT (vss,vdd,phi1,phi2,SysRst,Q13,Q26,Q39,Q52,BWfb: BIT; Write: out BIT;
Up,Upb,ClrCount: inout BIT);
END icpg;


-- Architecture Declaration

ARCHITECTURE structural_view OF icpg IS

 COMPONENT nd
  port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;
```

```
  COMPONENT o2nd
    port (vss,vdd,i0,i1: BIT; o: out BIT);
  END COMPONENT;

  COMPONENT o3ns
    port (vss,vdd,i0,i1,i2: BIT; o: out BIT);
  END COMPONENT;

  COMPONENT o4ns
    port (vss,vdd,i0,i1,i2,i3: BIT; o: out BIT);
  END COMPONENT;

  COMPONENT lddsd
    port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
  END COMPONENT;

  SIGNAL selectb: BIT;
  SIGNAL out1: BIT;
  SIGNAL out2,out2b,ClrCountb: BIT;

BEGIN

-- Generating Write

    nor4s1:  o4ns PORT MAP (vss,vdd,Q13,Q26,Q39,Q52,selectb);
    nor2d1:  o2nd PORT MAP (vss,vdd,selectb,BWfb,Write);


-- Generating Up & Up* for 2-bit counter

    nor3s1:  o3ns PORT MAP (vss,vdd,Q26,Q39,Q52,out1);
    dffs1:  lddsd PORT MAP (vss,vdd,phi1,phi2,out1,Upb,Up);

-- Generating Counter Clear

    nor2d2:  o2nd PORT MAP (vss,vdd,Q13,SysRst,out2);
    invd1:    nd PORT MAP (vss,vdd,out2,out2b);
    dffs2:  lddsd PORT MAP (vss,vdd,phi1,phi2,out2b,ClrCount,ClrCountb);

end structural_view;
```

Structural Description of the Memory Control module within the input control:

```
-------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   icmc.vst
-- Function:    Buffer Chip---Input Control Memory Control
-- Author:      Eric Chu
-- Date :       11th October, 1993
-- Modified:    Eric Chu 27th October, 1993
-- Comments:    Increased driving capability of the signal IC
-------------------------------------------------------------------------------


-- Entity Declaration

ENTITY icmc IS
  PORT (vss,vdd,phi1,phi2,SysRst,Write: BIT; PL: inout BIT; WriteEnable: out BIT);
END icmc;


-- Architecture Declaration

ARCHITECTURE structural_view OF icmc IS

 COMPONENT nu
  port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT ldddd
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lsdud
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT rrsds
  port (vss,vdd,r0,r1,s: BIT; q,qb: inout BIT);
 END COMPONENT;

 COMPONENT sink
  port (vss,vdd,i: BIT);
 END COMPONENT;

-- Signal Declaration

 SIGNAL PLb: BIT;
 SIGNAL QPL,QPLb,QQPL,QQPLb: BIT;
 SIGNAL W,Wb: BIT;
 SIGNAL
Rst1,Rst1b,Rst2,Rst2b,Rst3,Rst3b,Rst4,Rst4b,Rst5,Rst5b,Rst6,Rst6b,Rst7,Rst7b,Rst8,R
st8b,Rst9,Rst9b,Rst10,Rst10b,Rst11,Rst11b: BIT;
```

```
-- Structural Description

BEGIN

  dff:    lsdud PORT MAP (vss,vdd,phi1,phi2,Write,PL,PLb);

  dffd1:  ldddd PORT MAP (vss,vdd,phi1,phi2,PL,QPL,QPLb);
  dffd2:  ldddd PORT MAP (vss,vdd,phi1,phi2,QPL,QQPL,QQPLb);

  rrsds1: rrsds PORT MAP (vss,vdd,SysRst,Rst11,QQPL,W,Wb);
  sink1:   sink PORT MAP (vss,vdd,W);
  invu1:     nu PORT MAP (vss,vdd,Wb,WriteEnable);

  dffd3:  ldddd PORT MAP (vss,vdd,phi1,phi2,QQPL,Rst1,Rst1b);
  dffd4:  ldddd PORT MAP (vss,vdd,phi1,phi2,Rst1,Rst2,Rst2b);
  dffd5:  ldddd PORT MAP (vss,vdd,phi1,phi2,Rst2,Rst3,Rst3b);
  dffd6:  ldddd PORT MAP (vss,vdd,phi1,phi2,Rst3,Rst4,Rst4b);
  dffd7:  ldddd PORT MAP (vss,vdd,phi1,phi2,Rst4,Rst5,Rst5b);
  dffd8:  ldddd PORT MAP (vss,vdd,phi1,phi2,Rst5,Rst6,Rst6b);
  dffd9:  ldddd PORT MAP (vss,vdd,phi1,phi2,Rst6,Rst7,Rst7b);
  dffd10: ldddd PORT MAP (vss,vdd,phi1,phi2,Rst7,Rst8,Rst8b);
  dffd11: ldddd PORT MAP (vss,vdd,phi1,phi2,Rst8,Rst9,Rst9b);
  dffd12: ldddd PORT MAP (vss,vdd,phi1,phi2,Rst9,Rst10,Rst10b);
  dffd13: ldddd PORT MAP (vss,vdd,phi1,phi2,Rst10,Rst11,Rst11b);

end structural_view;
```

Structural Description of the 2-Bit Counter within the input control:

```
--------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   count.vst
-- Function:    Counter for Lower 2-Bit Write Address Generation
-- Author:      Eric Chu
-- Date :       27th September, 1993
--------------------------------------------------------------------------------


-- Entity Declaration

ENTITY count IS
PORT (vss,vdd,phi1,phi2,Clear,up,upb: BIT; WA1,WA0: out BIT);
END count;


-- Architecture Declaration

ARCHITECTURE structural_view OF count IS

 COMPONENT nu
  port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;
```

```
  COMPONENT o2nd
    port (vss,vdd,i0,i1: BIT; o: out BIT);
  END COMPONENT;

  COMPONENT o3nd
    port (vss,vdd,i0,i1,i2: BIT; o: out BIT);
  END COMPONENT;

  COMPONENT lddss
    port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
  END COMPONENT;


-- Signal Declaration

  SIGNAL DA,DB: BIT;
  SIGNAL QB,QBb,QA,QAb: BIT;
  SIGNAL out1,out2,out3,out4: BIT;


-- Structural Description

BEGIN

  lddss1: lddss PORT MAP (vss,vdd,phi1,phi2,DA,QA,QAb);
  lddss2: lddss PORT MAP (vss,vdd,phi1,phi2,DB,QB,QBb);

-- Input to Dff A:

  nor2d1: o2nd PORT MAP (vss,vdd,Up,QA,out1);
  nor2d2: o2nd PORT MAP (vss,vdd,Upb,QAb,out2);
  nor3d1: o3nd PORT MAP (vss,vdd,out1,out2,Clear,DA);

-- Input to Dff B:

  nor2d3: o2nd PORT MAP (vss,vdd,Upb,QAb,out3);
  nor2d4: o2nd PORT MAP (vss,vdd,out3,QB,out4);
  nor2d5: o2nd PORT MAP (vss,vdd,out4,Clear,DB);

-- Buffering Write Addresses

  invu1:  nu   PORT MAP (vss,vdd,QBb,WA1);
  invu2:  nu   PORT MAP (vss,vdd,QAb,WA0);

end structural_view;
```

Appendix B. VHDL Structural Description of the Buffer Manager

The VHDL structural description of the entire buffer manager is shown below:

```
--------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcbm.vst
-- Function:    Buffer chip buffer manager module
-- Author:      Jens Jakobsen
-- Date :       September 4 1993
--------------------------------------------------------------------------------

-- Entity Declaration

ENTITY bcbm IS
 PORT (
 vss,vdd: bit;                    -- Power supply
 phi1, phi2, Rst: bit;            -- Reset and clock
 IBR: bit_vector (1 downto 0);    -- Input block request
 OBR: bit;                        -- Output block request
 IBG: out bit;                    -- Input block granted
 IBNG: out bit;                   -- Input block not granted
 OBG: out bit;                    -- Output block granted
 OBNG: out bit;                   -- Output block not granted
 IBA: out bit_vector (4 downto 0); -- Input block address
 OBA: out bit_vector (4 downto 0)  -- Output block address
 );
END bcbm;


-- Architecture Declaration
ARCHITECTURE structural OF bcbm IS

 COMPONENT bcbmc5u
  PORT (
  vss : in BIT;                   -- vss
  vdd : in BIT;                   -- vdd
  phi1 : in BIT;                  -- phi1
  phi2 : in BIT;                  -- phi2
  Rst : in BIT;                   -- reset
  upb : in BIT;                   -- Up
  s: out bit_vector (4 downto 0)  -- Counter output
  );
 END COMPONENT;

 COMPONENT bcbmud
  PORT (
  vss : in BIT;                   -- vss
  vdd : in BIT;                   -- vdd
  phi1 : in BIT;                  -- phi1
  phi2 : in BIT;                  -- phi2
```

```
  Rst : in BIT;                    -- reset
  upb : in BIT;                    -- Up
  dnb : in BIT;                    -- Down
  s,sb: out bit_vector (4 downto 0)   -- Counter output
  );
END COMPONENT;

COMPONENT bcbmbr
  PORT (
  vss,vdd: bit;                    -- Power supply
  phi1, phi2: bit;                 -- Clock
  Rst : in BIT;                    -- Reset
  BR : in BIT;                     -- Block request
  size : in BIT;                   -- Empty or full
  BGR,BNGR : out BIT               -- Block granted or not granted
  );
END COMPONENT;

COMPONENT bcbmic
  PORT (
  vss : in BIT;                    -- vss
  vdd : in BIT;                    -- vdd
  phi1 : in BIT;                   -- phi1
  phi2 : in BIT;                   -- phi2
  Rst : in BIT;                    -- reset
  s,sb : in bit_vector (5 downto 0);  -- Counter state
  IBR: bit_vector (1 downto 0);       -- Input block request
  IBG: out bit;                    -- Input block granted
  IBNG: out bit;                   -- Input block not granted
  up : in BIT                      -- Up
  );
END COMPONENT;

COMPONENT bcbmoc
  PORT (
  vss : in BIT;                    -- vss
  vdd : in BIT;                    -- vdd
  phi1 : in BIT;                   -- phi1
  phi2 : in BIT;                   -- phi2
  Rst : in BIT;                    -- reset
  s,sb : in bit_vector (5 downto 0);  -- Counter state
  OBR: bit;                        -- Output block request
  OBG: out bit;                    -- Output block granted
  OBNG: out bit;                   -- Output block not granted
  dn : in BIT                      -- Down
  );
END COMPONENT;

COMPONENT ldddd
  PORT (
  vss : in BIT;      -- vss
  vdd : in BIT;      -- vdd
  phi1 : in BIT;     -- phi1
  phi2 : in BIT;     -- phi2
  D : in BIT;        -- D
  Q : inout BIT;     -- Q
  QB : inout BIT     -- QB
```

```
);
END COMPONENT;

COMPONENT lsdud
 PORT (
 vss : in BIT;      -- vss
 vdd : in BIT;      -- vdd
 phi1 : in BIT;     -- phi1
 phi2 : in BIT;     -- phi2
 D : in BIT;       -- D
 Q : inout BIT;     -- Q
 QB : inout BIT     -- QB
);
END COMPONENT;

COMPONENT nd
 PORT (
 vss : in BIT;              -- vss
 vdd : in BIT;              -- vdd
 i : in BIT;               -- in
 o: out bit                -- out
 );
END COMPONENT;

COMPONENT nu
 PORT (
 vss : in BIT;              -- vss
 vdd : in BIT;              -- vdd
 i : in BIT;               -- in
 o: out bit                -- out
 );
END COMPONENT;

COMPONENT ns
 PORT (
 vss : in BIT;              -- vss
 vdd : in BIT;              -- vdd
 i : in BIT;               -- in
 o: out bit                -- out
 );
END COMPONENT;

COMPONENT o2nd
 PORT (
 vss : in BIT;              -- vss
 vdd : in BIT;              -- vdd
 i0 : in BIT;              -- in
 i1 : in BIT;              -- in
 o: out bit                -- out
 );
END COMPONENT;

COMPONENT o2ns
 PORT (
 vss : in BIT;              -- vss
 vdd : in BIT;              -- vdd
 i0 : in BIT;              -- in
```

```
  i1 : in BIT;                -- in
  o: out bit                  -- out
  );
END COMPONENT;

COMPONENT o2dd
  PORT (
  vss : in BIT;               -- vss
  vdd : in BIT;               -- vdd
  i0 : in BIT;                -- in
  i1 : in BIT;                -- in
  o: out bit                  -- out
  );
END COMPONENT;

COMPONENT o3ns
  PORT (
  vss : in BIT;               -- vss
  vdd : in BIT;               -- vdd
  i0 : in BIT;                -- in
  i1 : in BIT;                -- in
  i2 : in BIT;                -- in
  o: out bit                  -- out
  );
END COMPONENT;

