1998

GaAs Implementation of FIR Filter

Anup Savla
Edith Cowan University

Edith Cowan University

Copyright Warning

Please note: page sequences are incorrect in some sections of this thesis. There is no page 17, 65 or 99. Page 106 has been incorrectly numbered as 98.

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

Acknowledgements

Project Supervisor: Dr. Stefan Lachowicz

Project Examiners: Dr. Stefan Lachowicz,
Dr. Ganesh Kothapalli.

I would like to express my sincere gratitude to my supervisor, Dr. Stefan Lachowicz, for his ongoing advice and guidance through this project. He has provided me with abundant amount of literature related to the thesis, and was also very helpful in other practical aspects of the project. He spent a considerable amount of time and effort to ensure the success of my project.

I am also thankful to Dr. Ganesh Kothapalli for his advice on improving the overall quality of my project and related thesis; and Dr. Daryoush Habibi for providing me with relevant literature related to video compression techniques. Suggestions and advice from post-graduate students in the department also proved very valuable.

Finally, I would like to thank my family and friends for their support during both this project and my undergraduate university career.

Anup Savla

Figures, Tables and Layouts

List of Figures.

List of Tables

List of Layouts

Table of Abbreviations

Abbreviation	Meaning	Abbreviation	Meaning
CCDL	Capacitively Coupled Domino Logic	nMOS	n Metal Oxide Semiconductor
DCFL	Direct Coupled FET Logic	PCS	Personal Communication Systems
DCT	Discrete Cosine Transform	PDLL	Pseudo Dynamic Latched Logic
DFT	Discrete Fourier Transform	pMOS	p Metal Oxide Semiconductor
FFT	Fast Fourier Transform	QMF	Quadrature Mirror Filter
FIR	Finite Impulse Response	SBFL	Super Buffered FET Logic
FLC	Ferro-Liquid Crystal	SNR	Signal to Noise Ratio
FSM	Finite State Machine	SPDL	Split Phase Dynamic Logic
IP	Intelligent Pixel	TDFL	Trickle Transistor Dynamic Logic
JPEG	Joint Photographic Experts Group	VHDL	VLSI Hardware Description Language
MESFET	Metal Semiconductor Field Effect Transistor	VHSI	Very High Speed Integrated Technology
MRN	Modified Ring Notation	VLSI	Very Large Scale Integration

TABLE OF CONTENTS

Abstract

This thesis discusses the findings of the final year project involving Gallium Arsenide implementation of a triangular FIR filter to perform discrete wavelet transforms.

The overall characteristics of Gallium Arsenide technology – its construction, behaviour and electrical charactersitics as they apply to VLSI technology – were investigated in this project. In depth understanding of its architecture is required to be able to understand the various design techniques employed. A comparison of Silicon and GaAs performance and other characteristics has also been made to fully justify the choice of this material for system implementation.

A lot of research and active interest has gone into the field of image and video compression. Wavelet-based image transformation is one of the very efficient compression techniques used. An analysis of discrete wavelet transformations and the required triangular FIR filter was done to be able to produce a transform algorithm and the related filter architecture.

Finally, the filter architecture was implemented as a VLSI design and layout. A variety of functional blocks required for the architecture were designed, tested and analysed. All these blocks were integrated to produce a model of a complete filter cell. The filter implementation was designed to be self-timed – without a system clock. Self-timed systems have considerable advantages over clocked architectures. Various design styles and handshaking mechanisms involved in designing a self-timed system were analysed and designed.

There are many avenues still to explore. One of them is the VHDL analysis of filter architecture. Further development on this project would involve integration of higher-level logic and formation of a complete filter array.

Introduction

Mobile communications technology has made tremendous development in the past few decades. The first generation devices in 1970/80's provided analog voice-only service, while the second generation devices in 1980/90's were capable of transmitting and receiving digital voice and data service. The next generation of communication devices will be heralded by truly Personal Communications Systems (PCS), which are expected to incorporate a high degree of multimedia functionality including voice, data, images and video.

A picture is worth a thousand words. This is literally true for multimedia storage and transmission devices. Transmission of high quality image and video takes vast amounts of bandwidth, making it too expensive and exclusive to be of general use. The current lack of high quality real-time image/video processing in small portable multi-purpose devices with sufficiently low power consumption has posed the greatest challenge for the technology. All these issues are being addressed coherently by the development of a Mobile Multi-Media Communicator.

The Communicator will enable advanced image and video processing, in addition to voice and data for mobile communications in the form of a portable handset for a true PCS. An Intelligent Pixel (IP) array is a circuit capable of realising such requirements. The entire IP array will be built within a single VLSI chip.

One of the most important features of the IP array is its capability to receive images and perform a wavelet transform on image data. A wavelet transform is a very effective data and image compression technique allowing video communications using low bandwidth. The IP array will have circuitry present to be able to produce a complete wavelet transform for the image. The FIR filter architecture will have to be implemented within the arithmetic control area for the IP array, and would thus perform discrete wavelet transforms on images received.

The implementation of the filter architecture also brings forward issues of power dissipation and circuit speed. To enable a fast system with less power dissipation, a self-timed architecture was chosen for the FIR filter. The absence of a global clock signal makes the

system more robust and greatly reduces current leakage in circuits. It requires incorporation of circuitry involved in handshaking - communicating with other parts of the system to be able to start and finish operation at appropriate times.

The speed requirement of the system is also responsible for the choice of GaAs as a design material. As the limitations of silicon based logic designs become increasingly apparent, alternative fabrication materials and technologies must be exploited. GaAs is one of the most promising new semiconductor materials used for VLSI design. Due to higher electron mobility and a direct bandgap, it has switching times much faster than silicon based circuits. A number of different transistor types occur in Gallium Arsenide. Out of these MESFET is the most established and tested device, and is suitable for application in very fast systems.

Project Definition

Aim

The aim of this project was to design a triangular (three tap) FIR filter capable of performing a discrete wavelet transform. This DWT algorithm is required to be implemented in an Intelligent Pixel (IP). The main objectives relating to this aim were:

- Investigation of properties of Gallium Arsenide, and a study of various transistor devices possible. A study of MESFET architecture led to further understanding of the layout rules for HGaAs III MESFET.

- Understanding the architectural requirements for an FIR filter cell to be able to perform a discrete wavelet transform within the Intelligent Pixel. This later led to design of the FIR filter system.

- The implementation of an FIR filter cell in Gallium Arsenide.

- Comparison of hardware and software simulations.

Scope

- The scope of this project involved analysing Gallium Arsenide and the related MESFET technology. Upon learning this, the next step was to understand layout rules for design using MAGIC software in the VLSI Research Laboratory.

- An understanding of self-timed systems and their architectures was essential. This was mainly obtained from IEEE transactions related to self-timed circuits in Gallium Arsenide.

- An in depth understanding of wavelets and FIR Filters was required. Again a major source was texts related to filter banks and IEEE transactions on wavelet transforms.

- Initially the wavelet transform chosen was Daubechie's Transform. However, this was later changed to a triangular wavelet transform, since the latter had a better probability of being fabricated. Understanding the triangular FIR filter and related wavelet transform

enabled the design of the filter cell using functional blocks. All these functional blocks were then implemented using MAGIC software and tested for validity and quality of outputs using the HSpice software, also in VLSI Research Laboratory.

- The scope of the project included testing of the hardware results with software simulation. VHDL simulation was considered as an option for this purpose. However, due to unavailability of software licence for VHDL and timing constraints on the project, a mathematical simulation was performed instead. This was done using MATLAB software in the VLSI Research Lab.

1 Gallium Arsenide Technology

1.1 Introduction

Silicon MOS technology has been the main medium for computer and system applications for a number of years and will continue to fill this role in the foreseeable future. However, silicon logic has speed limitations that are already becoming apparent in state-of-the-art fast digital system design. Paralleling developments in silicon technology, some very interesting results have emerged for Gallium Arsenide (GaAs) based technology. GaAs will not displace silicon but is being used in conjunction with silicon to satisfy the need for very high speed integrated (VHSI) technology in many new and innovative systems.

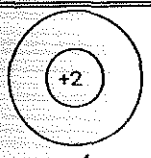
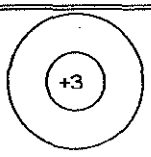
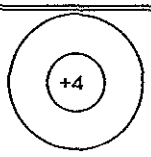
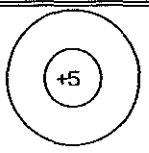
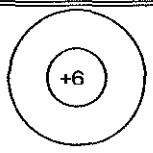
Group II	Group III	Group IV	Group V	Group VI
				
Be^{4}	B^{5}	C^{6}	N^{7}	O^{8}
Mg^{12}	Al^{13}	Si^{14}	P^{15}	S^{16}
Zn^{30}	Ga^{31}	Ge^{32}	As^{33}	Se^{34}
Cd^{48}	In^{49}	Sn^{50}	Sb^{51}	Te^{52}

Table 1.1: Periodic Table for Semiconductor Materials. Adopted from Eshraghian (1994)

As the table 1.1 indicates, there are numerous semiconductor materials. Much of the development work in semiconductor material technology that has paralleled that in silicon has been related to groups II-VI and groups III-V compounds. Out of these, GaAs, a group III-V compound has shown the most promise (Eshraghian, 1994).

GaAs is a compound semiconductor which was first discovered in 1926. In the early stages of development, GaAs devices were used as discrete components in microwave circuits and light emission applications in the late 1960's. The development of digital integrated circuit fabrication technology in the 1970's made integrated GaAs products a possibility and finally as the result of significant advances in ion implantation in the 1980's, GaAs VLSI technology is a commercial reality in the 1990's.

The direction from Ga to the nearest As is generally denoted by a Miller Index of (III). A convenient physical interpretation of the Miller Index is the fact that it represents the number of time planes of a particular type that intercept the crystal axes within one unit cell. For example, in the case of (110), planes intercept the x and y axes once, and are parallel to the z-axis.

1.3 Advantages of GaAs

GaAs transistors have two distinct advantages over Si transistors: speed and power. For the same power dissipation, a GaAs circuit is usually faster, and at the same speed, the power in a GaAs circuit is usually lower (at higher frequencies). The speed advantage comes from the fact that the peak average electron velocity in intrinsic or doped GaAs is several times higher than in Silicon and it is reached at a much lower value of electric field, and hence with a lower supply voltage. Since the current density in a device is proportional to the electron velocity, the amount of current available to charge or discharge a capacitor in a GaAs device is much larger and the switching speed is therefore higher than in a Si device with the same dimensions. In addition, a GaAs field effect transistor does not have p-n junction around its drain and source terminals and therefore the interelectrode capacitance in a GaAs device is much smaller. Smaller capacitance and higher current density, combined with a smaller voltage swing in a GaAs transistor, contribute to the realisation of low-power, high-speed circuits (Cui, 1996).

Both discrete components and integrated circuits made in GaAs are faster than those made of silicon because its low-field electron mobility is large, and it has a low saturation field. Devices made in semi-insulating GaAs substrates have low parasitic capacitance, which leads to further improvement in speed.

GaAs also has the ability to emit light, which makes it useful for making lasers, light-emitting diodes, and microwave emitters used in cellular phones.

1.4 Gallium Arsenide Devices

To further understand the significance of GaAs-based designing, we must understand the structure of the various devices constructed using GaAs. We know that silicon-based technology offers a lot of popular devices like nMOS, pMOS, CMOS and BiCMOS. Similarly there are different devices available in GaAs with different characteristics in terms of speed, load driving capacities and noise-thresholds. To better appreciate the characteristics of the logic families used in GaAs, we must first gain knowledge of various devices possible using GaAs. Some of these devices have been described below. A detailed description of the operation of GaAs MESFET, which shows resemblance to the silicon nMOS device, has been given.

A number of different GaAs devices have been developed, and can be classified into first and second generation devices. First generation GaAs devices include

- Depletion-mode metal semiconductor field-effect transistor, D-MESFET;
- Enhancement-mode metal-semiconductor field-effect transistor, E-MESFET;
- Enhancement-mode junction field-effect transistor, E-JFET; and
- Complementary enhancement-mode junction field-effect transistor, CD-JFET.