COMPONENT o3dd
  PORT (
  vss : in BIT;               -- vss
  vdd : in BIT;               -- vdd
  i0 : in BIT;                -- in
  i1 : in BIT;                -- in
  i2 : in BIT;                -- in
  o: out bit                  -- out
  );
END COMPONENT;


-------------------------------------------------------------------------------
-- Internal signals
-------------------------------------------------------------------------------
SIGNAL QueueSize,Queuesizeb: bit_vector (4 downto 0);

SIGNAL RstQ1,RstQ1b,RstQ2,RstQ2b : bit;
SIGNAL full: bit_vector (1 downto 0);
SIGNAL fulla,fullb : bit;
SIGNAL empty,emptya,emptyb : bit;

SIGNAL    IBGx,IBNGx        : bit_vector (1 downto 0);
SIGNAL    IBGb,IBNGb        : bit;
SIGNAL    OBGx,OBNGx        : bit;
SIGNAL    OBGb,OBNGb        : bit;

SIGNAL    IBGQ1,IBGQ1b      : bit;
SIGNAL    IBGy              : bit;
SIGNAL    IBC,IBCb          : bit;
SIGNAL    IBCQ1,IBCQ1b      : bit;
```

```
  SIGNAL      OBGQ1,OBGQ1b        : bit;
  SIGNAL      OBGy                : bit;
  SIGNAL      OBC,OBCb            : bit;
  SIGNAL      OBCQ1,OBCQ1b        : bit;

  SIGNAL      UP,UPb              : bit;
  SIGNAL      DN,DNb              : bit;

BEGIN

-------------------------------------------------------------------------------
-- Handle metastability at RST by using two flip flops
-------------------------------------------------------------------------------

  l0: ldddd PORT MAP (vss,vdd,phi1,phi2,Rst,RstQ1,RstQ1b);
  l1: lsdud PORT MAP (vss,vdd,phi1,phi2,RstQ1,RstQ2,RstQ2b);

-------------------------------------------------------------------------------
-- Input block request
-------------------------------------------------------------------------------

  bcbmibr0:bcbmbr PORT MAP
(vss,vdd,phi1,phi2,RstQ2,IBR(0),full(0),IBGx(0),IBNGx(0));
  bcbmibr1:bcbmbr PORT MAP
(vss,vdd,phi1,phi2,RstQ2,IBR(1),full(1),IBGx(1),IBNGx(1));
  o2n0: o2ns PORT MAP (vss,vdd,IBGx(0),IBGx(1),IBGb);
  o2n1: o2ns PORT MAP (vss,vdd,IBNGx(0),IBNGx(1),IBNGb);
  n0: nu PORT MAP (vss,vdd,IBGb,IBG);
  n1: nu PORT MAP (vss,vdd,IBNGb,IBNG);

-------------------------------------------------------------------------------
-- Output block request
-------------------------------------------------------------------------------

  bcbmobr:bcbmbr PORT MAP (vss,vdd,phi1,phi2,RstQ2,OBR,empty,OBGx,OBNGx);
  n2: ns PORT MAP (vss,vdd,OBGx,OBGb);
  n3: nu PORT MAP (vss,vdd,OBGb,OBG);
  n4: ns PORT MAP (vss,vdd,OBNGx,OBNGb);
  n5: nu PORT MAP (vss,vdd,OBNGb,OBNG);


-------------------------------------------------------------------------------
-- Generate count up pulse
-------------------------------------------------------------------------------
  l2: ldddd PORT MAP (vss,vdd,phi1,phi2,IBGb,IBGQ1b,IBGQ1);
  n6: nd PORT MAP (vss,vdd,IBGb,IBGy);
  o2n2: o2ns PORT MAP (vss,vdd,IBGQ1b,IBGy,IBC);
  n7: nu PORT MAP (vss,vdd,IBC,IBCb);

-------------------------------------------------------------------------------
-- Generate count down pulse
-------------------------------------------------------------------------------
  l3: ldddd PORT MAP (vss,vdd,phi1,phi2,OBGb,OBGQ1b,OBGQ1);
  n8: nd PORT MAP (vss,vdd,OBGb,OBGy);
  o2n3: o2ns PORT MAP (vss,vdd,OBGQ1b,OBGy,OBC);
  n9: nu PORT MAP (vss,vdd,OBC,OBCb);
```

```
-------------------------------------------------------------------------------
-- Input block address counter
-------------------------------------------------------------------------------

 bcbmc5ui:bcbmc5u PORT MAP (vss,vdd,phi1,phi2,RstQ2,IBCb,IBA);

-------------------------------------------------------------------------------
-- Output block address counter
-------------------------------------------------------------------------------

 bcbmc5uo:bcbmc5u PORT MAP (vss,vdd,phi1,phi2,RstQ2,OBCb,OBA);

-------------------------------------------------------------------------------
-- Buffer size counter
-------------------------------------------------------------------------------

-- n10: nd PORT MAP (vss,vdd,IBC,IBCbx);
 lx1: ldddd PORT MAP (vss,vdd,phi1,phi2,IBC,IBCQ1,IBCQ1b);
 o2n4: o2ns PORT MAP (vss,vdd,IBCQ1b,OBCQ1,UP);
 n11: nu PORT MAP (vss,vdd,UP,UPb);

-- n12: nd PORT MAP (vss,vdd,OBC,OBCbx);
 lx2: ldddd PORT MAP (vss,vdd,phi1,phi2,OBC,OBCQ1,OBCQ1b);
 o2n5: o2ns PORT MAP (vss,vdd,OBCQ1b,IBCQ1,DN);
 n13: nu PORT MAP (vss,vdd,DN,DNb);

 bcbmud:bcbmud PORT MAP
(vss,vdd,phi1,phi2,RstQ2,UPb,DNb,QueueSize,QueueSizeb);

-------------------------------------------------------------------------------
-- Empty decoding
-------------------------------------------------------------------------------

 o30: o3dd PORT MAP (vss,vdd,QueueSize(0),QueueSize(1),QueueSize(2),emptya);
 o20: o2dd PORT MAP (vss,vdd,QueueSize(3),QueueSize(4),emptyb);
 o2n6: o2ns PORT MAP (vss,vdd,emptya,emptyb,empty);


-------------------------------------------------------------------------------
-- Full decoding
-------------------------------------------------------------------------------

 o31: o3dd PORT MAP (vss,vdd,QueueSizeb(0),QueueSizeb(1),QueueSizeb(2),fulla);
 o21: o2dd PORT MAP (vss,vdd,QueueSizeb(3),QueueSizeb(4),fullb);
 o2n7: o2ns PORT MAP (vss,vdd,fulla,fullb,full(0));

 o2n8: o2ns PORT MAP (vss,vdd,QueueSizeb(3),QueueSizeb(4),full(1));


END structural;
```

Structural Description of the 5-Bit Up/Down Counter within the buffer manager:

```
-------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcbmud.vst
-- Function:    Buffer chip buffer manager up/down counter
-- Author:      Jens Jakobsen
-- Date :       October 3 1993
-------------------------------------------------------------------------------

-- Entity Declaration

ENTITY bcbmud IS
 PORT (
 vss,vdd: bit;                    -- Power supply
 phi1, phi2: bit;                 -- Clock
 Rst : in BIT;                    -- Reset
 upb : in BIT;                    -- Up
 dnb : in BIT;                    -- Down
 s,sb: out bit_vector (4 downto 0)   -- Counter output
 );
END bcbmud;


-- Architecture Declaration
ARCHITECTURE structural OF bcbmud IS

 COMPONENT bcbmudb
  PORT (
  vss,vdd: bit;                   -- Power supply
  phi1, phi2: bit;                -- Clock
  Rst: bit;                       -- Reset
  i: bit;                         -- Input
  Q,Qb: inout bit                 -- Output
  );
END COMPONENT;


 COMPONENT nd
  PORT (
  vss : in BIT;                   -- vss
  vdd : in BIT;                   -- vdd
  i : in BIT;                     -- in
  o: out bit                      -- out
  );
 END COMPONENT;

 COMPONENT o2nd
  PORT (
  vss : in BIT;                   -- vss
  vdd : in BIT;                   -- vdd
  i0 : in BIT;                    -- in
  i1 : in BIT;                    -- in
  o: out bit                      -- out
  );
 END COMPONENT;
```

```
COMPONENT o3nd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  i2 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;

COMPONENT o2dd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;

COMPONENT o3dd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  i2 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;


-------------------------------------------------------------------------------
-- Internal signals
-------------------------------------------------------------------------------
SIGNAL a,b,c : bit_vector (4 downto 0);
SIGNAL m3a,m3b    : bit;
SIGNAL m4a,m4b,n4a,n4b     : bit;


BEGIN


-------------------------------------------------------------------------------
-- Bit 0
-------------------------------------------------------------------------------
n0a: nd PORT MAP (vss,vdd,Upb,a(0));
n0b: nd PORT MAP (vss,vdd,dnb,b(0));
o2n0: o2nd PORT MAP (vss,vdd,a(0),b(0),c(0));
bcbmudb0: bcbmudb PORT MAP (vss,vdd,phi1,phi2,Rst,c(0),s(0),sb(0));

-------------------------------------------------------------------------------
-- Bit 1
-------------------------------------------------------------------------------
o2n1a: o2nd PORT MAP (vss,vdd,Upb,sb(0),a(1));
```

```
  o2n1b: o2nd PORT MAP (vss,vdd,dnb,s(0),b(1));
  o2n1c: o2nd PORT MAP (vss,vdd,a(1),b(1),c(1));
  bcbmudb1: bcbmudb PORT MAP (vss,vdd,phi1,phi2,Rst,c(1),s(1),sb(1));

-------------------------------------------------------------------------------
-- Bit 2
-------------------------------------------------------------------------------
  o3n2a: o3nd PORT MAP (vss,vdd,Upb,sb(0),sb(1),a(2));
  o3n2b: o3nd PORT MAP (vss,vdd,dnb,s(0),s(1),b(2));
  o2n2c: o2nd PORT MAP (vss,vdd,a(2),b(2),c(2));
  bcbmudb2: bcbmudb PORT MAP (vss,vdd,phi1,phi2,Rst,c(2),s(2),sb(2));

-------------------------------------------------------------------------------
-- Bit 3
-------------------------------------------------------------------------------
  o3m3a: o3dd PORT MAP (vss,vdd,sb(2),sb(1),sb(0),m3a);
  o3m3b: o3dd PORT MAP (vss,vdd,s(2),s(1),s(0),m3b);
  o2n3a: o2nd PORT MAP (vss,vdd,Upb,m3a,a(3));
  o2n3b: o2nd PORT MAP (vss,vdd,dnb,m3b,b(3));
  o2n3c: o2nd PORT MAP (vss,vdd,a(3),b(3),c(3));
  bcbmudb3: bcbmudb PORT MAP (vss,vdd,phi1,phi2,Rst,c(3),s(3),sb(3));

-------------------------------------------------------------------------------
-- Bit 4
-------------------------------------------------------------------------------
  o2m4a: o2dd PORT MAP (vss,vdd,sb(3),sb(2),m4a);
  o2n4a: o2dd PORT MAP (vss,vdd,sb(1),sb(0),n4a);
  o2m4b: o2dd PORT MAP (vss,vdd,s(3),s(2),m4b);
  o2n4b: o2dd PORT MAP (vss,vdd,s(1),s(0),n4b);
  o3n4a: o3nd PORT MAP (vss,vdd,Upb,m4a,n4a,a(4));
  o3n4b: o3nd PORT MAP (vss,vdd,dnb,m4b,n4b,b(4));
  o2n4c: o2nd PORT MAP (vss,vdd,a(4),b(4),c(4));
  bcbmudb4: bcbmudb PORT MAP (vss,vdd,phi1,phi2,Rst,c(4),s(4),sb(4));

END structural;
```

Structural Description of one bit-slice of the logics within the 5-bit up/down counter in the buffer manager:

```
-------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcbmudb.vst
-- Function:    Buffer chip buffer manager counter dff and module XOR
-- Author:      Jens Jakobsen
-- Date :       October 3 1993
-------------------------------------------------------------------------------

-- Entity Declaration

ENTITY bcbmudb IS
  PORT (
```

```
 vss,vdd: bit;                  -- Power supply
 phi1, phi2: bit;               -- Clock
 Rst: bit;                      -- Reset
 i: bit;                        -- Input
 Q,Qb: inout bit                -- Output
 );
END bcbmudb;


-- Architecture Declaration
ARCHITECTURE structural OF bcbmudb IS


 COMPONENT nd
  PORT (
  vss : in BIT;                 -- vss
  vdd : in BIT;                 -- vdd
  i : in BIT;                   -- in
  o: out bit                    -- out
  );
 END COMPONENT;

 COMPONENT muxr
  PORT (
  vss : in BIT;      -- vss
  vdd : in BIT;      -- vdd
  i0 :  in BIT;      -- i0
  i1 :  in BIT;      -- i1
  s0 :  in BIT;      -- s0
  s1 :  in BIT;      -- s1
  r  :  in BIT;      -- r
  o  :  out BIT      -- o
   );
 END COMPONENT;

 COMPONENT lsduu
  PORT (
  vss : in BIT;      -- vss
  vdd : in BIT;      -- vdd
  phi1 : in BIT;     -- phi1
  phi2 : in BIT;     -- phi2
  D : in BIT;        -- D
  Q : inout BIT;     -- Q
  QB : inout BIT     -- QB
 );
 END COMPONENT;

-------------------------------------------------------------------------------
-- Internal signals
-------------------------------------------------------------------------------
 SIGNAL D,ib : bit;


BEGIN

 n0: nd PORT MAP (vss,vdd,i,ib);
 muxr1: muxr PORT MAP (vss,vdd,Q,Qb,ib,i,Rst,D);
```

```
l0: lsduu PORT MAP (vss,vdd,phi1,phi2,D,Q,Qb);

END structural;
```

Structural Description of the block requester module within the buffer manager:

```
--------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcbmbr.vst
-- Function:    Buffer chip buffer manager counter dff and module XOR
-- Author:      Jens Jakobsen
-- Date :       October 3 1993
--------------------------------------------------------------------------------

-- Entity Declaration

ENTITY bcbmbr IS
 PORT (
 vss,vdd: bit;                  -- Power supply
 phi1, phi2: bit;               -- Clock
 Rst : in BIT;                  -- Reset
 BR : in BIT;                   -- Block request
 size : in BIT;                 -- Empty or full
 BGR,BNGR : out BIT             -- Block granted or not granted
 );
END bcbmbr;


-- Architecture Declaration
ARCHITECTURE structural OF bcbmbr IS

 COMPONENT ldddd
  PORT (
  vss : in BIT;      -- vss
  vdd : in BIT;      -- vdd
  phi1 : in BIT;     -- phi1
  phi2 : in BIT;     -- phi2
  D : in BIT;        -- D
  Q : inout BIT;     -- Q
  QB : inout BIT     -- QB
  );
 END COMPONENT;

 COMPONENT nd
  PORT (
  vss : in BIT;                 -- vss
  vdd : in BIT;                 -- vdd
  i : in BIT;                   -- in
  o: out bit                    -- out
  );
 END COMPONENT;
```

```
COMPONENT o2nd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;

COMPONENT o3ns
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  i2 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;

COMPONENT mux
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- i0
  i1 : in BIT;                 -- i1
  s0 : in BIT;                 -- s0
  s1 : in BIT;                 -- s1
  o  : out BIT                 -- o
 );
END COMPONENT;

-------------------------------------------------------------------------------
-- Internal signals
-------------------------------------------------------------------------------

  SIGNAL BRQ1,BRQ1b,BRQ2,BRQ2b  : bit;

  SIGNAL BRQ3,BRQ3b  : bit;

  SIGNAL strobe,strobeb  : bit;

  SIGNAL sizeq1,sizeq1b,sized1 : bit;
BEGIN

-------------------------------------------------------------------------------
-- Handle metastability by using two flip-flops at BR input
-------------------------------------------------------------------------------

  l0: ldddd PORT MAP (vss,vdd,phi1,phi2,BR,BRQ1,BRQ1b);
  l1: ldddd PORT MAP (vss,vdd,phi1,phi2,BRQ1,BRQ2,BRQ2b);

-------------------------------------------------------------------------------
-- Generate strobe at positive edge of BRQ2
-------------------------------------------------------------------------------
```

```
 l3: ldddd PORT MAP (vss,vdd,phi1,phi2,BRQ2,BRQ3,BRQ3b);
 o2n0: o2nd PORT MAP (vss,vdd,BRQ2b,BRQ3,strobe);

-------------------------------------------------------------------------------
-- Latch current value of size
-------------------------------------------------------------------------------

 n0: nd PORT MAP (vss,vdd,strobe,strobeb);
 mux0: mux PORT MAP (vss,vdd,size,sizeQ1,strobeb,strobe,sizeD1);
 l4: ldddd PORT MAP (vss,vdd,phi1,phi2,sizeD1,sizeQ1,sizeQ1b);

-------------------------------------------------------------------------------
-- Generate output
-------------------------------------------------------------------------------
 o3n1: o3ns PORT MAP (vss,vdd,Rst,SizeQ1,BRQ3b,BGR);
 o3n2: o3ns PORT MAP (vss,vdd,Rst,SizeQ1b,BRQ3b,BNGR);


END structural;
```

Structural Description of the 5-bit up counters within the buffer manager:

```
-------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcbmc5u.vst
-- Function:    Buffer chip buffer manager counter dff and module XOR
-- Author:      Jens Jakobsen
-- Date :       October 3 1993
-------------------------------------------------------------------------------

-- Entity Declaration

ENTITY bcbmc5u IS
 PORT (
 vss,vdd: bit;                    -- Power supply
 phi1, phi2: bit;                 -- Clock
 Rst: bit;                       -- Reset
 upb : in BIT;                    -- Count up
 s: out bit_vector (4 downto 0)      -- Counter output
 );
END bcbmc5u;
-- Architecture Declaration
ARCHITECTURE structural OF bcbmc5u IS

 COMPONENT bcbmc5ub
  PORT (
  vss,vdd: bit;                    -- Power supply
  phi1, phi2: bit;                 -- Clock
  Rst: bit;                       -- Reset
  i: bit;                        -- Input
  S,Sb: inout bit                  -- Output
  );
```

```
END COMPONENT;


COMPONENT nd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i : in BIT;                  -- in
  o: out bit                   -- out
  );
END COMPONENT;

COMPONENT o2nd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;

COMPONENT o3nd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  i2 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;

COMPONENT o2dd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;
COMPONENT o3dd
  PORT (
  vss : in BIT;                -- vss
  vdd : in BIT;                -- vdd
  i0 : in BIT;                 -- in
  i1 : in BIT;                 -- in
  i2 : in BIT;                 -- in
  o: out bit                   -- out
  );
END COMPONENT;


-------------------------------------------------------------------------------
-- Internal signals
-------------------------------------------------------------------------------
```

```
  SIGNAL sb, i : bit_vector (4 downto 0);
  SIGNAL       m3a,m3b,m4a,m4b      : bit;


BEGIN


-------------------------------------------------------------------------------
-- Bit 0
-------------------------------------------------------------------------------
  n0: nd PORT MAP (vss,vdd,Upb,i(0));
  bcbmc5ub0: bcbmc5ub PORT MAP (vss,vdd,phi1,phi2,Rst,i(0),s(0),sb(0));

-------------------------------------------------------------------------------
-- Bit 1
-------------------------------------------------------------------------------
  o2n1: o2nd PORT MAP (vss,vdd,Upb,sb(0),i(1));
  bcbmc5ub1: bcbmc5ub PORT MAP (vss,vdd,phi1,phi2,Rst,i(1),s(1),sb(1));

-------------------------------------------------------------------------------
-- Bit 2
-------------------------------------------------------------------------------
  o3n2: o3nd PORT MAP (vss,vdd,Upb,sb(0),sb(1),i(2));
  bcbmc5ub2: bcbmc5ub PORT MAP (vss,vdd,phi1,phi2,Rst,i(2),s(2),sb(2));

-------------------------------------------------------------------------------
-- Bit 3
-------------------------------------------------------------------------------
  o23a: o2dd PORT MAP (vss,vdd,sb(2),sb(1),m3a);
  o23b: o2dd PORT MAP (vss,vdd,Upb,sb(0),m3b);
  o2n3: o2nd PORT MAP (vss,vdd,m3a,m3b,i(3));
  bcbmc5ub3: bcbmc5ub PORT MAP (vss,vdd,phi1,phi2,Rst,i(3),s(3),sb(3));

-------------------------------------------------------------------------------
-- Bit 4
-------------------------------------------------------------------------------
  o24: o2dd PORT MAP (vss,vdd,sb(3),sb(2),m4a);
  o34: o3dd PORT MAP (vss,vdd,Upb,sb(0),sb(1),m4b);
  o2n4: o2nd PORT MAP (vss,vdd,m4a,m4b,i(4));
  bcbmc5ub4: bcbmc5ub PORT MAP (vss,vdd,phi1,phi2,Rst,i(4),s(4),sb(4));

END structural;
```

Structural Description of one bit-slice of the logics within the 5-bit up counter in the buffer manager:

```
-------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcbmc5ub.vst
-- Function:    Buffer chip buffer manager counter dff and module XOR
-- Author:      Jens Jakobsen
-- Date :       October 3 1993
-------------------------------------------------------------------------------

-- Entity Declaration

ENTITY bcbmc5ub IS
 PORT (
 vss,vdd: bit;                  -- Power supply
 phi1, phi2: bit;               -- Clock
 Rst: bit;                    -- Reset
 i: bit;                     -- Input
 S,Sb: inout bit                -- Output
 );
END bcbmc5ub;


-- Architecture Declaration
ARCHITECTURE structural OF bcbmc5ub IS


 COMPONENT nd
  PORT (
  vss : in BIT;                  -- vss
  vdd : in BIT;                  -- vdd
  i : in BIT;                   -- in
  o: out bit                    -- out
  );
 END COMPONENT;

 COMPONENT nu
  PORT (
  vss : in BIT;                  -- vss
  vdd : in BIT;                  -- vdd
  i : in BIT;                   -- in
  o: out bit                    -- out
  );
 END COMPONENT;

 COMPONENT muxr
  PORT (
  vss : in BIT;      -- vss
  vdd : in BIT;      -- vdd
  i0 :  in BIT;      -- i0
  i1 :  in BIT;      -- i1
  s0 :  in BIT;      -- s0
  s1 :  in BIT;      -- s1
  r  :  in BIT;      -- r
```

```
  o  :  out BIT       -- o
   );
 END COMPONENT;

 COMPONENT lsduu
  PORT (
  vss : in BIT;       -- vss
  vdd : in BIT;       -- vdd
  phi1 : in BIT;      -- phi1
  phi2 : in BIT;      -- phi2
  D : in BIT;         -- D
  Q : inout BIT;      -- Q
  QB : inout BIT      -- QB
 );
 END COMPONENT;

-------------------------------------------------------------------------------
-- Internal signals
-------------------------------------------------------------------------------
 SIGNAL D,Q,ib : bit;


BEGIN

 n0: nd PORT MAP (vss,vdd,i,ib);
 muxr1: muxr PORT MAP (vss,vdd,Q,Sb,i,ib,Rst,D);
 l0: lsduu PORT MAP (vss,vdd,phi1,phi2,D,Q,Sb);
 n1: nu PORT MAP (vss,vdd,Sb,S);

END structural;
```

Appendix C. VHDL Structural Description of the Output Control

The VHDL structural description of the entire output control is shown below:

```
--------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcoc.vst
-- Function:    Buffer chip output control
-- Author:      Jens Jakobsen
-- Date :       September 29, 1993
-- Modified:    Eric CHU, 19th October, 1993.
--------------------------------------------------------------------------------
-- Entity Declaration

ENTITY bcoc IS
 PORT (
 vss,vdd: bit;                    -- Power supply
 phi1, phi2: bit;                 -- Clock
 Rst: bit;                       -- Reset
 OBA: bit_vector (4 downto 0);       -- Output block address
 OBG: bit;                       -- Output block granted
 OBNG: bit;                       -- Output block not granted
 OC: inout bit;                   -- Output conversion
 OBR: inout bit;                  -- Output block request
 RE: out bit;                    -- Read Enable
 RA: out bit_vector (6 downto 0)     -- Read address
 );
END bcoc;

-- Architecture Declaration

ARCHITECTURE structural_view OF bcoc IS

 COMPONENT bcocprc
  PORT (
  vss,vdd:bit;                    -- Power supply
  phi1,phi2:bit;                  -- Clocks
  Rst: BIT;                       -- Reset
  Q10,Q25,Q38,Q51: inout BIT         -- Decoded outputs
  );
 END COMPONENT;

 COMPONENT bcocc
  PORT (
  vss,vdd:bit;                    -- Power supply
  phi1,phi2:bit;                  -- Clocks
  Clear: BIT;                     -- Clear
  up,upb: BIT;                     -- Up signals
  A1,A0: out BIT                   -- Address outputs
  );
 END COMPONENT;
```

```
COMPONENT rrssd
 PORT (
 vss,vdd:bit;                -- Power supply
 r0 : in BIT;                -- Reset
 r1 : in BIT;                -- Reset
 s : in BIT;                 -- set
 q : inout BIT;              -- q
 qb : inout BIT              -- qb
 );
END COMPONENT;

COMPONENT rrsds
 PORT (
 vss,vdd:bit;                -- Power supply
 r0 : in BIT;                -- Reset
 r1 : in BIT;                -- Reset
 s : in BIT;                 -- set
 q : inout BIT;              -- q
 qb : inout BIT              -- qb
 );
END COMPONENT;