First generation GaAs devices have exhibited switching delays as low as 70 to 80 picoseconds for a power dissipation in the order of 1.5mW to 150μW.

The more sophisticated second generation devices include:

- High electron mobility transistor, HEMT;
- Heterojunction bipolar transistor, HBT.

Electron mobility in second generation transistor can be up to five times greater than in the first generation. In consequence, very fast devices are possible.

1.4.1 The Metal-Semiconductor FET (MESFET)

The GaAs MESFET, a bulk-current-conduction majority-carrier device, is fabricated from bulk GaAs by high-resolution photolithography and ion implantation into a semi-insulating GaAs substrate. Processing is relatively simple, requiring no more than six to eight masking

stages. For the purpose of comparison, Figure 1.2 shows the evolution of process complexity in terms of mask count as a function of time for both Silicon and GaAs.

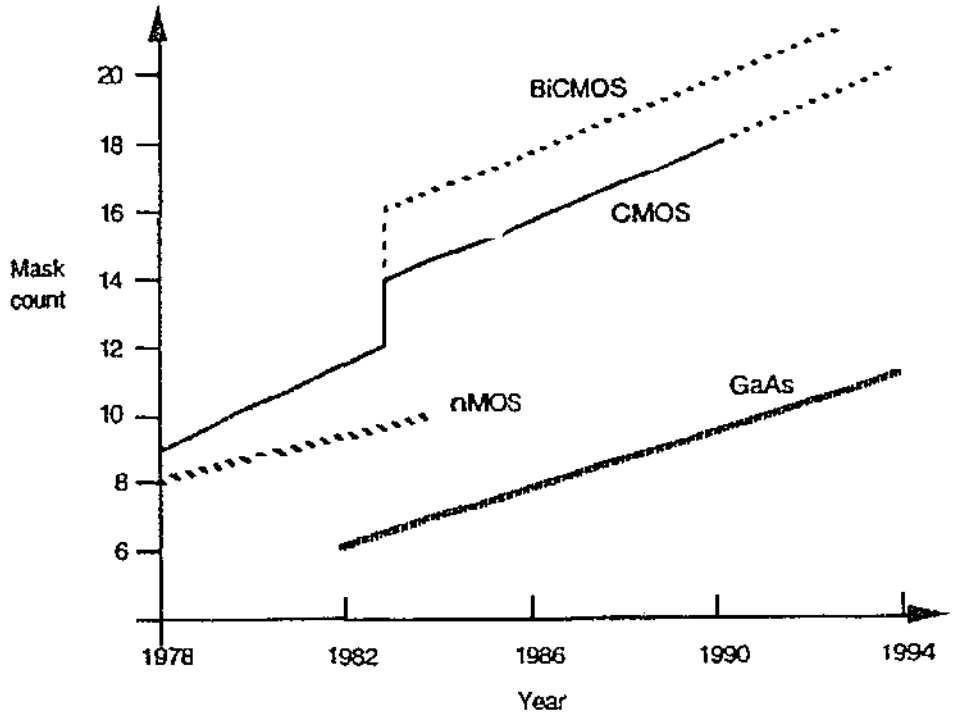


Figure 1.2 Evolution of Process Complexity for Silicon and Gallium Arsenide technologies. Adopted from Eshraghian (1994)

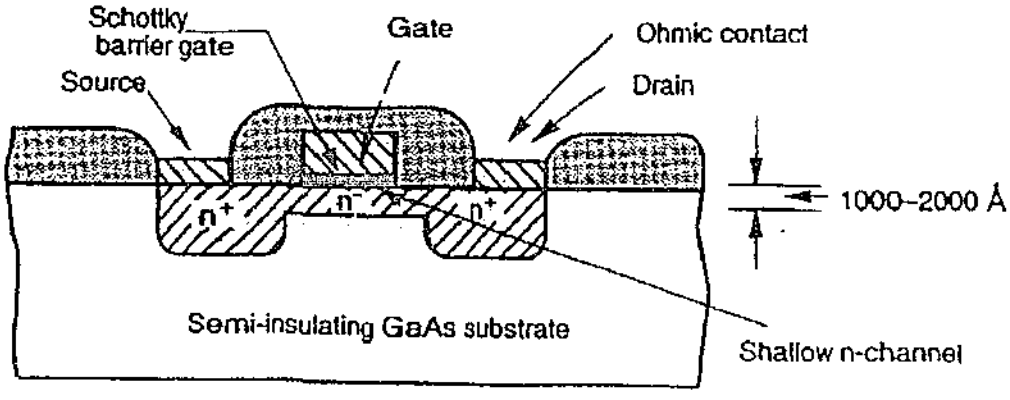


Figure 1.3 Side View of the MESFET. Adopted from Eshraghian (1994)

The structure of basic MESFET as shown in figure 1.3 is very simple. It consists of a thin n-type active region joining two ohmic contacts with a narrow metal Schottky barrier gate that separates the more heavily doped drain and source. (Eshraghian, 1994)

GaAs MESFETs are similar to silicon MOSFETs. The major difference is the presence of Schottky diode at the gate region which separates two thin n-type active regions, that is, source and drain, connected by ohmic contacts. Both D type and E type MESFETs, that is, 'ON' and 'OFF' devices, operate by the depletion of an existing doped channel. This can be contrasted with silicon MOS devices where the E (enhancement) mode transistor functions by inverting the region below the gate to produce a channel, while the D (depletion) mode device operates by doping the region under the gate slightly in order to shift the threshold to a normally 'ON' condition.

This similarity provides us with the basis for extending to GaAs the design methodology used so successfully in silicon to simplify circuit and system design and layout issues.

The D-MESFET is normally 'ON' and its threshold voltage, V_{tdep}, is negative. The E-MESFET is normally 'OFF' and its threshold voltage V_{tenh} is positive. The threshold voltage is determined by the channel thickness, a, and concentration density of the implanted impurity, N_D. A highly doped, thick channel exhibits a larger negative threshold voltage. By reducing the channel thickness, and decreasing the concentration density a normally 'OFF' enhancement mode MESFET with a positive threshold voltage can be fabricated. Circuit symbols for the depletion and enhancement mode MESFETs are set out in figure 1.4.

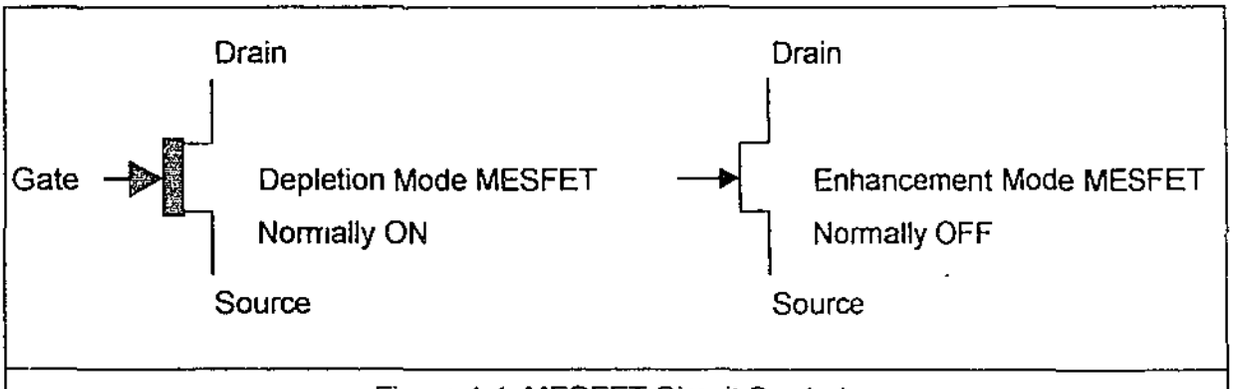


Figure 1.4: MESFET Circuit Symbols

The MESFET has a maximum gate to source voltage V_{gs} of about 0.7-0.8 volt owing to the diode action of the Schottky diode gate. Thus the logic '1' in these devices corresponds to a 0.7 V voltage. (Butner and Long, 1990)

1.5 The Vitesse E/D GaAs IC Technology

The previous section described characteristics of the GaAs MESFET. This description holds constant for MESFET devices produced by various wafer-fabricating foundries. However the manufacturing processes differ between them, resulting in changes in the level of scaling (minimum gate length) and the layout rules of a device. Consequently, different manufacturers supply their own technology files, which are in turn used by the layout-editing software to make sure that chip layouts for the circuits comply to the design rules required by the corresponding fabrication process.

The MESFET fabrication process and technology used for designing circuits for this design was the H-GaAs III developed by Vitesse Semiconductor Corporation. H-GaAs III is a third generation process characterised by 0.6 μm effective gate length. The information and facts provided in this section are obtained from the Vitesse Foundry Manual (1993), and in-text reference has not been made for all the material adopted.

1.5.1 The Technology Overview

The Vitesse H-GaAs III foundry process is an advanced, proprietary process for the fabrication of high-performance GaAs VLSI digital circuits. H-GaAs III is a scaled version of the original Vitesse process developed in 1986. It was designed to produce circuits with low power dissipation and high speed at high levels of integration. To achieve these characterstics, high-performance self-aligned MESFETs are used and up to five levels of interconnections are provided; the fourth and fifth are available to simplify power distribution on large chips. H-GaAs III is continuously improved to provide higher performance and greater yield. For the H-GaAs III implemented in 1993, circuits with more than 1 million active devices could be implemented. (Vitesse Foundry Manual, 1993)

Two versions of the H-GaAs III are available for customer use. The H-GaAs III "P" version uses polyimide as the intermetal insulator. Polyimide reduces the routing capacitance resulting in faster loaded gate delays. It is rated for use over the commercial temperature range (max T_J = 100°C). The H-GaAs III "O" version uses SiO_2 as the intermetal dielectric. The larger dielectric constant of SiO_2 results in circuits that are approximately 20% slower but can operate over a wider temperature range (eg. -55°C to +125°C).

Three active devices, an enhancement mode MESFET, a depletion mode MESFET, and a Schottky-barrier diode are the active elements available for circuit design. The transistors

are used for switching and as active loads. The Schottky-barrier diodes are built from depletion mode MESFETs and are used primarily for level shifting. Minimum sized devices have nominal printed gate lengths of 0.4μm. The effects of series resistance are minimized by the self-aligned procedure used to define the channel and contact regions. The transistors therefore have high transconductance and large current-handling capacity.

The choice of the mean value of threshold voltage (V_T) for enhancement and depletion mode transistors in H-GaAs III is primarily determined by the logic family chosen (DCFL) for most Vitesse digital products. The threshold voltages (V_{TE} and V_{TD}) are centered at values which give the highest circuit performance (i.e. lowest τ_{pd}, P_d) achievable consistent with the observed variations of MESFET threshold voltages locally across a die and, globally, across the process. The threshold voltage, V_T, determines the relative sizes (gate widths) of the transistors in ratioed logic designs, and consequently, the switching speed of the logic gate. Nominal values of V_{TE} and V_{TD} for the 1993 H-GaAs III model are given in Table 1.2. Typical DCFL gate delays for devices with these threshold voltages, determined from FI=FO=1 ring oscillator performance are 50-100ps/gate. (Vitesse Foundry Manual, 1993)

Transistor Type	V_T	Global σV_T
Enhancement	220mV	60mV
Depletion	-825mV	60mV

Table 1.2: Vitesse IC Process Parameters for W=10μm and L=.6μm Transistors. Adopted from Vitesse Foundry Manual. V. 6 (1993)

1.5.2 Process Description

The H-GaAs III fabrication sequence consists of 32 major steps, and requires upto 13 mask levels. The substrate material is 4-inch diameter, undoped, semi-insulating GaAs. A direct step-on-wafer exposure system is used for pattern definition. This system uses 5X image reduction and provides a variable wafer-level reticle area for circuits and devices. Alignment marks and process control monitors (standard electrical test structures) are external to the design space.

The individual steps which comprise the Vitesse E/D process are summarised in table 1.3. Subsequent to incoming inspection, the wafer is cleaned and cappped with a dielectric film.