COMPONENT jdd
 port (
 vss,vdd: bit;               -- Power supply
 Ldb: bit;                   -- Latch enable
 D: bit;                     -- Data
 Q: inout bit;               -- Q output
 Qb: inout bit               -- Q* output
 );
END COMPONENT;

COMPONENT jds
 port (
 vss,vdd: bit;               -- Power supply
 Ldb: bit;                   -- Latch enable
 D: bit;                     -- Data
 Q: inout bit;               -- Q output
 Qb: inout bit               -- Q* output
 );
END COMPONENT;

COMPONENT jcdd
 port (vss,vdd,Clear,Ldb,D: BIT; Q,Qb: inout BIT);
END COMPONENT;

COMPONENT ldddd
 port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
END COMPONENT;

COMPONENT lddss
 port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
END COMPONENT;

COMPONENT lddsd
 port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
```

```
END COMPONENT;

COMPONENT lsdud
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
END COMPONENT;

COMPONENT nd
  port (vss,vdd,i: BIT; o: out BIT);
END COMPONENT;

COMPONENT nu
  port (vss,vdd,i: BIT; o: out BIT);
END COMPONENT;

COMPONENT o2nd
  port (vss,vdd,i0,i1: BIT; o: out BIT);
END COMPONENT;

COMPONENT o3nd
  port (vss,vdd,i0,i1,i3: BIT; o: out BIT);
END COMPONENT;

COMPONENT o2dd
  port (vss,vdd,i0,i1: BIT; o: out BIT);
END COMPONENT;

COMPONENT sink
  port (vss,vdd,i:bit);
END COMPONENT;


-- SIGNAL DECLARATION

-------------------------------------------------------------------------------
-- Delayed versions of input signals
-------------------------------------------------------------------------------
SIGNAL OBGD1,OBGD2,OBGD1b,OBGD2b : bit;    -- Block granted
SIGNAL OBNGD1,OBNGD2,OBNGD1b,OBNGD2b : bit; -- Block not granted
SIGNAL RstD1,RstD2,RstD1b,RstD2b : bit;    -- Reset

-------------------------------------------------------------------------------
-- Timing signals
-------------------------------------------------------------------------------

SIGNAL P0,P1,P2,P3:bit;          -- Output of Counter

SIGNAL     P3Q1,P3Q1b     : bit; -- Delayed versions of P3
SIGNAL     P3Q2,P3Q2b     : bit;
SIGNAL     P3Q3,P3Q3b     : bit;
SIGNAL     P3Q4,P3Q4b     : bit;
SIGNAL     P3Q5,P3Q5b     : bit;

SIGNAL     P3Q2a,P3Q2ab     : bit; -- More of Delayed P3
SIGNAL     P3Q3a,P3Q3ab     : bit;
SIGNAL     P3Q4a,P3Q4ab     : bit;
SIGNAL     P3Q5a,P3Q5ab     : bit;
```

```
SIGNAL      P01,P01b,P01D1   : bit; -- Combining P0 and P1
SIGNAL      P23,P23b,P23D1   :bit;  -- Combining P2 and P3

SIGNAL      OCD1          : bit; -- Output convert
SIGNAL      OCb           : bit;
SIGNAL      OCQ1,OCQ1b    : bit;
SIGNAL      OCQ2,OCQ2b    : bit;
SIGNAL      OCQ3,OCQ3b    : bit;
SIGNAL      OCQ4,OCQ4b    : bit;

-------------------------------------------------------------------------------
-- Local copies of block address and granted signals
-------------------------------------------------------------------------------

SIGNAL BlA0,BlA0b,BlA1,BlA1b: bit_vector (4 downto 0);-- Block address
SIGNAL OBGranted0,OBGranted0b,OBGranted1,OBGranted1b: bit;-- Block granted

-------------------------------------------------------------------------------
-- Generation of Block request
-------------------------------------------------------------------------------

SIGNAL      OB,OBR1,OBR1b,OBRb          : bit;

-------------------------------------------------------------------------------
-- Read enable signals
-------------------------------------------------------------------------------

SIGNAL  RED1,RED1b,RED2          : bit;


-- Structural Description

BEGIN


-------------------------------------------------------------------------------
-- Handle metastability by using two flip flops on asynchronous control inputs
-------------------------------------------------------------------------------

m0: ldddd PORT MAP (vss,vdd,phi1,phi2,OBG,OBGD1,OBGD1b);
m1: ldddd PORT MAP (vss,vdd,phi1,phi2,OBGD1,OBGD2,OBGD2b);

m2: ldddd PORT MAP (vss,vdd,phi1,phi2,OBNG,OBNGD1,OBNGD1b);
m3: ldddd PORT MAP (vss,vdd,phi1,phi2,OBNGD1,OBNGD2,OBNGD2b);

m4: ldddd PORT MAP (vss,vdd,phi1,phi2,Rst,RstD1,RstD1b);
m5: lsdud PORT MAP (vss,vdd,phi1,phi2,RstD1,RstD2,RstD2b);

-------------------------------------------------------------------------------
-- Generate Output block request
-------------------------------------------------------------------------------

o20: o2dd PORT MAP (vss,vdd,OBGD2,OBNGD2,OB);
rs0: rrsds PORT MAP (vss,vdd,RstD2,OB,P0,OBR1,OBR1b);
sink1: sink PORT MAP (vss,vdd,OBR1);
n1: nu PORT MAP (vss,vdd,OBR1b,OBR);
```

```
--------------------------------------------------------------------------------
-- Make local copy of OBG and Block address
--------------------------------------------------------------------------------

 n2: nu PORT MAP (vss,vdd,OBR,OBRb);

 j0: jdd PORT MAP (vss,vdd,OBRb,OBA(0),BlA0(0),BlA0b(0));
 j1: jdd PORT MAP (vss,vdd,OBRb,OBA(1),BlA0(1),BlA0b(1));
 j2: jdd PORT MAP (vss,vdd,OBRb,OBA(2),BlA0(2),BlA0b(2));
 j3: jdd PORT MAP (vss,vdd,OBRb,OBA(3),BlA0(3),BlA0b(3));
 j4: jdd PORT MAP (vss,vdd,OBRb,OBA(4),BlA0(4),BlA0b(4));
 j5: jdd PORT MAP (vss,vdd,OBRb,OBG,OBGranted0,OBGranted0b);
 sink3: sink PORT MAP (vss,vdd,OBGranted0b);
 sink10: sink PORT MAP (vss,vdd,BlA0b(0));
 sink11: sink PORT MAP (vss,vdd,BlA0b(1));
 sink12: sink PORT MAP (vss,vdd,BlA0b(2));
 sink13: sink PORT MAP (vss,vdd,BlA0b(3));
 sink14: sink PORT MAP (vss,vdd,BlA0b(4));


--------------------------------------------------------------------------------
-- Counter
--------------------------------------------------------------------------------
 bcocprc: bcocprc PORT MAP (vss,vdd,phi1,phi2,RstD2,P0,P1,P2,P3);

--------------------------------------------------------------------------------
-- Decoding Output Convert signals
--------------------------------------------------------------------------------
 o2n1: o2nd PORT MAP (vss,vdd,P0,P1,P01D1);
 l0: ldddd PORT MAP (vss,vdd,phi1,phi2,P01D1,P01,P01b);

 o2n2: o2nd PORT MAP (vss,vdd,P2,P3,P23D1);
 l1: ldddd PORT MAP (vss,vdd,phi1,phi2,P23D1,P23,P23b);

 o21: o2dd PORT MAP (vss,vdd,P01b,P23b,OCD1);
 l2: lsdud PORT MAP (vss,vdd,phi1,phi2,OCD1,OC,OCb);

 l3: ldddd PORT MAP (vss,vdd,phi1,phi2,OC,OCQ1,OCQ1b);
 l4: lddsd PORT MAP (vss,vdd,phi1,phi2,OCQ1,OCQ2,OCQ2b);
 l5: ldddd PORT MAP (vss,vdd,phi1,phi2,OCQ2,OCQ3,OCQ3b);
 l51: ldddd PORT MAP (vss,vdd,phi1,phi2,OCQ3,OCQ4,OCQ4b);

 l6: ldddd PORT MAP (vss,vdd,phi1,phi2,P3,P3Q1,P3Q1b);
 l7: ldddd PORT MAP (vss,vdd,phi1,phi2,P3Q1,P3Q2,P3Q2b);
 l8: ldddd PORT MAP (vss,vdd,phi1,phi2,P3Q2,P3Q3,P3Q3b);
 l9: ldddd PORT MAP (vss,vdd,phi1,phi2,P3Q3,P3Q4,P3Q4b);
 l10: lsdud PORT MAP (vss,vdd,phi1,phi2,P3Q4b,P3Q5b,P3Q5);

 l7a: ldddd PORT MAP (vss,vdd,phi1,phi2,P3Q1,P3Q2a,P3Q2ab);
 l8a: ldddd PORT MAP (vss,vdd,phi1,phi2,P3Q2a,P3Q3a,P3Q3ab);
 l9a: ldddd PORT MAP (vss,vdd,phi1,phi2,P3Q3a,P3Q4a,P3Q4ab);
 l10a: ldddd PORT MAP (vss,vdd,phi1,phi2,P3Q4ab,P3Q5ab,P3Q5a);


--------------------------------------------------------------------------------
-- Current copy of OBG
--------------------------------------------------------------------------------
```

```
  j11: jcdd PORT MAP (vss,vdd,RstD2,P3Q5ab,OBGranted0,OBGranted1,OBGranted1b);
  sink4: sink PORT MAP (vss,vdd,OBGranted1);

-------------------------------------------------------------------------------
-- Generate read enable
-------------------------------------------------------------------------------
  o2n4: o2nd PORT MAP (vss,vdd,OCQ4b,OBGranted1b,RED2);
--  rs1: rrssd PORT MAP (vss,vdd,OCQ2,RstD2,RED2,RED1,RED1b);
  rs1: rrsds PORT MAP (vss,vdd,OCQ2,RstD2,RED2,RED1,RED1b);
  sink0: sink PORT MAP (vss,vdd,RED1);
  n3: nu PORT MAP (vss,vdd,RED1b,RE);

-------------------------------------------------------------------------------
-- Generation of read address
-------------------------------------------------------------------------------
  j6: jds PORT MAP (vss,vdd,P3Q5b,BlA0(0),BlA1(0),BlA1b(0));
  j7: jds PORT MAP (vss,vdd,P3Q5b,BlA0(1),BlA1(1),BlA1b(1));
  j8: jds PORT MAP (vss,vdd,P3Q5b,BlA0(2),BlA1(2),BlA1b(2));
  j9: jds PORT MAP (vss,vdd,P3Q5b,BlA0(3),BlA1(3),BlA1b(3));
  j10: jds PORT MAP (vss,vdd,P3Q5b,BlA0(4),BlA1(4),BlA1b(4));
  sink5: sink PORT MAP (vss,vdd,BlA1(0));
  sink6: sink PORT MAP (vss,vdd,BlA1(1));
  sink7: sink PORT MAP (vss,vdd,BlA1(2));
  sink8: sink PORT MAP (vss,vdd,BlA1(3));
  sink9: sink PORT MAP (vss,vdd,BlA1(4));


  n4: nu PORT MAP (vss,vdd,BlA1b(0),RA(2));
  n5: nu PORT MAP (vss,vdd,BlA1b(1),RA(3));
  n6: nu PORT MAP (vss,vdd,BlA1b(2),RA(4));
  n7: nu PORT MAP (vss,vdd,BlA1b(3),RA(5));
  n8: nu PORT MAP (vss,vdd,BlA1b(4),RA(6));


  bcocc: bcocc PORT MAP (vss,vdd,phi1,phi2,P3Q4a,OCQ2,OCQ2b,RA(1),RA(0));

end structural_view;
```

Structural Description of the Pseudo Random Counter module within the output control:

```
--------------------------------------------------------------------------------
-- VHDL structorial description of module
-- File name:   bcocprc.vst
-- Function:    Buffer Chip output control pseudo random counter
-- Author:      Jens Jakobsen
-- Date :       September 29, 1993
-- Modified:    Eric CHU on 12th October, 1993.
--------------------------------------------------------------------------------


-- Entity Declaration

ENTITY bcocprc IS
 PORT (
 vss,vdd:bit;                    -- Power supply
 phi1,phi2:bit;                  -- Clocks
 Rst: BIT;                       -- Reset
 Q10,Q25,Q38,Q51: inout BIT      -- Decoded outputs
 );
END bcocprc;


-- Architecture Declaration

ARCHITECTURE structural_view OF bcocprc IS

 COMPONENT bcocprcd
  PORT (
  vss,vdd:bit;                   -- power supply
  phi1,phi2:bit;                 -- Clock
  i0,i1,i2,i3,i4,i5: BIT;        -- Decoder inputs
  O,Ob: inout BIT                -- Decoder outputs
  );
 END COMPONENT;

 COMPONENT nd
  port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT ns
  port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o2nd
  port (vss,vdd,i0,i1: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT ldddd
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lddsd
```

```
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
END COMPONENT;

COMPONENT lsdud
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
END COMPONENT;

COMPONENT lsduu
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
END COMPONENT;

COMPONENT muxr
  PORT (
  vss : in BIT;      -- vss
  vdd : in BIT;      -- vdd
  i0 :  in BIT;            -- i0
  i1 :  in BIT;            -- i1
  s0 :  in BIT;            -- s0
  s1 :  in BIT;            -- s1
  r  :  in BIT;            -- r
  o  :  out BIT            -- o
 );
END COMPONENT;


-- Signal Declaration

 SIGNAL   S,Sb,DS: bit_vector (5 downto 0);

 SIGNAL   RESET,RESETb,RESETD1,RESETD1b: BIT;

-------------------------------------------------------------------------------
-- Decoder outputs
-------------------------------------------------------------------------------

 SIGNAL   Q10b,Q25b,Q38b,Q51b : BIT;

 SIGNAL   S0b1,S5b1,s52,s53 : BIT;

-- Structural Description

BEGIN

-- Pseudo Random Counter

-- 6th Bit
  invd1: nd  PORT MAP (vss,vdd,S(0),S0b1);
  invd2: nd  PORT MAP (vss,vdd,S(5),S5b1);
  muxr: muxr PORT MAP (vss,vdd,S(0),S0b1,S5b1,S(5),RESET,DS(5));--  Xnor
  dffu1:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS(5),S(5),Sb(5));

-- 5th Bit
  nor2d5:  o2nd  PORT MAP (vss,vdd,RESET,Sb(5),DS(4));
  dffu2:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS(4),S(4),Sb(4));

-- 4th Bit
  nor2d6:  o2nd  PORT MAP (vss,vdd,RESET,Sb(4),DS(3));
```

```
  dffu3:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS(3),S(3),Sb(3));

-- 3rd Bit
  nor2d7:  o2nd  PORT MAP (vss,vdd,RESET,Sb(3),DS(2));
  dffu4:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS(2),S(2),Sb(2));

-- 2nd Bit
  nor2d8:  o2nd  PORT MAP (vss,vdd,RESET,Sb(2),DS(1));
  dffu5:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS(1),S(1),Sb(1));

-- 1st Bit
  nor2d9:  o2nd  PORT MAP (vss,vdd,RESET,Sb(1),DS(0));
  dffu6:  lsduu  PORT MAP (vss,vdd,phi1,phi2,DS(0),S(0),Sb(0));

-- End of Pseudo Random Counter


 bcocprcd10: bcocprcd          -- Decode 10
 PORT MAP (vss,vdd,phi1,phi2,sb(5),sb(4),s(3),s(2),sb(1),s(0),Q10,Q10b);

 bcocprcd25: bcocprcd          -- Decode 25
 PORT MAP (vss,vdd,phi1,phi2,s52,sb(4),sb(3),s(2),sb(1),s(0),Q25,Q25b);
 invd3: nd PORT MAP (vss,vdd,sb(5),s52);

 bcocprcd38: bcocprcd          -- Decode 38
 PORT MAP (vss,vdd,phi1,phi2,s53,s(4),s(3),sb(2),s(1),sb(0),Q38,Q38b);
 invd4: nd PORT MAP (vss,vdd,sb(5),s53);

 bcocprcd51: bcocprcd          -- Decode 51
 PORT MAP (vss,vdd,phi1,phi2,sb(5),sb(4),s(3),s(2),sb(1),sb(0),Q51,Q51b);


-- Internal Header End Decode

  o2nxx: o2nd  PORT MAP (vss,vdd,Q51,Rst,RESETD1b);
  nxx:    ns  PORT MAP (vss,vdd,RESETD1b,RESETD1);
  lxx:  lsdud  PORT MAP (vss,vdd,phi1,phi2,RESETD1,RESET,RESETb);


end structural_view;
```

Structural Description of the Pseudo Random Counter Decoding module within the output control:

```
-------------------------------------------------------------------------------
-- VHDL structorial description of module
-- File name:   bcocprcd.vst
-- Function:    Buffer Chip output control pseudo random counter decoder
-- Author:      Jens Jakobsen
-- Date :       September 29, 1993
-------------------------------------------------------------------------------


-- Entity Declaration

ENTITY bcocprcd IS
 PORT (
 vss,vdd:bit;                    -- power supply
 phi1,phi2:bit;                  -- Clock
 i0,i1,i2,i3,i4,i5: BIT;           -- Decoder inputs
 O,Ob: inout BIT                   -- Decoder outputs
 );
END bcocprcd;


-- Architecture Declaration

ARCHITECTURE structural_view OF bcocprcd IS

 COMPONENT o2nd
  port (vss,vdd,i0,i1: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o3nd
  port (vss,vdd,i0,i1,i2: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT ldddd
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lddss
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;

 COMPONENT lsduu   -- Currently not used
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;


-- Signal Declaration

 SIGNAL  out0: BIT;
 SIGNAL  out1: BIT;
 SIGNAL  out2: BIT;
 SIGNAL  Qout0: BIT;
```

```
 SIGNAL  Qout0b: BIT;
 SIGNAL  Qout1: BIT;
 SIGNAL  Qout1b: BIT;


-- Structural Description

BEGIN


  o3n0:  o3nd  PORT MAP (vss,vdd,i0,i1,i2,out0);
  l0:  ldddd  PORT MAP (vss,vdd,phi1,phi2,out0,Qout0,Qout0b);
  o3n1:  o3nd  PORT MAP (vss,vdd,i3,i4,i5,out1);
  l1:  ldddd  PORT MAP (vss,vdd,phi1,phi2,out1,Qout1,Qout1b);
  o2n0:  o2nd  PORT MAP (vss,vdd,Qout0b,Qout1b,out2);
  l2:  lddss  PORT MAP (vss,vdd,phi1,phi2,out2,o,ob);


end structural_view;
```

Structural Description of the 2-Bit Counter within the output control:

```
--------------------------------------------------------------------------------
-- VHDL structural description of module
-- File name:   bcocc.vst
-- Function:    Counter for Lower 2-Bit read Address Generation
-- Author:      Eric Chu
-- Date :       27th September, 1993
-- Revised:     Jens Jekobsen, September 30, 1993
--------------------------------------------------------------------------------


-- Entity Declaration

ENTITY bcocc IS
PORT (vss,vdd,phi1,phi2,Clear,up,upb: BIT; A1,A0: out BIT);
END bcocc;


-- Architecture Declaration

ARCHITECTURE structural_view OF bcocc IS

 COMPONENT nu
   port (vss,vdd,i: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o2nd
   port (vss,vdd,i0,i1: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT o3nd
```

```
  port (vss,vdd,i0,i1,i2: BIT; o: out BIT);
 END COMPONENT;

 COMPONENT lddss
  port (vss,vdd,phi1,phi2,D: BIT; Q,QB: out BIT);
 END COMPONENT;


-- Signal Declaration

 SIGNAL DA,DB: BIT;
 SIGNAL QB,QBb,QA,QAb: BIT;
 SIGNAL out1,out2,out3,out4: BIT;


-- Structural Description

BEGIN

 lddss1: lddss PORT MAP (vss,vdd,phi1,phi2,DA,QA,QAb);
 lddss2: lddss PORT MAP (vss,vdd,phi1,phi2,DB,QB,QBb);

-- Input to Dff A:

 nor2d1: o2nd PORT MAP (vss,vdd,Up,QA,out1);
 nor2d2: o2nd PORT MAP (vss,vdd,Upb,QAb,out2);
 nor3d1: o3nd PORT MAP (vss,vdd,out1,out2,Clear,DA);

-- Input to Dff B:

 nor2d3: o2nd PORT MAP (vss,vdd,Upb,QAb,out3);
 nor2d4: o2nd PORT MAP (vss,vdd,out3,QB,out4);
 nor2d5: o2nd PORT MAP (vss,vdd,out4,Clear,DB);

-- Buffering Write Addresses

 invu1:  nu   PORT MAP (vss,vdd,QBb,A1);
 invu2:  nu   PORT MAP (vss,vdd,QAb,A0);