The film is patterned and silicon ions are implanted to form the transistor channels. A two-step implant scheme is used to define all transistor regions and distinguish enhancement-mode from depletion mode devices. This approach ensures parametric tracing between the two types of devices. A refractory metal, which is used to form the gate, is deposited next. The gate metal, after being patterned, serves as a mask to protect the channel region from the high-dose implant used to create the low-resistance source and drain regions. A single high-temperature thermal annealing cycle is used to activate the implanted ions. Multi-element metal films are deposited and sintered to form contacts to the source and drain regions of the transistors. A cross-section of a typical MESFET after these steps have been completed is shown in figure 1.5

1	Deposit Dielectric Cap	17	Interlayer Dielectric Deposition
2	Mask 1 - Active Area Definition	18	Mask 7 - Via 2 Definition
3	Enhancement Device Implant	19	Metal 2 Deposition
4	Mask 2 - Depletion Implant Definition	20	Mask 8 - Metal 2 Interconnect Definition
5	Depletion Device Implant	21	Metal 2 Etch
6	Gate Metal Deposition	22	Interlayer Dielectric Deposition
7	Mask 3 - Gate Metal Definition	23	Mask 9 - Via 3 Definition
8	Implant Source and Drain Regions	24	Metal 3 Etch
9	Activation Anneal	25	Mask 10 - Metal 3 Interconnection Definition
10	Mask 4 - Source and Drain Metallization	26	Metal 3 Etch
11	Source/Drain Metal Sintering	27	Interlayer Dielectric Deposition
12	Interlayer Dielectric Deposition	28	Mask 11 - Via 4 Definition
13	Mask 5 - Via 1 Definition	29	Metal 4 Deposition
14	Metal 1 Deposition	30	Mask 12 - Metal 4 Interconnect Definition
15	Mask 6 - Metal 1 Interconnect Definition	31	Passivation Dielectric Deposition
16	Metal 1 Etch	32	Mask 13 - Passivation Contact Definition

Table 1.3 Major Steps in Vitesse GaAs Digital IC Fabrication Process. Shaded area shows steps not applicable to the version used. Adopted from Vitesse Foundry Design Manual V. 6 (1993)

The remainder of the process consists of the deposition and patternising of both dielectric films and interconnect metalisation. Spin-on glass is used for planarisation when the intermetal insulator is SiO_2. This is not required in H-GaAs III "P" since polyimide is self-planarising.

Five levels of interconnection are available *as a default* in the Vitesse process; gate metal, Metal 1, Metal 2, Metal 3, and Metal 4. Of these, gate metal, Metal 1, and Metal 2 are typically used for signal routing. Metal 3 and Metal 4 are typically reserved for power and ground busses. Interconnections formed using gate metal are generally restricted to short distances due the high sheet resistance of the material.

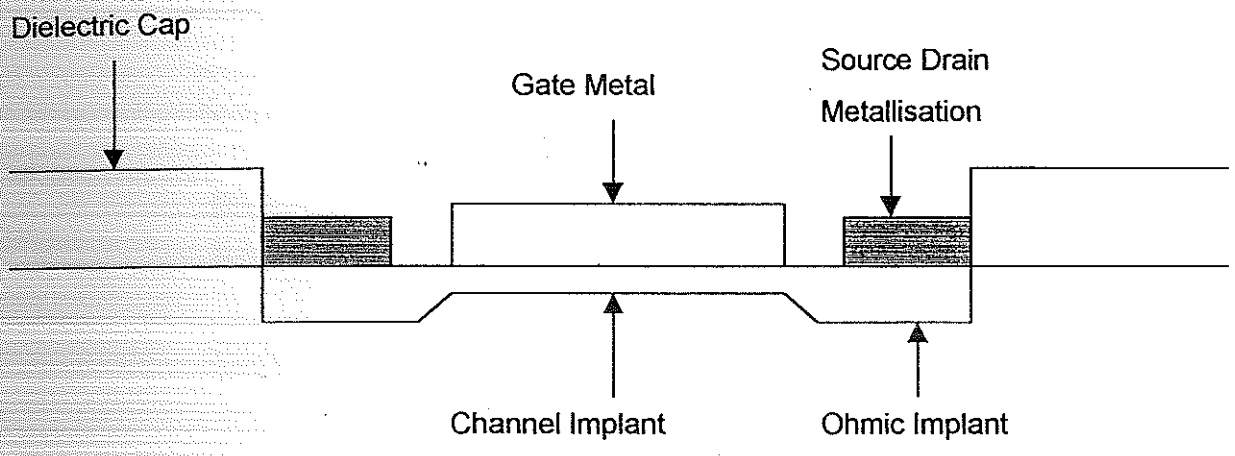


Figure 1.5 MESFET Cross-section after completion of step 11. Adopted from Vitesse Factory Design manual. V. 6. (1993)

Metal 3 can also be used when additional interconnect resources are necessary. Typically Metal 4 is used for power distribution on large die where it is necessary to minimize the voltage drop due to ohmic losses. On small die it may be eliminated. Table 1.4 compares the pitch, typical values of the resistance and electromigration limits of each interconnect layer.

	Pitch (μm)	R_s (Ω/□)	I_{max}(mA/μm)
Gate Metal	2.7	1	5
Metal 1	3.1	<0.07	1
Metal 2	3.6	<0.035	2
Metal 3	7	<0.025	2.8
Metal 4	-	<0.025	2.8

Table 1.4: H-GaAs III Nominal Interconnect Parameters. Values valid at 25°C. Adopted from Vitesse Factory Design Manual (1993)

Wafer fabrication is completed by the depostion and patterning of a protective passivation dielectric. A cross-sectional representation of a FET after completion of the fabrication cycle is shown in Figure 1.6. The finished wafers are probed to determine whether the device characterstics are within the specified process window. After these tests have been completed, the wafers are ready for circuit characterisation, packaging, final test, and shipment.

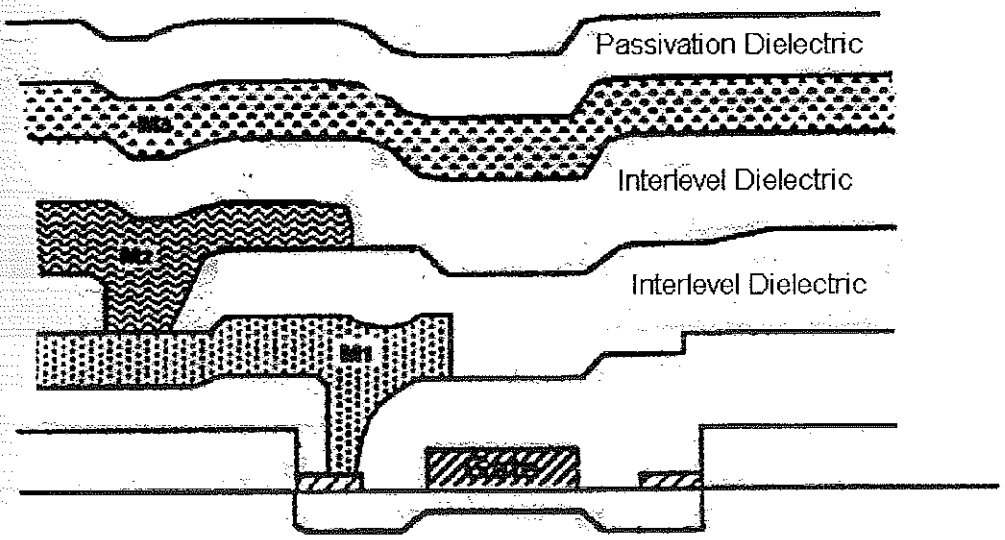


Figure 1.6: Cross Section of MESFET after completion of 11-Mask Fabrication Cycle. Adopted from Vitesse Foundry Design Manual. V. 6. (1993)

Notice that the above described five metal layers used in layout design are described as the *default* for the Vitesse technolgy. The H-GaAs III technology used for the project, however, is only a *three-metal version* and requires only 11mask layers. It can be used effectively for circuits containing up to several thousand gates, like the FIR filter circuit. The reduced process also has a shorter turnaround time from the design submission to chip shipment.

In the three-metal version, gate metal is used to form logic gates for E/D MESFET logic; Metal 1 is used to form metal connections within the individual logic cells and Metal2 is used for global power, ground and clock signals.

1.6 Comparison between GaAs and Silicon Devices

Having covered a brief description of the various GaAs devices and the H-GaAs III process technology, it is appropriate to provide a comparison between the GaAs MESFET and Si MOSFET devices to be able to judge the relative advantages and disadvantages of adopting GaAs based devices for implementing the project.

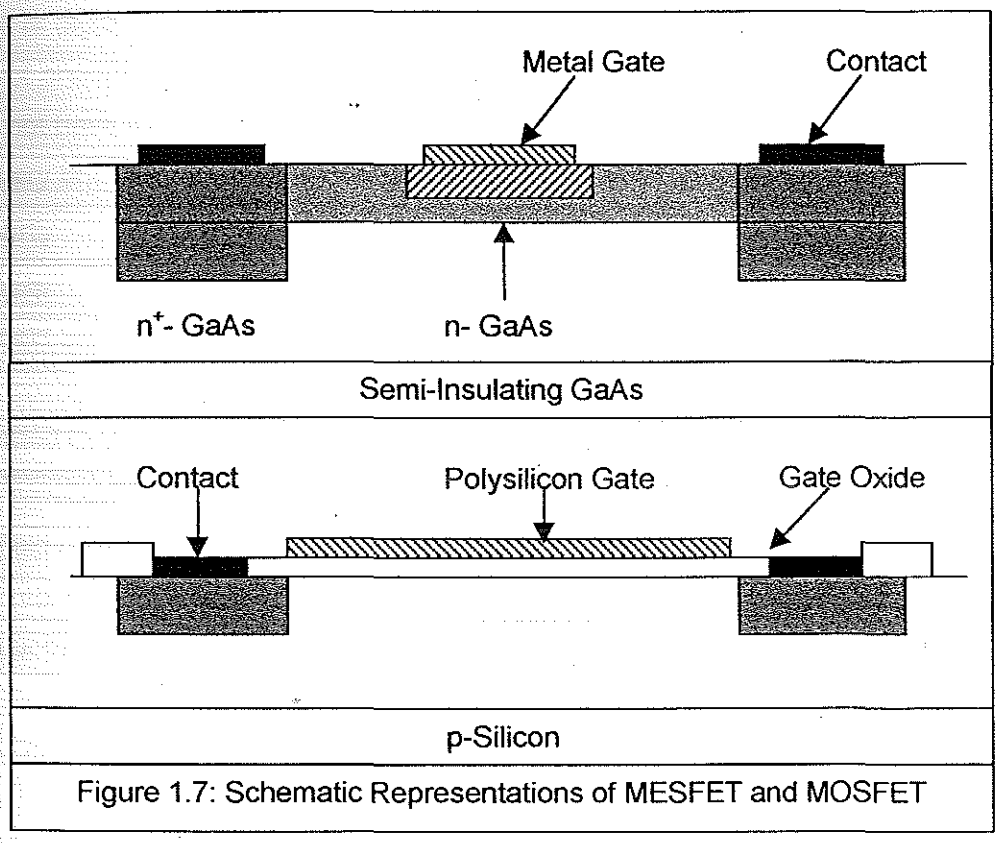


Figure 1.7: Schematic Representations of MESFET and MOSFET

The structure of a typical GaAs MESFET and Si MOSFET are compared in figure 1.7. The main differences between the devices are:

- The formation of the channel
- The coupling of the gate-control electrode to the channel

The GaAs MESFET uses a thin, doped channel whose thickness is controlled through depletion by a metal/semiconductor junction. The metal gate directly contacts the channel. The MOSFET, other the other hand, forms a channel when the silicon surface is inverted (p-type silicon now contains a high density of electrons due to band-bending). The gate electrode is separated from the channel by a thin oxide dielectric layer.