end structural_view;
```

Appendix D. Published Paper

The paper "Comparison of GaAs Static Logic Families" which was published in the 11th NorChip seminar held in Trondheim, Norway on November 1993 is included in this appendix.

COMPARISON OF GaAs STATIC LOGIC FAMILIES

Eric Chu, Jens Jakobsen

Jydsk Telefon R&D
30 Sletvej, 8310 Tranbjerg J, Denmark
Phone: + 45 89 45 45 45
Fax: + 45 86 29 90 68
E-MAIL: eric@lab.jt.dk, jj@lab.jt.dk

ABSTRACT

Optimisation of logic families plays an important role in VLSI designs. In this paper we analyse DCFL for speed, power and noise margins. Three other logic families based on DCFL are discussed. SDCFL, SBFL, UBFL as well as DCFL are then optimised and compared on the basis of speed, power, noise margins and the noise induced in the power supply. It is concluded that DCFL has good performance at low fan-ins and fan-outs. SDCFL can be used for larger fan-ins. For large fan-outs SDCFL, SBFL or UBFL have to be used according to the noise margin and noise induction requirements.

1. INTRODUCTION

Optimisation of logic families plays an important role in VLSI designs as it determines the size, speed and power consumption of the final product.

In this paper we consider the optimisation of a number of logic families for Gallium Arsenide VLSI. Initially we analyse Direct Coupled FET Logic (DCFL) regarding speed, power, and noise margins. DCFL has the advantage of being simple and having good performance at low fan-in and fan-out. For large fan-in the noise margin degrades rapidly, and the driving capability is poor at large fan-out. In these situations other logic families have to be used.

From DCFL a series of compatible logic families are derived. These include Source-follower Direct Coupled FET Logic (SDCFL) [DAVENPORT], Super Buffer FET Logic (SBFL) [NAKAMURA] and Ultra Buffer FET Logic (UBFL). The logic families are analysed and a comparison is done on the basis of speed, power, noise margins and induced noise in the power supplies.

2. DIRECT COUPLED FET LOGIC (DCFL)

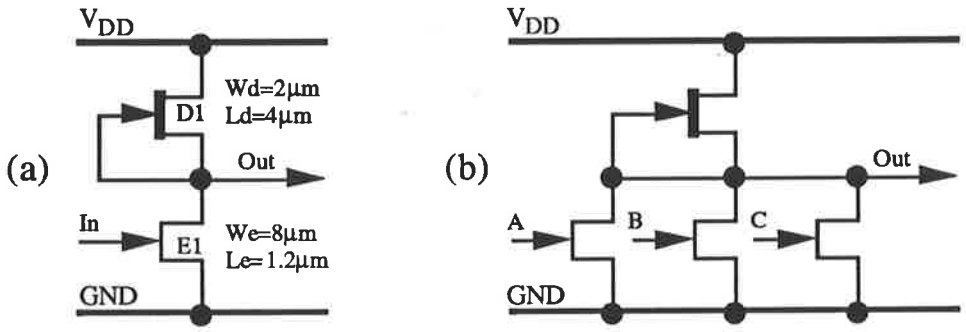


Figure 1: (a) DCFL inverter and (b) DCFL 3-input nor gate

Figure 1a shows a DCFL inverter. It consists of a pull-up depletion mode MESFET (DFET) and a pull-down enhancement mode MESFET (EFET). The operation is similar to the NMOS inverter except the output high level (V_{OH}) of DCFL is limited by the gate-source Schottky diode of the following stage. In the following context we assume the clamping voltage of this Schottky diode to be 0.6V.

The supply voltage should be chosen so that the DFET always operates in saturation. This optimises speed and ensures that the current drawn from V_{dd} is constant. Thus minimal current noise is induced in the power supply. A 3-input nor gate can be constructed by connecting EFETs in parallel as shown in Figure 1b.

The operation of the MESFETs can be modelled by the Curtice model [CURTICE] which is given by the expression below:

$$I_{ds} = \beta \cdot \left(\frac{W}{L}\right) \cdot \left(V_{gs} - V_t\right)^2 \cdot \left(1 + \lambda \cdot V_{ds}\right) \cdot \tanh\left(\alpha \cdot V_{ds}\right)$$

In order to analyse the DCFL inverter some simplifications to the model are practical. In the following analysis, we will assume that $\tanh(\alpha \cdot V_{ds}) = 1$ for $\alpha \cdot V_{ds} \geq 1$ (saturation) and $\tanh(\alpha \cdot V_{ds}) = \alpha \cdot V_{ds}$ for $\alpha \cdot V_{ds} < 1$ (linear region). Furthermore we assume λ to be 0.

The pull-up DFET will always operate in saturation mode. Thus its drain-source current can be written as:

$$I_{dsd} = \beta_d \cdot \left(\frac{W_d}{L_d}\right) \cdot \left(V_{td}\right)^2$$

where the subscript "d" denotes DFET. The EFET will operate in 3 different regions: cut off, linear and saturation in which the drain-source currents can be written as:

$$I_{dse} = \begin{cases} 0 & \text{for } V_{gs} < V_{te} \\ \beta_e \cdot \dfrac{W_e}{L_e} \cdot \left(V_{gs} - V_{te}\right)^2 & \text{for } V_{gs} > V_{te};\ V_{ds} \geq \dfrac{1}{\alpha_e} \\ \alpha_e \cdot \beta_e \cdot \dfrac{W_e}{L_e} \cdot \left(V_{gs} - V_{te}\right)^2 \cdot V_{ds} & \text{for } V_{gs} > V_{te};\ V_{ds} < \dfrac{1}{\alpha_e} \end{cases}$$

where the subscript "e" denotes EFET.

2.1 Inverter Threshold

We define the inverter threshold V_{inv} as the input voltage of the inverter where the saturation current of the EFET equals that of the DFET. V_{inv} can be written as:

$$V_{inv} = V_{te} - \frac{V_{td}}{\sqrt{x}} \quad \text{where } x = \frac{\beta_e \cdot W_e \cdot L_d}{\beta_d \cdot W_d \cdot L_e} \text{ (pull - down / pull - up ratio)}$$

2.2 Inverter Characteristics

Given V_{inv} the inverter transfer characteristics can be determined as:

$$V_{out} = \begin{cases} V_{OH} & \text{for } V_{in} \leq V_{inv} \\ \dfrac{V_{td}^2}{x \cdot \alpha_e \left(V_{in} - V_{te}\right)^2} & \text{for } V_{in} > V_{inv} \end{cases}$$

As can be seen, the above expression does not model the behaviour of the inverter very well when V_{in} is in the vicinity of V_{inv}. The characteristics of a DCFL inverter simulated with foundry supplied model at 125°C is shown in Figure 2.

The input voltage is swept from 0V to 0.6V. Initially the EFET is turned off while the DFET pulls the output up to the clamping voltage of the gate-source Schottky diode of the following stage.

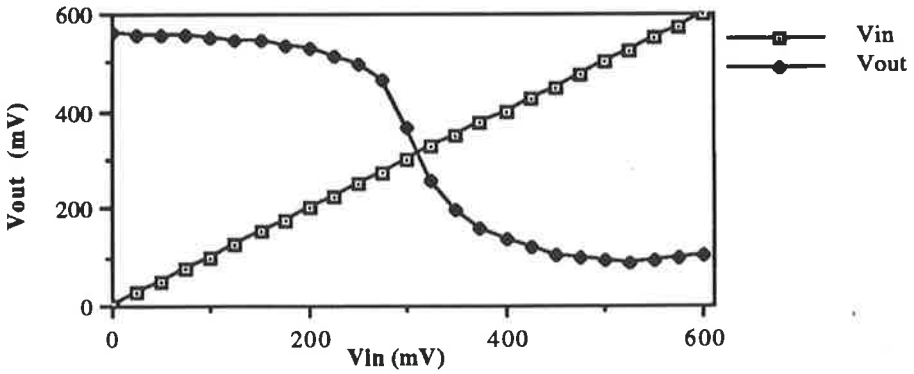


Figure 2: DC characteristics of DCFL inverter with 1 fan-out

When the input voltage reaches V_{te}, the EFET enters the saturation region. As long as the current in the EFET is lower than that of the DFET, the output voltage stays high. When the current in the EFET approaches to that of the DFET, the output voltage reduces until the EFET enters the linear region. From this point onwards the output level decreases. The slight rise in the output at 0.6V is due to gate-drain Schottky diode conduction.

2.3 Inverter Noise Margin

From the transfer characteristics we can calculate the intrinsic noise margins of a DCFL inverter and nor gates.

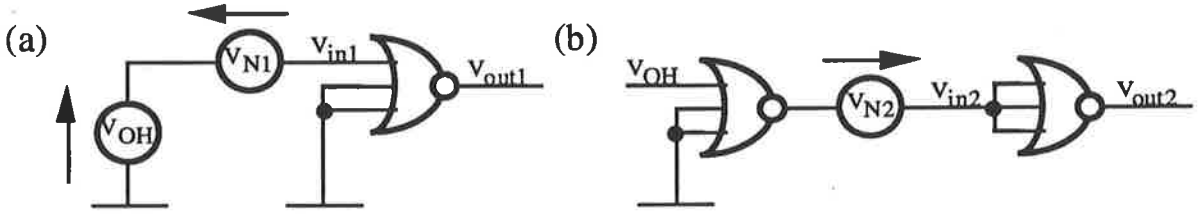


Figure 3: Circuits for determining (a) the intrinsic high and (b) the intrinsic low noise margin

For a logic high level, the worst case is when only one input of the driven nor gate is high. Thus the intrinsic high noise margin can be written as:

$$V_{NH} = V_{OH} - V_{inv} = V_{OH} - V_{te} + \frac{V_{td}}{\sqrt{x}}$$

For a logic low level (V_{OL}), the worst case is when only one input of the driving gate is high while all inputs of the driven gate are low. Thus V_{inv} is modified by the fan-in (F_{in}) of the driven gate. The intrinsic low noise margin can be written as:

$$V_{NL} = V_{inv} - V_{OL} = V_{te} - \frac{V_{td}}{\sqrt{F_{in} \cdot x}} - \frac{V_{td}^{\,2}}{x \cdot \alpha_e (V_{OH} - V_{te})^2}$$

The intrinsic noise margins are plotted against x in Figure 4. It can be seen that the low noise margin degrades rapidly for a large fan-in and x should be large for a good high noise margin.

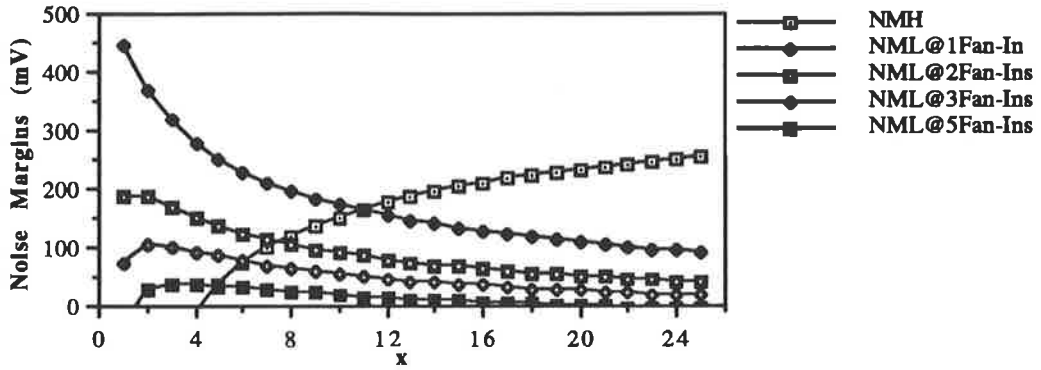


Figure 4: Intrinsic noise margins of DCFL gates as a function of x at 125°C

2.4 Inverter Delay

The delay of a DCFL inverter is assumed to be the average of the rise and fall times:

$$\text{Delay} = \frac{\tau_r + \tau_f}{2} = \frac{C_L}{2}\left(\frac{\Delta V}{I_{dsd}} + R_{ON}\right)$$

where C_L is the load capacitance, ΔV is the voltage swing and R_{ON} is the "on" resistance of the EFET.

If we assume C_L is directly proportional to the width of the EFET then it follows that the fall-time is constant. The rise-time, however, is proportional to C_L/I_{dsd} to a first order approximation. Thus the rise-time is directly proportional to x.

The choice of x is therefore a trade off between the noise margin and the speed.

2.5 Power-Delay Product

Once x is determined the size of the devices should be chosen accordingly. The power delay product of DCFL gates can be written as:

$$\text{Power - Delay Product} = \frac{V_{dd} \cdot C_L}{2} \cdot \left(\Delta V + R_{ON} \cdot I_{dsd}\right)$$

where C_L is the sum of the driver, fan-out and the interconnect capacitances:

$$C_L = C_{gdd} + F_{out} \cdot C_{gse} + C_I$$

where C_{gdd} denotes the gate-drain capacitance of the DFET, F_{out} is the number of fan-outs, C_{gse} is the gate-source capacitance of the EFET and C_I is the interconnect capacitance. For large devices C_{gdd} is small compared to C_{gse}. For short EFETs, however, the gate length of the DFET has to be scaled in order to achieve a weak pull-up. In such a case C_{gdd} becomes comparable to, or even exceeds C_{gse}. The power delay product of a DCFL inverter is shown in Figure 5 with one fan-out and constant x. It can be seen that the power delay product is minimum when $W_e \approx 8\mu m$. Minimum delay, however, can be obtained using large devices.

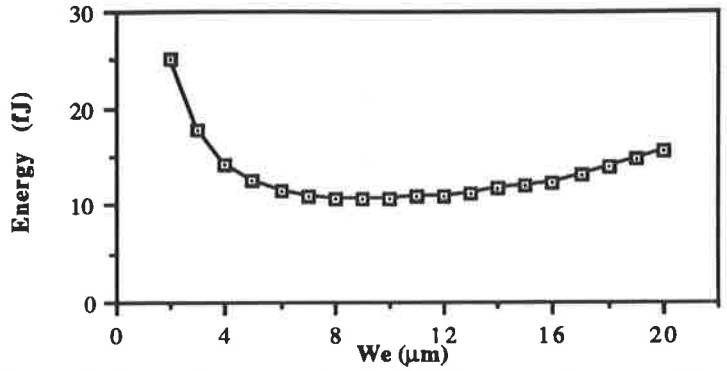


Figure 5: Power delay product of DCFL inverters as a function of W_e. C_{gdd} is minimum when $W_e = 16\mu m$.

3. SOURCE-FOLLOWER DIRECT COUPLED FET LOGIC (SDCFL)

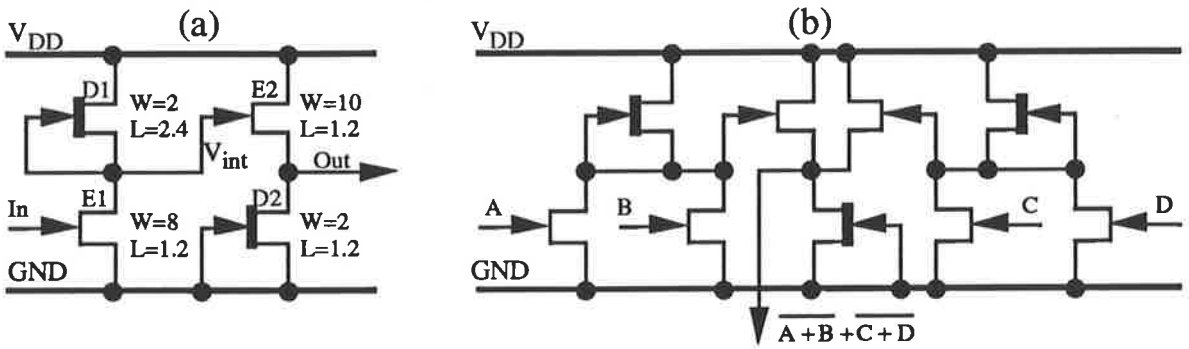


Figure 6: (a) SDCFL inverter and (b) SDCFL O-A-I structure

Both the noise margin and the driving capability of DCFL can be improved by appending a source follower at its output. An SDCFL inverter is shown in Figure 6(a).

The driving stage of an SDCFL inverter (E1, D1) operates identically to a DCFL inverter except the output high level (V_{OH}) is not clamped to 0.6V. The buffer stage (E2, D2) reduces the swing of the DCFL output. Thus SDCFL gates can be designed with 0V for V_{OL}. Similar to DCFL gates, V_{OH} of SDCFL is also clamped to 0.6V by the following fan-out gates which loads the output. SDCFL gates has good noise margins due to the improved output low level.