The material and electronic properties of GaAs and Silicon are compared in table 1.5. From this table it can be seen that GaAs has a relatively large band gap. This means that it can operate at relatively high temperatures. Its intrinsic carrier concentration is low so the material is a semi-insulator. Its resistivity is large so special measures need to be taken to provide isolation between devices on the chip. (Cui, 1996). Direct band gap leads to short lifetime of minority carriers so that electron-hole pairs generated by radiation, for example, will recombine quickly before they can cause degradation of circuit performance. Its high electron mobility gives rise to the high cut-off frequency of amplifiers and fast switching speed of digital circuits. On the other hand, its hole mobility is disproportionately low so that it is not practical to construct complementary circuits such as Si CMOS. In addition, GaAs does not have a native oxide, so that it is not possible to build a MOS-like structure. GaAs is also very brittle and yield loss through handling is significant.

Property	GaAs	Silicon
Bandgap	Direct; 1.42 eV	Indirect; 1.12 eV
Low-field electron drift mobility	5000 $cm^2/(V.s)$ at $N_D = 10^{17}/cm^{-3}$	800$cm^2/(V.s)$
Saturated Drift Velocity	8×10^6 cm/s	6.5×10^6 cm/s
Peak Electron Velocity	1.7×10^7 cm/s	6.5×10^6 cm/s
Low-field hole (drift) mobility	250 $cm^2/(V.s)$	300 $cm^2/(V.s)$
Substrate Resistivity	10^6 to 10^8 Ω.cm	Low
Surface State Density	High ($10^{12}/cm^{-2}$ or greater)	Low
Native Oxide	Several reactive and unstable compounds of Ga and As	S_iO_2; very stable
Effective electron mass	0.063 m_0	0.33 m_0
Effective hole mass	0.090 m_0	0.16 m_0
Dielectric Constant	13.1 ε_0	11.9 ε_0
Minority Carrier Lifetime	10^{-8} s	2.5×10^{-3}s
Thermal Conductivity	0.46 W/cm-C°	1.5 W/cm-C°
Intrinsic Debye length	2290μm	24μm
Breakdown field	4×10^5V/(cm)	3×10^5V/(cm)
Schottky barrier height	0.7 to 0.8 V	0.4 to 0.6 V

Table 1.5 Electronic Properties of GaAs and Si. Adopted from Cui (1996)

A significant point worth notice while comparing GaAs MESFET and silicon MOS is the *similarity between GaAs MESFET and nMOS circuit designs*. Like nMOS and pMOS devices, GaAs D and E-MESFET are normally-on and normally-off respectively. The similarity in circuit design is apparent by looking at the basic inverter configuration for D-MESFET and nMOS. This similarity allows us to use a design style much similar to the well-established nMOS technology. Also like the nMOS, there is no need to explicitly define complimentary logic in GaAs MESFET. This results in simpler layouts than CMOS.

2 Gallium Arsenide Logic Families

2.1 Introduction

Different types of transistor devices made available through the use of GaAs have been discussed in previous sections. The D-MESFET and E-MESFET are the two most mature technologies used extensively in logic design. For VLSI design the logic style and related transistor devices should be chosen so that the gate area and switching time are both optimised. A number of logic families have been developed for GaAs primarily to satisfy the needs of different design environments taking account of the maturity of the manufacturing process involved.

There are basically two main approaches towards static logic design:

- Normally-on logic;
- Normally-off logic;

The normally-on logic approach utilises D-MESFETs which are 'ON' devices. However, their major drawback is that they are required to be turned off. Thus the related dissipation together with the logic complexity renders this class of logic unsuitable for VLSI applications. The normally-off logic approach uses E-MESFETs as the switching element, thus remaining in off state as default.

2.2 Direct Coupled FET Logic (DCFL)

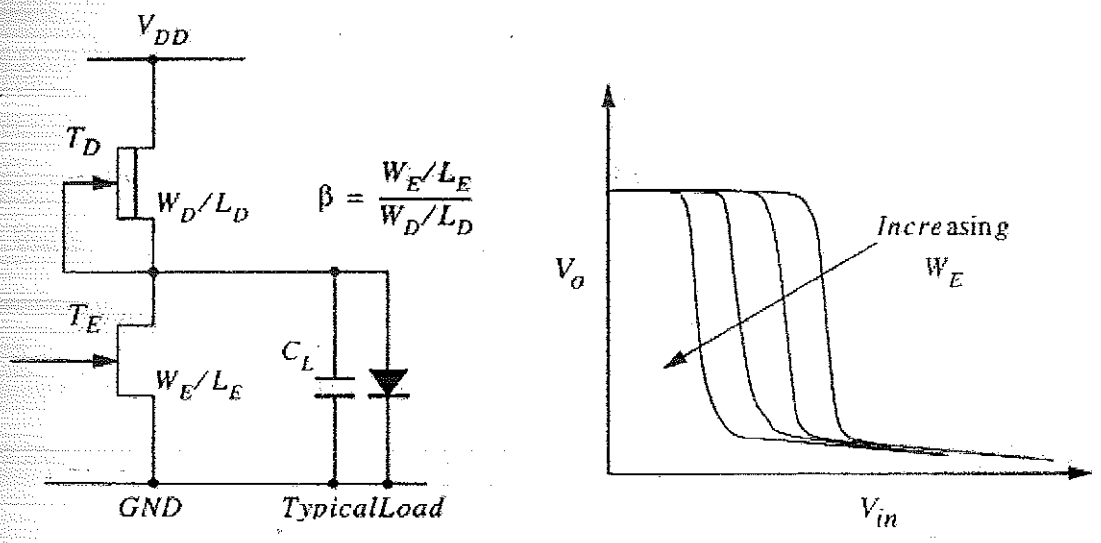
DCFL is the most popular logic family used for layout designing in GaAs. It uses nMOS-style inverter configuration to implement logic functions. For the design implemented in the project, we shall be using other logic families more suitable to self-timed designing. However, such other logic families are derived as extensions of the DCFL logic and build upon the functionality set using the DCFL family. Thus it is essential to cover the important characteristics of DCFL before describing other logic families relevant to the circuit being designed.

This class of logic, as the name suggests, is derived by coupling both D-type and E-type devices; it uses E-type device as the switching element and a D-type device as its pull-up or

load. DCFL circuits are thus combined E/D circuits. The simplicity of logic gates, together with the small chip area and relatively low power dissipation, makes this class of logic suited to *conventional* VLSI applications. The maturity of this approach along with advances in process technology and fabrication services have made this a popular class of logic for high performance VLSI circuits. (Cui, 1996)

To enable derivation of various DCFL characteristics, the design of a basic DCFL inverter has been illustrated in figure 2.1.

Direct-Coupled FET Logic (DCFL)



W_D/L_D determined *by* $I = C_L\, dV/dT$ requirement

W_E/L_E determined by restoring logic requirement (Node voltage must be $\leq V_{ol}$ to achieve proper logic low)

Baseline speed determined by g_m / C_g of enhancement switch and by β ratio

β ratio can be scaled to drive various capacitive loads

Figure 2.1: Summary of DCFL Basic Characteristics

As mentioned earlier, one of advantages of GaAs is that it implements logic in similar manner to silicon nMOS circuits. Thus the behaviour of the inverter is easily deduced by relating it to the nMOS inverter. When the input is a logic '0', the E-MESFET is 'cut-off' an

the conduction path is through the load D-MESFET, which connects the positive V_{DD} supply to the output, making it logic '1'. When the input is at logic '1', the E-MESFET is 'ON' and output of the inverter is connected to GND through the channel resistance of the E-MESFET, giving output '0'. The 'low' and 'high' voltages depend upon the choice of width and length ratio for the E-MESFET and D-MESFET.

In this device, D-MESFET always behaves as a constant current source and hence draws almost a constant current regardless of the logic states, whereas the E-MESFET behaves as a resistor or an open switch during input '1' and '0' respectively. Thus the saturation current I_{dssat} through the D-MESFET is switched between either the output that drives the next stage or alternatively through the 'ON' of the E-MESFET. Thus the current drawn from V_{DD} supply rail is constant irrespective of input (1 or 0). Thus DCFL causes minimum voltage perturbations on the supply rail.

2.2.1 Voltage Swing

To establish the logic voltage swing ΔV_O for the DCFL inverter two conditions must be satisfied:

1. The low logic voltage level V_{low} must satisfy

$$V_{low} < V_{tenh}$$

Which means we need to ensure that the E-type device can be turned-off

2. The gate of the switching E-type device should not be higher than the barrier potential, Φ_B. The logic high level, V_{high}, therefore should satisfy

$$V_{high} < \Phi_B$$

Thus the maximum voltage swing can be expressed as:

$$\Delta V_O = V_{high} - V_{low}$$
$$= \Phi_B - V_{tenh}$$

The logic swing ΔV_O for a DCFL inverter is in the order of 500mV to 600mV (Eshraghian, 1994)

2.2.2 Pull-up to Pull-Down Ratio (Z_{pu}/Z_{pd})

The transfer characteristic for the DCFL inverter has been shown in figure 2.1. The ideal pull-up/pull-down ratio for any device can be obtained by cascading it with an inverter and then determining the ratio such that there is no degradation of signal. Thus we need to cascade two inverters without degradation of logic levels. This requirement can be expressed by the equation

$$V_{in} = V_O = V_{inv}$$

Substituting the values of threshold voltages V_{tdep} = -800mV, V_{tenh}= +150mV and with V_{inv} = $\Phi_B/2$ = 400mV, we obtain the principal result for the pull-up to pull-down ratio, giving

$$Z_{pu}/Z_{pd} = 8/1.$$

However, the ratio of 10/1 is used as a rule-of-thumb value in DCFL design.

2.3 Pseudo Dynamic Latched Logic

The aim of introducing DCFL logic family in the previous section was to provide the reader with a background of logic design basics in GaAs, which are essential in grasping the essence of other logic families.

GaAs systems are mainly based on static logic families. DCFL permits layouts together with a good power-delay tradeoff. However, when supporting high fan-out or large load capacitance, the use of other logic families such as Super Buffered FET logic (described in appendix) is a must, creating what has been named merged logic. The drawback when using these two logic families is the high power dissipation and the poor resources to use, based exclusively on low fan-in NOR gates and inverters. Due to these facts, a new kind of logic family was suggested by Eshraghian (1993), named Pseudo Dynamic Latched Logic (PDLL). The filter architecture for the project was implemented using the PDLL and Latch Coupled FET Logic (LCFL), both of which are derivatives of the DCFL..

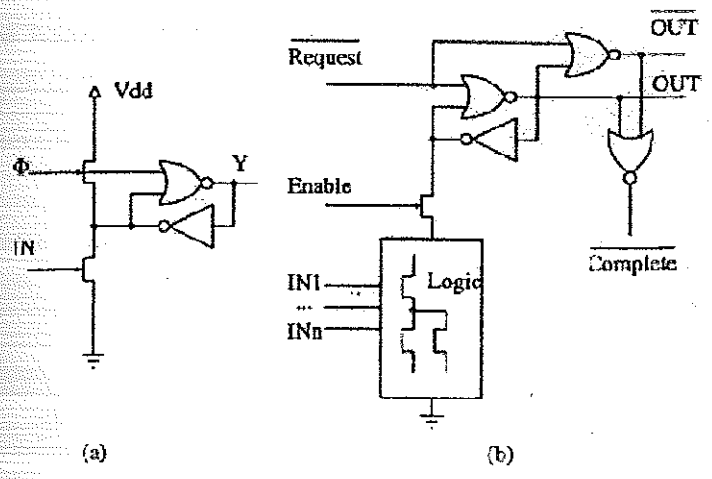


Figure 2.2: The PDLL and LCFL Logic Cells. Adopted from Lachowicz et al (IEEE Transaction [1])

Recently introduced, PDLL and LCFL GaAs logic families have shown to be a good compromise for both synchronous (with a global clock), and asynchronous (self-timed) systems. They compare well with other GaAs design styles in terms of speed/area and speed/(area x power) figures of merit and are especially efficient for highly pipelined systems. The main advantage of the latched structure is provided by the feedback which ensures that noise margin is higher than a DCFL gate (Lachowicz et al: IEEE Transaction [1]). This enables us to use serial connections of E-type transistors in the pull-down section. Therefore, in GaAs latched logic it is possible to implement logic gates based on the AND function which gives more freedom in the design and leads to more area efficient circuits. LCFL circuits are essentially just modifications of the PDLL logic suited to perform self-timed design, as required by the project.