When the input is high, the driving stage pulls down V_{int} and E2 is turned off. Whereas a low input causes the DCFL stage to pull up V_{int} and turn on E2. As a result, there is a large difference in the current drawn from V_{dd} between the output high and the output low states.

An additional advantage of SDCFL is that or-and-invert (O-A-I) functions can be implemented as shown in Figure 6(b).

4. SUPER BUFFER FET LOGIC (SBFL)

Similar to SDCFL, SBFL improves the performance of DCFL by appending a buffer (E2, E3) at the output. A SBFL inverter is shown in Figure 7(a). When compared with SDCFL, SBFL has the advantage that the pull-down EFET in the buffer (E2) is only active during

output low. Thus the pull-down EFET can be much stronger than that of SDCFL. The disadvantage of SBFL lies in a high fan-in load compared to both SDCFL and DCFL.

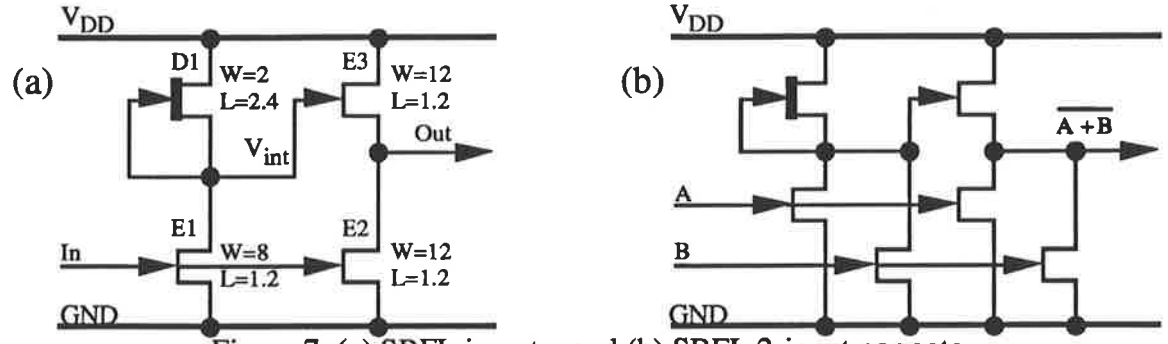


Figure 7: (a) SBFL inverter and (b) SBFL 2-input nor gate

As for SDCFL, SBFL has a large difference in the current drawn from V_{dd} at the output high and output low states. Moreover when the input has a low to high transition, both E2 and E3 are turned on simultaneously until V_{int} is pulled down which in turn draws a large current spike from the power supply [LONG&BUTNER].

A 2-input nor gate is shown in Figure 7(b). Parallel transistors are required in both the driver and the buffer stages.

5. ULTRA BUFFER FET LOGIC (UBFL)

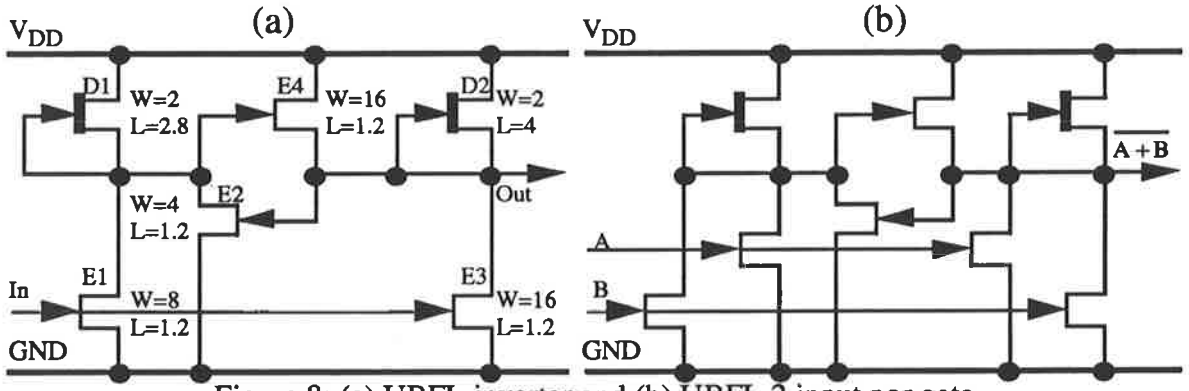


Figure 8: (a) UBFL inverter and (b) UBFL 2 input nor gate

The UBFL inverter as shown in Figure 8(a) is a further improvement to SBFL. The ultra buffer has both an active pull-up EFET (E4) and a passive pull-up DFET (D2). E4 is only active during the output low to high transition, otherwise it is turned off. This is achieved by using a two input nor gate (formed by E1, E2 and D1) as the driver. The buffer output is fed back to one of the driver inputs. When the input is high, both E1 and E3 are turned on while both the buffer and the driver outputs are low. As a result, both E2 and E4 are turned off.

When the input goes low the output of the driver goes high. This turns E4 on which pulls the output high. When the output is high E2 turns on which pulls the driver output low and turns E4 off. The output is held high by D2.

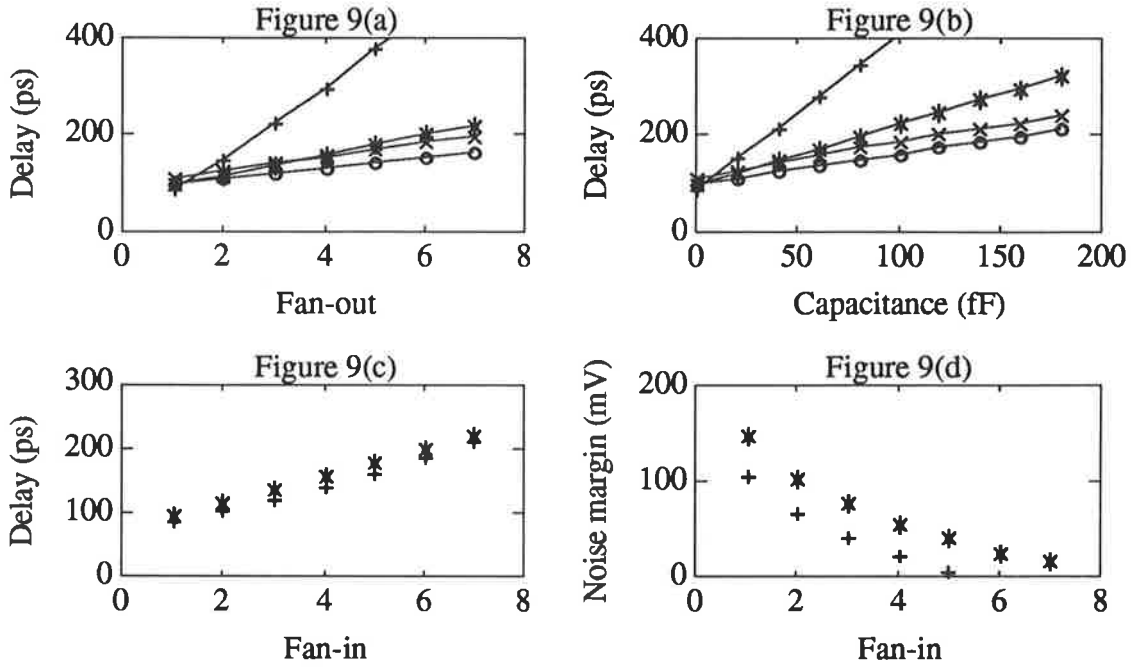
It can be seen that UBFL does not have the current spike as SBFL does. Moreover the static current drawn from V_{dd} in both the output high and the output low state is determined by D1 and D2. Thus UBFL has low power consumption and only minimal noise is induced in V_{dd}. Furthermore, the current within the gate will always have a direct path to ground.

UBFL has similar noise margins as DCFL since they have the same output levels.

6. COMPARISON BETWEEN DCFL, SDCFL, SBFL AND UBFL

The above logic families have been optimised and compared in terms of speed, power, noise margins and noise induced in the power supply. The optimisation is done for the TCS SAGA 0.8μm process with 1.5V V_{dd} at 125°C.

For SBFL and UBFL, only inverters are considered whereas for DCFL and SDCFL we have also considered nor gates. All delay simulations are carried out using a 7-stage ring oscillator. For capacitance and fan-out measurements, we include extra capacitance and DCFL inverters respectively at the output of each gate in the ring oscillator. The noise margin is determined by the Maximum Square Method [HILL]. Some of the simulated results are shown in Figure 9. A performance summary is given in table I.



Key: "+" denotes DCFL, "*" denotes SDCFL, "o" denotes SBFL and "x" denotes UBFL

Figure 9: (a) Effect of fan-out on speed of inverters (b) Effect of capacitive loads on speed of inverters (c) Effect of fan-in on speed at 1 fan-out (d) Effect of fan-in on noise margins

Table I: Performance summary of various static logic families

	DCFL	SDCFL	SBFL	UBFL
Dissipation (μW)	110	360	409	283
Noise Margin (mV)	107	142	192	87
V_{inv} (mV)	311	331	331	277
Inverter Delay (ps)	88	97	97	110
Delay/Fan-Out (ps)	75	22	11	15
Delay/Fan-In (ps)	21	20	–	–
Delay/20fF Load (ps)	64	25	13	15
Static Current Balance (% deviation from mean)	7	42	49	10
Number of transistors for an n-Input Nor Gate	n+1	n+3	2n+2	2n+4
Power-Delay Product at 1 Fan-Out (fJ)	10	35	40	31

7. CONCLUSION

Various GaAs static families have been analysed for speed, power dissipation, noise margins and noise induced in the power supply.

Analytical results on DCFL show that the pull-down/pull-up ratio has to be optimised as a compromise between speed and the noise margins. The device size can be optimised for either maximum speed or maximum power delay product.

DCFL, SDCFL, SBFL and UBFL have been optimised and compared. It can be seen that DCFL is only good for low fan-ins and fan-outs. It fails to operate correctly at a large fan-in and is slow at large fan-out.

SDCFL has good noise margins, high driving capabilities and can operate with a larger fan-in. But it has a larger power consumption and induces noise in the power supply. Thus SDCFL is a good substitution for DCFL in critical paths where extra speed and/or fan-ins are required.

SBFL has even better noise margins and driving capability than SDCFL. However it has a large fan-in load and induces even more noise into the power supply than SDCFL. This makes its use limited.

In cases like wide buses where a large number of signals shift at the same time, noisy gates cannot be tolerated. UBFL should be used since it induces only a small amount of noise into the power supply while preserving good driving capability. However, its noise margin is as low as DCFL and the input load is as large as that of SBFL.

It can be seen that a mixed logic approach should be used. DCFL should be the most widely used gates and SDCFL, SBFL and UBFL should be used in special circumstances to enhance performance.

8. ACKNOWLEDGMENTS

The authors would like to thank Assoc. Prof. Kamran Eshraghian in the Centre for Gallium Arsenide VLSI Technology, South Australia for his informative discussions.

9. REFERENCES

[CURTICE]: Curtice, W. R.: "A MESFET Model for Use in the design of GaAs Integrated Circuits," IEEE Trans. Microwave Theory and Tech., MTT-28, pp. 448-456, May 1980.

[DAVENPORT]: Davenport, W. H.: "Macro Evaluation of a GaAs 3000 Gate Array," Proc. IEEE GaAs IC Symp., Grenelefe, Fla., pp. 19-22, October 1986.

[HILL]: Hill, C. F.,: "Noise Margin and Noise Immunity in Logic Circuits," Microelectronics, vol. 1, pp. 16-22, April 1968.

[LONG&BUTNER]: Long, S. I. and Butner S. E.: "Gallium Arsenide Digital Integrated Circuit Design", pp. 214-215, McGraw Hill, 1990.

[NAKAMURA]: Nakamura, H., et al.: "A 390ps 1000 Gate Array Super-Buffer FET Logic." Dig. of Tech. Papers, 1985 Int. Solid-State Cir. Conf., pp 204-205, February 1985.

Appendix E. Power Requirement Calculations

The anticipated power dissipation for a Buffer chip is 1W whereas for the Router chip and the Multiplexer chip, they are 0.7W and 0.8W respectively[11]. As a result, the total power dissipation of a *2 x 2* switching element (which composes of one Multiplexer chip, one Router chip and two Buffer chips) is 3.5W. Similarly, the power dissipation of a 4-to-1 multiplexer (which composes of three Multiplexer chips) is 2.4W whereas the power dissipation for a 1-to-4 demultiplexer (which composes of six Buffer chips and three Router chips) is 8.1W.

For a *1024 x 1024* 622Mb/s switch, it requires 1920 (128 rows by 15 columns) *2 x 2* switching elements, 256 4-to-1 multiplexers and 256 1-to-4 demultiplexers. Summing all these together, the total power dissipation of the switch becomes:

$$1920 \times 3.5W + 256 \times 2.4W + 256 \times 8.1W = 9408W.$$

Note that this calculation is based on the worst case condition that all the elements in the multiplexers and demultiplexers run at their maximum speed. In reality, different stages of these components run at different speed and there is a slight power saving. However, it will not be as significant as that for CMOS since the power dissipation in GaAs MESFETs is mainly static whereas it is mainly dynamic in CMOS.

For comparison, a *1024 x 1024* 155Mb/s switch requires 352 (128 rows by 15 columns) *2 x 2* switching elements, 64 4-to-1 multiplexers and 64 1-to-4 demultiplexers. Summing all these together, the total power dissipation of the switch becomes:

$$352 \times 3.5W + 64 \times 12W + 64 \times 40.5W = 4592W.$$

The power saving is achieved by the use of the bit-rate conversion technique which increases the incoming data rate so that the number of *2 x 2* switching elements is reduced.

---

11 All power dissipations are based on simulation results running at 50% higher than their designed operational speed at a 125°C environment with typical (tt) parameters.

Bibliography

[ABDEL,R&Y]: I. M. Abdel-Mortaleb, W. C. Rutherford and L. Young. *GaAs Inverted Common Drain Logic (ICDL) and Its Performance Compared with Other GaAs Logic Families*. Solid-State Electronics, 1987, vol. 30, pp. 403-414.

[BARNA&L]: A. Barna and C. Liechti. *Optimization of GaAs MESFET Logic Gates with Subpicosecond Propagation Delays*. IEEE Journal of Solid-State Circuits, vol. SC-14, pp. 708-715, August 1979.

[BENES]: V. E. Benes. *Optimal Rearrangeable Multistage Connecting Networks*. Bell Systems Technical Journal, vol. 43, no. 7, pp. 1641-1656, July 1964.

[BOSCH]: B. G. Bosch. *Gigabit Electronics—A Review*. Proceedings IEEE, vol. 67, March 1979, pp. 340.

[BUSHEHRI]: E. Bushehri. *Critical Design Issues for Gallium Arsenide VLSI Circuits*. Ph. D. thesis, Middlesex Polytech University, London, 1992.

[CHU1]: E. Chu. *Module Description (MOB) for Primitives Used within the Buffer Chip in High Bandwidth Packet Switch*. Jydsk Telefon Internal Technical Report, version 1, November 1993.