2.3.1 Pseudo Dynamic Latched Configuration

PDLL gates are composed of a dynamic circuitry where the logic is performed in the pull-down section using E-MESFETs. Around the pull-up section is a static latch whose function is to permanently refresh the data stored in the dynamic mode. This double nature allows PDLL to get benefits of both static and dynamic GaAs logic families, allowing low operating

frequencies. It also enables an increase in functional complexity and reduces the power dissipation in the circuit. At the same time, because of its self-latching characteristics, it allows the implementation of highly efficient pipelined systems.

2.3.2 PDLL Theory and Operation

The basic structure of a PDLL gate for a self timed architecture is shown in figure 2.3 together with its operational diagram. The clock signal depicted in the diagram is implemented as a request signal for asynchronous systems.

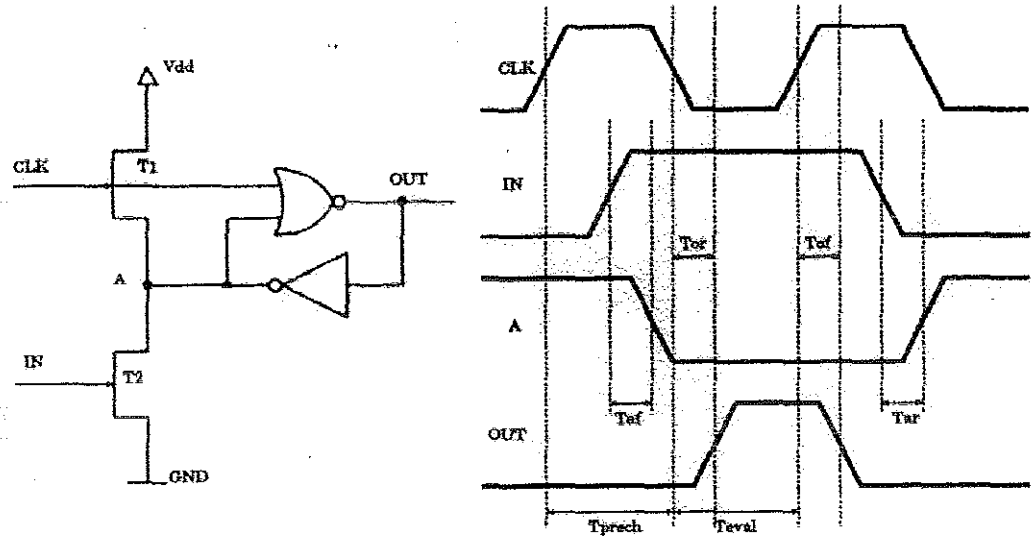


Figure 2.3: PDLL Latch and Operation. Adopted from Lopez et al

Transistors T_1 and T_2 form the logic while the NOR and inverter gates form the static latch which is implemented using DCFL. Because of the structure of their connection (creating a latch), the dimensions can be much smaller than normal DCFL gates, thus dissipating less power and reducing area. The operation of this basic gate is synchronised by the Request signal which distinguishes between the two, precharge and evaluation phases of the Request cycle.

The operation is as follows:

Precharge Cycle: Request signal is high, T_1 is conducting and commences to charge the internal node, unless the input is high. Because of the forward conduction of the gate-to-source diode in the NOR gate, the voltage in the internal node is limited to around 0.7V.

Evaluation Phase: Request signal is low. The output of the NOR gate evaluates to logic 0 or logic 1 depending upon if the input level was at logic 0 or logic 1 respectively during the precharge phase.

In such a system, although inputs cannot be changed during precharge phase, when we connect or cascade this circuit with other self-timed components, the request signals for two cascaded circuits should have non-overlapping request signals. (Lopez et al, Paper [7])

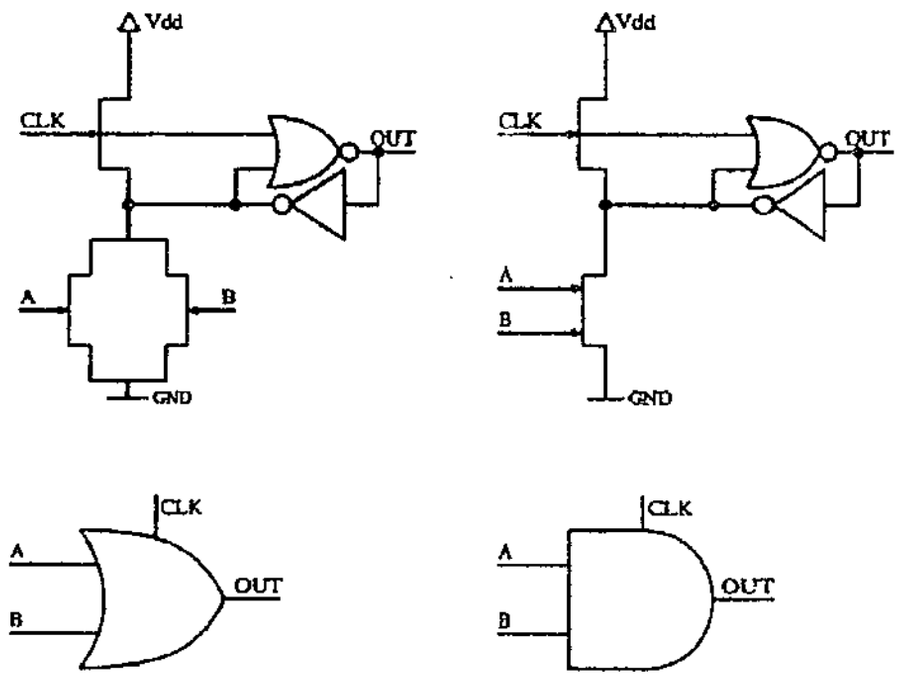


Figure 2.4: AND and OR Gates Implemented in PDLL.
Adopted from Lopez et al

2.3.3 PDLL Trial Simulations

Simulations were performed with HSPICE. The figures depicted here are for the basic gate simulated using Vitesse H-GaAs III technology, which has a minimum length of 0.6μm and

makes use of enhancement and depletion transistors. These devices have threshold voltages of 0.21V and –0.8V for E-MESFET and D-MESFET respectively. The simulations have been carried out at 25°C and 100°C at typical, 2σ slow-slow and 1.5σ fast-fast process. The clock frequency was 2GHz. As it can be seen from the simulations in figure 2.5, the request, input and output signals are totally compatible with DCFL levels. (Lopez et al: Paper [7])

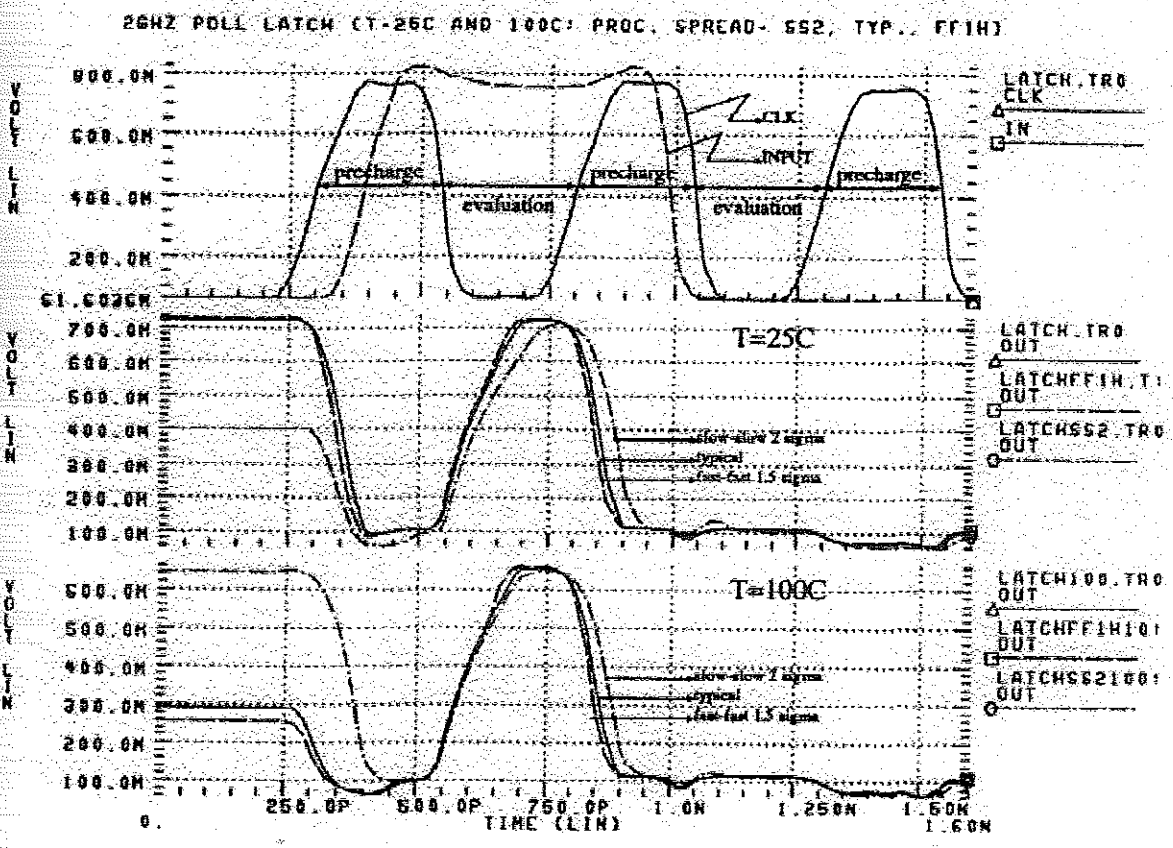


Figure 2.5: PDLL Latch Simulation. Adopted from Lopez et al

2.3.4 Power Dissipation

PDLL gates are composed of two terms, which contribute to the total power consumption. Firstly, there is a dynamic contribution due to the clock signal, precharge transistor and logic function. Secondly, there is a static contribution due to the static latch. Thus the total power dissipation can be expressed as:

$$P = P_{DYNAMIC} + P_{STATIC}$$

Figure 2.6 shows a transient analysis for the power consumption of the latch simulated in figure 2.5 at 25°C and typical parameters, where static and dynamic contributions have been drawn separately.

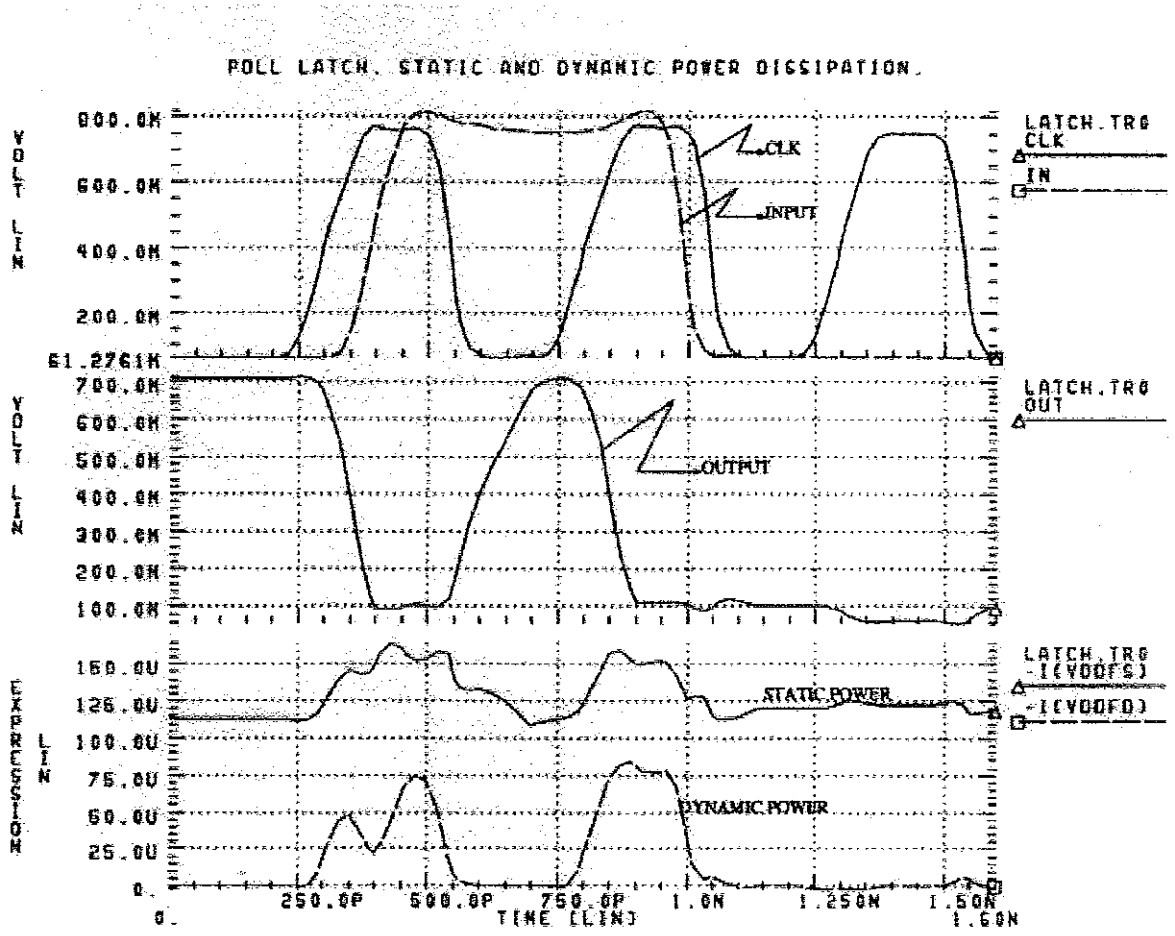


Figure 2.6: PDLL latch static and dynamic power dissipation. Adopted from Lopez et al

A 1V power supply was used, giving as a result an average static and dynamic power dissipation of 128μW and 31μW respectively. This gives a total average power of 150μW. In terms of the power/megahertz gate, this power corresponds to only 75nW/MHz. Generally, the static latch at the output of the PDLL gate is implemented in DCFL which is dominated by static power because the active load (pull-up) current constantly flows to the ground, either through the switch transistor or through the gate of the next stage. However, decreasing the pull-up current can reduce this power dissipation. But this fact leads to high logic level degradation as well as to low-to-high transitions at the output, hence decreasing noise margins. On the other hand, permanently restoring action of the latch makes possible

the reduction of the pull-up current by decreasing transistor widths. This fact optimises power and area. As a side-effect, though, delays in the circuit are increased. The dynamic block can also be reduced in size at the expense of increasing delays in the precharge cycle. Thus, a tradeoff between area-power-delay has to be chosen depending upon the required performance and application.

To compare the characteristic of PDLL logic family with other GaAs static and Dynamic logic families, an adder with identical functionality was implemented in each of the logic families. The comparison is shown in table 2.1, where a figure of merit, η, is introduced to represent the performance in terms of *frequency/area.power*.

Logic Family	Area (mm^2)	Fmax (MHz)	Power (mW)	η (MHz/mm^2.mW)
DCFL	0.32	714	47	47
BFL	0.75	500	190	3
CCDL	0.70	900	96	13
TTDL	0.55	1250	130	17
DPTL	0.35	1000	228	13
TDFL	0.16	770	1.7	2831
SPDL	0.30	1520	40	327
PDLL	0.05	833	5.2	3204

Table 2.1: Performance Comparisons for several GaAs logic families. Adopted from Lopez et al (Paper [7])

It can be seen that in terms of power dissipation, TDFL is by far the best choice. However, due to the low noise margin levels and the use of pass transistors, its operation is critical at high temperature because of leakage current problems. On the other hand, SPDL presents as its main characteristic high operating frequency, but its power dissipation and area consumption, close to DCFL, makes its figure of merit η to be in a low range. PDLL is located in between these two solutions, offering an extremely low area consumption with low power dissipation and moderate operating frequency. Compared to the rest of the solutions, it gives the highest figure of merit and permits a wide range of temperatures and frequencies. The logic families BFL, CCDL, TTDL and DTPL are not suitable for VLSI implementation because of their extremely high power dissipation.

2.4 Modified Ring Notation Approach

For CMOS technology, there are tools such as COMPASS, OCTTOOLS which have functions to generate mask layouts from schematics automatically. However, there are no such tools available for GaAs technology, mainly because that layout style is a critical design parameter for high speed systems.

A Modified Ring Notation (MRN) approach is presented in this section which translates a circuit schematic into mask layout suitable for use in layouts using the Vitesse 0.8μm GaAs IC foundry. However, the same idea can be used for other GaAs processes and foundries.

Various primitives have been implemented using different layout styles and compared. The MRN approach proves to be the best one to reduce coupling and to improve GaAs technology layout density. The HSPICE simulated results show that the combination of the fast arithmetic architecture and the new layout style makes MRN circuitry faster, more compact and less power-dissipating. (Cui, 1996)

2.4.1 The Original Ring Notation Approach

This layout style has a major impact on the performance of very high speed VLSI Ics. The main issues of concern are:

1. To minimise interconnect lengths to reduce parasitic capacitance so that the coupling between high speed signals can be minimised.
2. To reduce the inductance and increase the capacitance associated with the power buses so that voltage and current spikes can be reduced.
3. To achieve high layout density.
4. Pitchmatching of basic blocks.

Eshraghian(1993) introduced *Ring Notation* approach explicitly to try to meet the above requirements. In the Ring notation approach, the GND (Ground) rail is placed between the logic and the power supply Vdd (High) rail as compared to the nMOS style layout, as shown in figure 2.7. This reduces the self-inductance of the buses and hence increases immunity to noise induced by signal crosstalk and switching spikes.

Ring notation also provides an intermediate method to describe the layout of a circuit design. In Ring notation, as shown in figure 2.8, the 'dashed' line represents the E-MESFET while the 'solid' line represents the D-MESFET. The two MESFETs are joined together

using metal which is implicit in the ring representation for simplicity. However, it should be noted that the missing geometry will appear when the ring notation is translated into mask layout as shown in figure 2.9

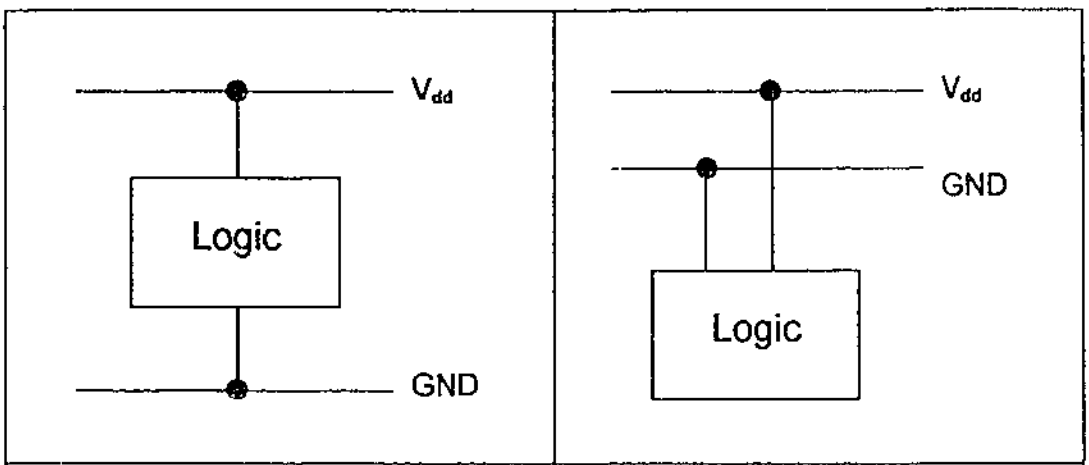


Figure 2.7: nMOS Style vs Ring Notation Style of Layout

It can be seen from figures 2.6 and 2.7 that the translation from a circuit to the layout is straightforward using ring notation approach. Ring notation allows paths to be highlighted and the interconnection strategy to be formulated before the design proceeds to layout. Complex structures can be readily and systematically mapped thus providing an easy translation method from symbolic notation to mask layout and hence ensuring the portability of designs.

	Original Ring Notation				Modified Ring Notation			
	Delay (ps)		Area(hm2)	power (mW)	delay (ps)		area(hm2)	Power (mW)
nor2	60.0		569.6	0.25	63.0		432.0	0.250
nor3	81.8		801.6	0.287	89.7		541.0	0.278
full adder	Co	282	1452.8	3.467	Co	161	4720	3.467
	Sum	462			sum	350		
5 x 1 Mux	371		7873.6	3.03	282		6205.7	3.03

Table 2.2. Performance of MRN for basic logic blocks. Adopted from Cui (1996).

From the designers' point of view, ring notation methodology is easier to implement that the nMOS style, allowing a straightforward approach to circuit design. Eshraghian (1993) also showed that ring notation technique produces circuits with higher speeds than the nMOS style technique, mainly because of the shorter interconnection length of the path. In this sense, the delay incurred by overcrossing capacitance effects is less in ring style layouts.

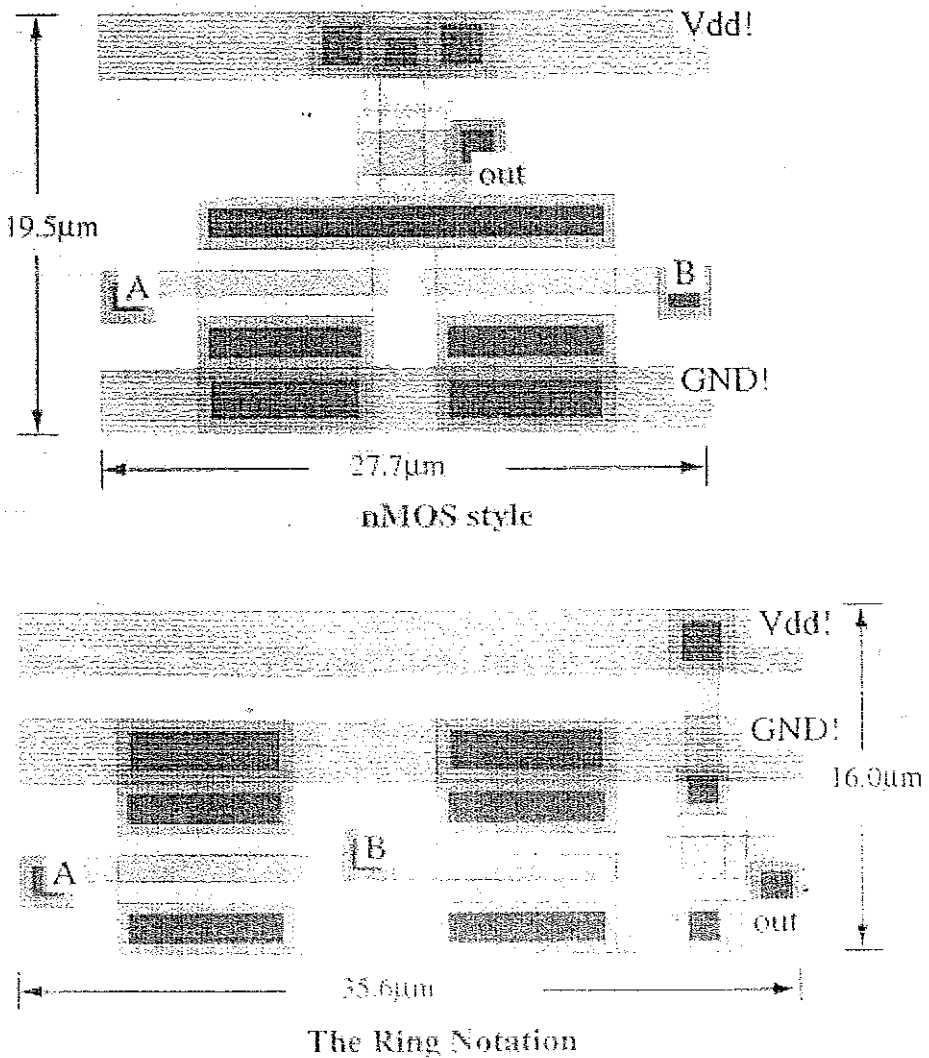


Figure 2.8 Comparison of design styles for 2 input nor gate.