[CHU2]: E. Chu. *Module Description (MOB) for Input Control of Buffer Chip in High Bandwidth Packet Switch.* Jydsk Telefon Internal Technical Report, version 1, November 1993.

[CHU3]: E. Chu. *Module Description (MOB) for Buffer Manager of Buffer Chip in High Bandwidth Packet Switch.* Jydsk Telefon Internal Technical Report, pilot version, November 1993.

[CHU4]: E. Chu. *Module Description (MOB) for Output Control of Buffer Chip in High Bandwidth Packet Switch.* Jydsk Telefon Internal Technical Report, version 1, November 1993.

[CHU&JAKOBSEN]: E. Chu and J. Jakobsen. *Comparison of GaAs Static Logic Families.* Proceedings of the 11th NorChip Seminar, Trondheim, Norway, November 1993, pp. 93-98.

[CMP]: *Foundry Design Manual Version 5.0.* Thomson Composants Microondes, March 1991.

[CURTICE]: W. R. Curtice. *A MESFET Model for Use in the Design of GaAs Integrated Circuits.* IEEE Transactions on Microwave Theory and Technology, MTT-28, pp. 448-456, May 1980.

[DAVENPORT]: W. H. Davenport. *Macro Evaluation of a GaAs 3000 Gate Array.* Proceedings IEEE GaAs IC Symposium, Grenelefe, Fla., October 1986, pp. 19-22.

[DEPRYCKER]: M. de Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*. Ellis Horwood Limited, 1991.

[DUNCAN,S&S]: R. A. Duncan, K. C. Smith and A. S. Sedra. *Gallium Arsenide Pseudo-Current-Mode Logic*. Electronics Letter, 1990, vol. 26, pp. 2130-2132.

[EDEN1]: R. C. Eden. *Capacitor Diode FET Logic (CDFL)—Circuit Approach for GaAs D-MESFET IC's*. Proceedings 1984 IEEE GaAs IC Symposium, Boston, Massachusetts, pp. 11-14, October 1984.

[EDEN2]: R. C. Eden, B. M. Welch and R. Zucca. *Low Power GaAs Digital ICs Using Schottky Diode FET Logic*. Digest of Technical Papers, 1978 International Solid-State Circuit Conference, pp. 66-69, February 1978.

[EDEN,L,L,W&Z]: R. C. Eden, F. S. Lee, S. I. Long, B. M. Welch and R. Zucca. *Multi-level Logic Gate Implementation in GaAs IC's Using Schottky Diode FET Logic*. Digest of Technical Papers, 1980 International Solid-State Circuit Conference, pp. 122-123, February 1980.

[ESH,B,S,L,B&B]: K. Eshraghian, A. Blanksby, R. Sarmiento, C. C. Lim, E. Bushehri and R. Bayford. *Gallium Arsenide Design Methodology & Performance Estimates for Very High Speed Circuits Using Normally-Off Classes of Logic*. Proceedings of the 12th Australian Microelectronics Conference, pp. 227-232, Queensland, Australia, October, 1993.

[ESH,C,M&C]: K. Eshraghian, E. Chu, A. Moini and S. Cui. *Comparison of GaAs Static Logic Families Suitable for VLSI Implementation*. Internal Report GAAS-92-15, Centre for GaAs VLSI Technology, The University of Adelaide, South Australia, November 1992.

[ESH,S,C&N]: K. Eshraghian, R. Sarmiento, P. P. Carballo and A. Núñez. *Speed-Area-Power Optimisation for DCFL and SDCFL Class of Logic Using Ring Notation*. Microprocessing and Microprogramming 32, pp. 75-82, North Holland, 1991.

[ESHRAGHIAN]: K. Eshraghian. *Fundamentals of Very High Speed Systems: Gallium Arsenide VLSI Technology Course Notes*. Centre for GaAs VLSI Technology, The University of Adelaide, South Australia, 1991.

[ESHRAGHIAN92]: K. Eshraghian. *Design Methodology and Layout Style for Very High Speed Circuits and Subsystems*. Internal Report GAAS-92-4, Centre for GaAs VLSI Technology, The University of Adelaide, South Australia, January 1992.

[FIRSTENBERG]: A. Firstenberg. *GaAs ICs for New Defense Systems Offer Speed and Radiation Hardness Benefits*. Microwave Journal, pp. 145, March 1985.

[FULKERSON]: D. E. Fulkerson. *Feedback FET Logic: A Robust, High-speed, Low-power GaAs Logic Family*. IEEE Journal of Solid-State Circuits, 1991, vol. 26, pp. 70-74.

[GLOANEC]: M. Gloanec, et. al. *GaAs Digital Integrated Circuits (GaAs MESFET Circuit Design, R. Soares, ed.)*, Artech House, Chapter 8, 1988.

[GOKE&L]: L. R. Goke and G. J. Lipovski. *Banyan Networks for Partitioning Multiprocessor Systems*. Proceedings 1st Annual International Symposium in Computer Architecture, pp. 21-28, December 1973.

[HELIX,J,C&S]: M. J. Helix, S. A. Jamison, S. A. Chao, C. and M. S. Shur. *Fan Out and Speed of GaAs SDFL Logic*. IEEE Journal of Solid State Circuits, vol. SC-17, pp. 1226-1231, December 1982.

[HILL]: C. F. Hill. *Noise Margin and Noise Immunity in Logic Circuits*. Microelectronics, vol. 1, April 1968, pp. 16-22.

[HOE&SALAMA]: D. H. Hoe and C. A. T. Salama. *Dynamic GaAs Capacitively Coupled Domino Logic (CCDL)*. IEEE Journal of Solid-State Circuits, 1991, vol. 26, pp. 844-849.

[HSU]: M. S. Hsu. *Aspects of Designing a High Speed Analog to Digital Converter*. M. Eng. Sc. thesis, Department of Electrical and Electronic Engineering, The University of Adelaide, South Australia, 1992.

[I.121]: *Broadband Aspects of ISDN*. CCITT Recommendations, June 1990.

[I.361]: *BISDN ATM Layer Specification*. CCITT Recommendations, June 1990.

[ISHIKAWA]: H. Ishikawa, et al. *Normally-Off Type GaAs MESFET for Low Power, High Speed Logic Circuits*. IEEE Int. SSCC Digest, February 1977, pp. 200.

[JAKOBSEN93]: J. Jakobsen. *Buffered Benes Networks with Bit Rate Conversion*. Proceedings of the Australian Broadband Switching and Services Symposium 1993, July 1993, volume 2, pp. 363-370.

[JTFEB93]: J. Jakobsen. *Functional Description (FEB) for Buffer Chip in High Bandwidth Packet Switch*. Jydsk Telefon Internal Technical Report, version 2, September 1993.

[JTMOB93]: J. Jakobsen. *Module Description (MOB) for Start Extraction in Router Chip in High Bandwidth Packet Switch*. Jydsk Telefon Internal Technical Report, version 1, August 1993.

[JTSKS93]: J. Jakobsen. *Systems Specifications (SKS) for High Bandwidth Packet Switch*. Jydsk Telefon Internal Technical Report, version 2, April 1993.

[JTSYB93]: J. Jakobsen. *Systems Description (SYB) for High Bandwidth Packet Switch*. Jydsk Telefon Internal Technical Report, version 2, August 1993.

[KATSY]: S. Katsy, et al. *A Source Coupled FET Logic—A New Current-Mode Approach to GaAs Logics*. IEEE Transaction of Electron Devices, 1985, vol. ED-32, no. 6, pp. 1114-1118.

[LARUE,W&C]: G. Larue, T. Williams and P. Chan. *FET FET Logic: A High Performance, High Noise Margin E/D Logic Family*. GaAs IC Symposium, 1990, pp. 223-226.

[LOHS,S&DEG]: J. Lohstroh, E. Seevinck and J. De Groot. *Worst-Case Static Noise Margin Criteria for Logic Circuits and Their Mathematical Equivalence*. IEEE Journal of Solid-State Circuits, vol. SC-18, number 6, pp. 803-807, December 1983.

[LONG]: S. I. Long, et al. *MSI High Speed, Low Power, GaAs Integrated Circuits Using Schottky Diode FET Logic*. IEEE Transaction on Microwave Theory and Techniques, vol. MTT-28, no. 5, May 1980, pp. 466.

[LONG&BUTNER]: S. I. Long and S. E. Butner. *Gallium Arsenide Digital Integrated Circuit Design*. McGraw Hill, 1990.

[MELLOR]: P. J. T. Mellor, et al. *Capacitor-Coupled Logic Using GaAs Depletion mode FETs*. Electronic Letters, vol. 16, September 1980, pp. 749.

[NAKAMURA]: H. Nakamura, et al. *A 390ps 1000 Gate Array Super-Buffer FET Logic. Digest of Technical Papers*. 1985 International Solid-State Circuits Conference, February, 1985, pp. 204-205.

[NARY&LONG]: K. R. Nary and S. I. Long. *GaAs Two-Phase Dynamic FET Logic: A Low-Power Logic Family for VLSI*. IEEE Journal of Solid-State Circuits, vol. 27, no. 10, October, 1992, pp. 1364-1371.

[NUNEZ]: A. Nuñez. *A Survey of GaAs Computer Designs*. Microprocessing and Microprogramming 21, pp. 665-670, North Holland, 1987.

[NUZILLAT]: G. Nuzillat, et al. *Quasi-Normally-Off MESFET Logic for High-Performance GaAs ICs*. IEEE Transaction on Electron Devices, vol. ED-27, no. 6, June 1980, pp. 1102.

[PARTRIDGE]: C. Partridge. *Gigabit Networking*. Addison-Wesley, 1993.

[PASTERNAK&S]: J. H. Pasternak and C. A. T. Salama. *GaAs MESFET Differential Pass-Transistor Logic*. IEEE Journal of Solid-State Circuits, 1991, vol. 26, pp. 1309-1316.

[PECZALSKI]: A. Peczalski, et al. *Design Analysis of GaAs Direct Coupled Field Effect Transistor Logic*. IEEE Transaction of Computer-Aided Design, 1986, vol. CAD-5, pp. 266-273.

[SIMONS]: M. Simons. *Radiation Effects of GaAs Integrated Circuits: A Comparison with Silicon*. GaAs IC Symposium, Technical Digest, 1983.

[STALLINGS]: W. Stallings. *Advances in ISDN and Broadband ISDN*. IEEE Computer Society Press, 1992.

[SUYAMA]: K. Suyama, et al. *Design and Performance of GaAs Normally-Off MESFET Integrated Circuits*. IEEE Transaction on Electron Devices; vol. ED-27, no. 6, June 1980, pp. 1092.

[SZE]: S. M. Sze. *Semiconductor Devices: Physics and Technology*. U. S. A., Bell Telephone Laboratory, 1985.

[TOBAGI]: F. A. Tobagi. *Fast Packet Switch Architectures For Broadband Integrated Services Digital Networks*. Proceedings of the IEEE, vol. 78, no. 1, January 1990, pp. 133-167.

[VANTUYL1]: R. L. Van Tuyl, et al. *High-Speed Integrated Logic with GaAs MESFETs*. IEEE Journal of Solid-State Circuits, vol. SC-9, October 1974, pp. 269.

[VANTUYL2]: R. L. Van Tuyl, et al. *GaAs MESFET Logic with 4GHz Clock Rate*. IEEE Journal of Solid-State Circuits, vol. SC-12, October 1977, pp. 485.

[VU&PECZALSKI]: T. T. Vu, A. Peczalski, et al. *The Performance of Source-Coupled FET Logic Circuits that Use GaAs MESFETs*. IEEE Journal of Solid-State Circuits, 1988, vol. 23, pp. 267-279.

[WELBOURN]: A. D. Welbourn, et al. *A High Speed GaAs 8-Bit Multiplexer using Capacitor-Coupled Logic*. IEEE Journal of Solid-State Circuits, vol. SC-18, no. 3, June 1983, pp. 359.

[WESTE&E]: N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective*. Addison Wesley, second edition, 1992, pp. 69-71.

[ZULEEG,N&T]: R. Zuleeg, J. K. Notthoff and G. L. Troeger. *Double-Implanted GaAs Complementary JFETs*. IEEE Electronic Device Letter, vol. EDL-5, pp. 21-23, January 1984.