For the same circuit, ring notation layouts occupy a little more area than nMOS style layouts, as can be seen from table 2.2. Its is also apparent from figure 2.8, which shows a nMOS style two input nor gate side by side with a ring notation two input nor gate. In this

case, the ring style takes 5% more area than the nMOS style. This situation will worsen for larger circuits, because when the circuit is large and complex, the ring style layout has to use several Vdd/GND bus lines instead of extended transistors along one Vdd/GND bus line. The extra Vdd/GND bus lines occupy extra area (Cui, 1996).

Thus, in order to maintain the advantage of ring notation and at the same time reduce the layout area, a *modified ring notation* (MRN) has been developed and will be described in the following section.

2.4.2 The New Modified Ring Notation Approach

The MRN uses the same idea as 'ring notation' or 'ring diagram'. But when translating to the mask layout, the MRN stacks transistors vertically instead of horizontally along the bus lines. The layout is more compact, because E-MESFET transistors can share connected vias and no horizontal spacing is required between the transistors.

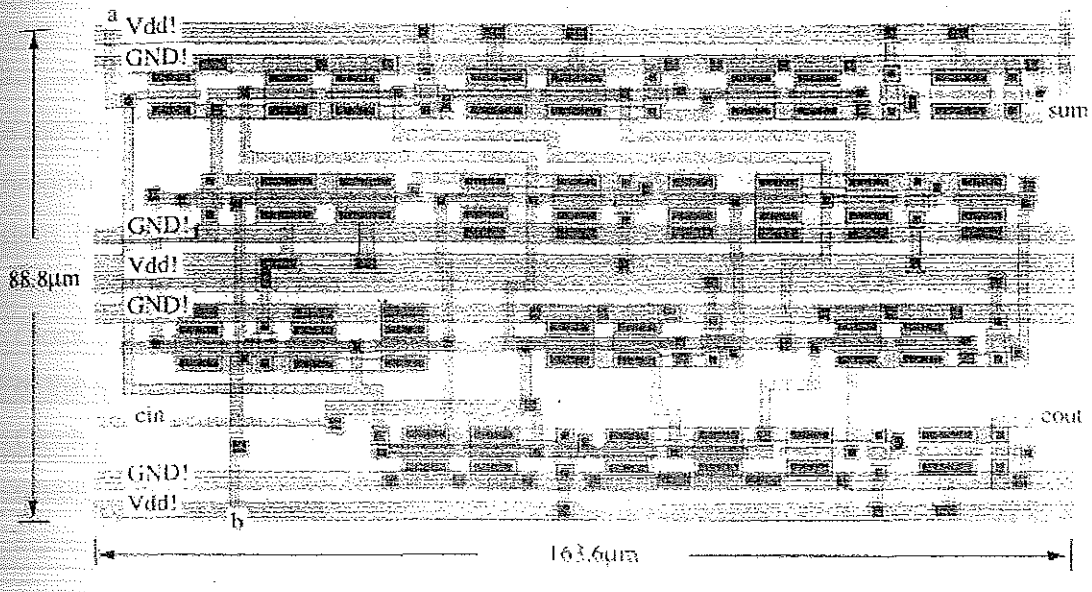
2.4.2.1 The Improvement in Layout Area

For complex circuits, the advantage of MRN over original ring notation in terms of area is more obvious. Figure 2.9 shows a full adder implemented by the two ring styles at the same scale. Because the original ring style lays out the transistors horizontally along the Vdd/GND bus lines, it has to use three bus lines to make the layout the desired shape. The extra bus lines and the space between lines occupy a lot more area. In contrast, MRN style full adder uses only one Vdd/GND bus line; the transistors are stacked vertically along the bus line. Thus there is no wasted space in the layout. The MRN approach reduces the full adder area by a factor of 3, which is very impressive (Cui, 1996).

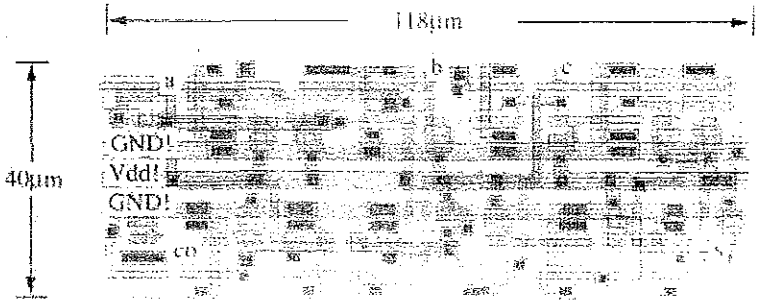
2.4.2.2 The Improvement in Speed

The MRN not only improves the layout area, but also the speed of the circuit. Table 2.2 shows the simulated performance of some basic logic cells implemented by these two ring notations with fan-out = 1. It can be seen that when the circuit is as simple as nor gates, the area of MRN is improved, but delay is increased. This is because using shared vias reduces the area of the vias, hence the resistance is increased. Consequently, the current through the transistor is decreased, which reduces the speed of the circuit. However, when the circuit is more complex, such as a full adder, the MRN also has a significant

improvement in terms of speed. It can be seen that the MRN approach increases the speed of the full adder by a factor of 1.31-1.75. This is because the more compact layout reduces the length of connections and therefore the capacitive loads.



(a) A full adder using original ring style.



(b) A full adder using modified ring style.

Figure 2.9: Full adder implemented in original style and MRN. Adopted from Cui (1996)

2.5 The Self-Timed Design Approach

A self-timed circuit, as the name suggests, regulates its own signal timing and is independent of any external timing control signals, like the clock.

2.5.1 Self-timed (Asynchronous) versus Clocked (Synchronous) Circuits

The most popular design technique in today's VLSI systems is the synchronous approach, Eshraghian (1993), where a global timing signal coordinates the movement of data in the system. However, there are some unique problems concerning this:

- Clock distribution is not easy since the synchronising clock signal affects nearly all parts of the system and it has an unusually large fan-out.
- Clock-skew (the difference the times when a clock-edge appears at different parts in the circuit) increases to significant proportions in high-speed circuits, including clock signals.
- Since current transients on power and ground busses are synchronised with clock-edges, inductive voltage spikes can be severe. (Lachowicz et al: IEEE transaction [1])
- Clock speed must be selected to accommodate the worst-case conditions associated with both, the process technology and data-path delays for the circuits employed.
- To make the circuit robust, a detailed timing analysis and calculation of worst-case timings becomes essential. Various factors like the design style, leakage effects, parasitic capacitance, environmental noise etc must be determined accurately and be accomodated in circuit design and clock speed.

In the realm of VLSI, exploiting concurrency is a prerequisite to high-performance: as systems become larger and more complex, one can no longer afford to ignore the parallelism in control operations. The central control (clock) style creates difficulties in high-performance systems by imposing an unnecessary sequential order on the execution of control operations. In choosing a control style which allows for parallel execution of unrelated operations, we naturally move towards numerous distributed control modules which can operate concurrently. (Chu, 1987)

A self-timed or speed independent circuit is one whose correct operation is independent of the delays of logic gates composing the circuits. One immediate consequence of this property is that speed-independent circuits are always hazard-free. These types of circuits

are desirable in VLSI systems because it is usually difficult to fine-tune the delays of logic gates to make an asynchronous digital circuit work properly. Perhaps most importantly, the use of speed-independent circuits enables the separation of the correctness of systems from timing considerations, which inextricably depends on many physical factors and phenomena in VLSI circuits. It is no coincidence that a number of research efforts in silicon compilation have utilised speed-independent circuits as a basis for hardware implementation.

GaAs technology is a good candidate for very high speed systems. However, as mentioned above, this makes global clock distribution in synchronous systems a difficult task. An alternative to this approach, self-timed logic design has increasingly obtained new adaptations to justify its use. Self-timed circuits work on asynchronous principles at a rate determined by their internal construction and interconnection rather than the speed of the applied clock. Two prevalent concepts describing the properties of self-timed systems are delay-insensitivity and speed-independence. (Lachowicz et al, Paper [6]). While these are commonly used interchangeably, we often find that delay-insensitivity has connoted a stress on the communication aspect and hence, the composition of systems. On the other hand, speed independence is usually understood as a property of control circuits, which operate correctly regardless of variation in delays of logic gates. Thanks to their delay insensitivity, self-timed GaAs designs are found to be very robust, operating correctly over a wide range of supply voltage with corresponding changes in performance.

2.5.2 History of Self-Timed Systems: Formerly Used Techniques

Self-timed systems have been a subject of research since 1960's, various approaches have emerged to be able to form self-timed system design from specifications provided. Research works in self-timed systems can be categorised according to two attributes: the theoretical models on which the research is based, and the particular aspects chosen for study. The two major models for self-timed systems are

1. *finite automata* and
2. *Petri nets.*

With regards to aspects of study, the two chief areas of concern are composition of systems from modular descriptions and Synthesis of modules from specifications.

2.5.2.1 Petri Nets

Petri Nets is a formal and graphically appealing language, which is appropriate for modeling systems with concurrency. Petri Nets has been under development since the beginning of the 1960, where C.A. Petri defined the language. It was the first time a general theory for discrete parallel systems was formulated. The language is a generalization of automata theory such that the concept of concurrently occurring events can be expressed. (Chu, 1987).

Coloured Petri Nets: Petri Nets has proven to be a useful language for describing concurrency. However, in its original form, the language is inappropriate for describing more complex systems. Thus many research activities have been directed towards making Petri Nets more suitable for describing complex systems. These activities have resulted in the development of the so called high-level Petri Nets. They have structuring facilities that are suitable for describing complex systems in praxis. Since late in the 70's Kurt Jensen has worked in this research area and developed the well-known Coloured Petri Nets (CP-nets or CPN). Like Petri Nets, CP-nets has a formal definition allowing the popular analysis methods from Petri Nets to be adopted in CP-nets.

Today CP-nets are widely used for modeling many different kinds of systems with concurrency. One of the most applied areas of CP-nets is protocol especially within communication systems. Another area is hardware design. A different approach to modeling of systems is to describe the system in a well-known language other than CP-nets, and then translate the model into a CP-net for further analysis. The long list of references to papers about modelling projects using CP-nets indicates the increasing popularity of the language. (Mortensen, 1998).

With regards to synthesis of actual circuits based on model provided by Petri nets, one of the most important contributions was made by Patil and Dennis at MIT. Patil invented asynchronous logic arrays as a systematic way of directly implementing Petri nets to form circuit designs. In asynchronous logic arrays, columns of wires are connected to storage elements to simulate the places of a net, while rows of wires decode the state of the columns to simulate occurrence of transitions. Thus, this method of implementation transfers the logical structure of the language Petri nets directly to circuit design and consequently to hardware level. Dennis has shown that Petri nets can be used to model asynchronous hardware systems at many levels of description in a very clear and easily understood manner. His description of the control logic of the Control Data 6000 computer,

which embodied instruction look-ahead and interleaved execution, demonstrated this technique. Dennis and Patil also elucidated an organisation principle for asynchronous hardware systems in which a system is partitioned into data paths and distributed control structures, the latter organised as asynchronous modules which communicate with each other using certain signaling protocols.

2.5.2.2 Finite Automata: Finite State Machine

The State Machine, or more accurately, Finite State Machine (FSM), is a device and a technique that allows simple and accurate design of sequential logic and control functions. They can be used in computer programs, sequential logic circuits or electronic control systems. The idea behind the FSM is that a system such as a machine with electronic controls can only be in a limited (finite) number of states.

A FSM is a machine that takes in a word which it either accepts or rejects. We can think of the machine as answering the question: Is this word in your language? The language of an FSM is the collection of all those words that it accepts, and none of the words that it rejects. If the word pattern matches the required, a true is output. The operation of an FSM is based on a state diagram. An example state diagram is shown below in figure 2.10. It has circles showing various states. The transition from one state to another can be made using signals, which are shown by arrows. (MIT, 1998)

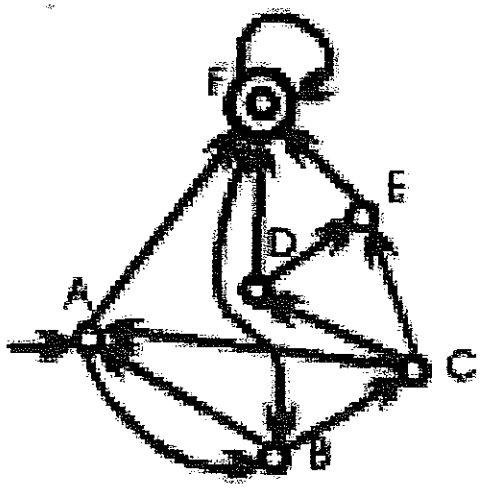


Figure 2.10: Example FSM Chart

The notable characteristic of an FSM is that it is relatively easy to form state diagrams for complex systems from given specifications. These state diagrams can then be implemented into FSMs using conventional logic gates and latches.

FSMs have many applications. They are used for pattern recognition or string searching, to simulate logic required for various systems like CD players, vending machines etc. In logic design, they can used to construct systems where the outputs depend on previous states of the system and other factors. They are used to simplify the logic involved with analysing multiple sequential inputs and related actions for a logic system.

2.5.2.3 Disadvantages of FSM

Despite being a very useful tool in logic design, FSMs have limitations when implemented in self-timed systems. (Chu, 1987). Some of frequently cited disadvantages are

- A FSM cannot handle unrestricted input changes. For an FSM, if two input transitions occur within a time interval (i) less than $\min(D_{rise} + D_{fall})$, then they are considered simultaneous, and if (ii) greater than $\max(D_{rise} + D_{fall})$, then they are considered separate. However, if the separation between the inputs is between these two values, the circuit malfunctions.
- The FSM model cannot describe concurrent behaviour directly. The FSM model and the Huffman state machine are based on the use of central states. At any moment, the machine resides in one state and it reacts to input excitations in different ways depending on which state it is in. A serious drawback of his state-based approach is that it is incapable of describing concurrency directly.
- The state assignment problem. Since it is difficult to match gate delays to achieve simultaneous transition of state signals, one has to make sure that simulataneous changes in state signals do not occur, of if they do, the circuit must be designed such that it behaves the same no matter which sequences of state changes take place. Hence most state assignment techniques only allow at most one signal change between states. This is a well-known hard problem for which many heuristic techniques have been proposed.
- The number of entries in the flow table of a state diagram depends exponentially on the number of input signals. Due to the absence of a controlling signal (clock), an asynchronous state machine continuously senses the changes in the input and produces changes at the output and the state variables. Therefore, in contrast to a

synchronous implementation of FSM, the self-timed implemetation requires the listing of all input combinations in the flow table, resulting in the exponential increase in the number of entries in the table.

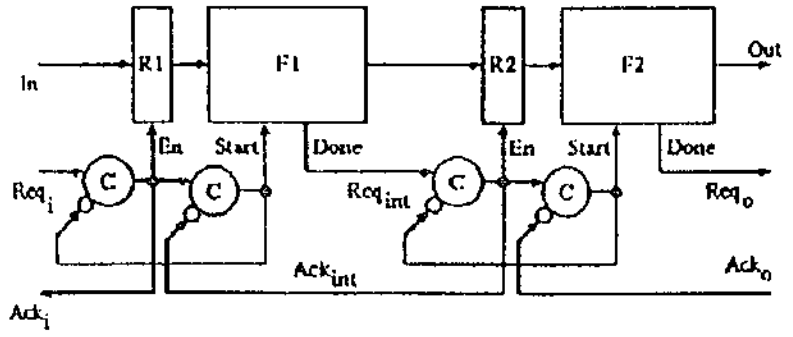
2.5.3 Self-timed Techniques for the Filter Design

The self-timed approach has been chosen for filter design in order to eliminate the need for global distribution of extremely high frequency clock signals with the expected benefits of reduced power dissipation and inherent delay insensitivity.

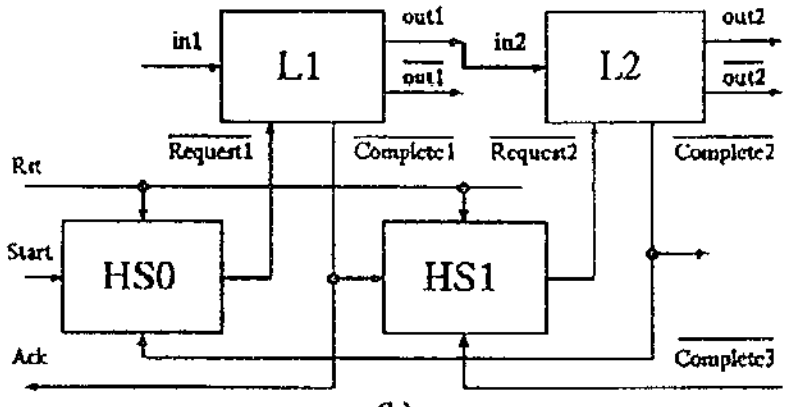
The self-timing technique uses a model where by various logic blocks are triggered to execution by a request signal, and in turn output a complete signal upon achieving correct output. An enable signal controls the execution of a logic cell.

Self-timed systems require that the logic cells have several control inputs and that they generate at least one control signal for handshaking. For the typical 4-phase handshaking protocol the input signals are *Enable* and *Complete* as shown in figure 2.11. The *Done* signal triggers the *request* input in the next stage's handshaking block. The logic path consists of register latching the input signals and the functional block implementing the logic function. (Lachowicz et al, Paper [6])

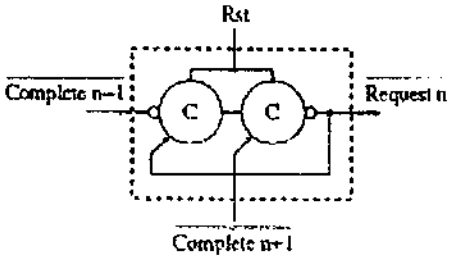
As discussed before, the PDLL and LCFL logic families are geared towards designing self-timed circuits with ease. Figure 2.12 shows basic PDLL and LCFL cells for self-timed applications. As can be seen, the latch is an inherent component of the cell. This property together with the appropriate modifications the handshake path to suit the GaAs latched logic design style can be utilised to eliminate the separate latch blocks from the pipeline. The modified pipeline is shown in figure 2.11. In the LCFL cell of figure figure 2.12, the *Complete'* signal is generated by first producing the complement of the output with a NOR gate which is also controlled by the *Request'* signal. This NOT gate is sized appropriately to achieve equal signal delay at the input of the following NOR gate producing the *Complete* signal.



(a)



(b)



(c)

Figure 2.11 Self-timed pipelined datapath:
(a) Standard four-phase protocol,
(b) A pipeline modified for LCFL,
(c) Details of the handshake block.

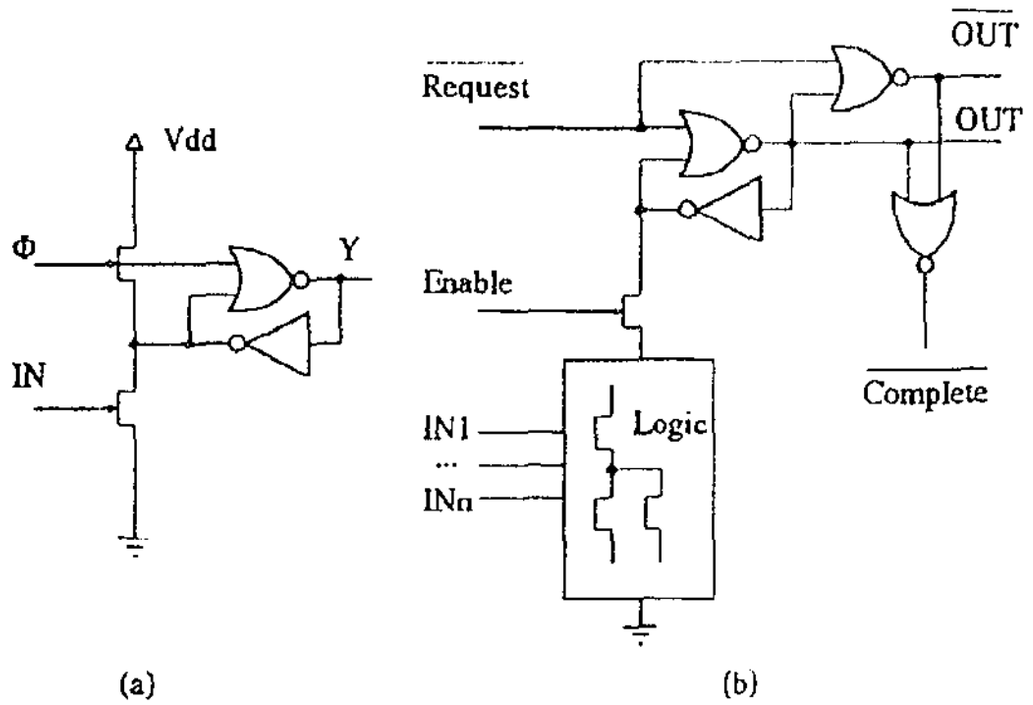


Figure 2.12: GaAs Latched Logic Gates: PDLL and LCFL cells. Adopted from Lachowicz et al

2.5.3.1 Operation of the Self-Timed Logic Cell

The logic cell operates as follows: When the request line is high the cell is in the reset stage, both lines *Out* and *Out'* are low and the *Complete'* is high. When Request' goes low and *Enable* is high, the cell evaluates the output and one of the lines *Out* or *Out'* conditionally goes high, which causes the *Complete'* line to go low and this indicates that logic evaluation has been completed. The *Enable* line is not always needed, and in the pipeline structure from figure 2.11 the logic cells do not contain the *Enable* line. There are cases, however, when *Enable* line cannot be omitted, as will be later demonstrated.

The modification of handshake path as compared to the standard four-phase handshake protocol is such that a particular LCFL logic cell is not allowed to enter into the reset state until the following cell completes its evaluation, and also it cannot perform the next evaluation until the cell following it enters the reset state. This handshake protocol operation removes the need for the separate latches between the logic gates.

3 Video Compression and Wavelet Transforms

3.1 Introduction

Analog video communications technology has been around for a few decades and had significantly affected the lives of millions of people worldwide. The television and VCRs have become very common consumer electronic items.

Digital image and video offers many advantages over analog video for the vast majority of applications. Therefore, as digital video technology becomes more mature, we are witnessing a shift from analog to digital technology. The ease with which digital video can be integrated with computers, together with the introduction of global communication networks and devices that support the integration of video with audio and conventional computer data make a whole range of new applications possible. (Ozer, 1994)

The most important issue in digital communications, however, is the fact that digital representation of an image or of a sequence of images requires a very large number of bits. The goal of image coding is to reduce this number as much as possible, and to reconstruct a faithful duplicate of the original picture.

Wavelet transform of an image is one of the very efficient image compression techniques. There are various types of wavelet transforms. The transform used is this project is a simple 3-tap triangular filter-based transport.

3.2 Digital Video

Digital video is video information that is stored and transmitted in video format. However, until recently a lot of factors have prevented the wide-spread use of the digital video, including a large amount of bandwidth required. The solution is to compress the digital video information and to store and transmit it in a compressed form.

Compression techniques for digital video have been continuously improving for the last two decades. International standards now provide standard techniques for digital video coding. At the same time, processor technology has improved dramatically over the recent years.

The availability of cheap, high-performance processors together with the development of international standards for video compression have enabled a wide range of video communications applications.

The standards in video coding make use of the redundancy inherent within digital video information in order to achieve a substantial reduction in bit rate. A still image, for a single frame within a video sequence contains a significant amount of *spatial* redundancy. (Richardson and Riley, 1996) It is possible to represent or encode the information in a more compact form that eliminates some of this redundancy. For example, the image data may be transformed from the spatial domain into another domain in which the information is represented more compactly. Certain components of the transformed information can be eliminated without seriously reducing the visual quality of the decoded image. The information can then be efficiently encoded using entropy encoding (for example, variable length encoding such as Huffman coding).

A moving video sequence contains *temporal* redundancy ie. Successive frames of video are usually very similar. It is possible to achieve further compression by sending only the parts of the image which have been changed from the previous frame. In most cases, changes between frames are due to movement in the scene, which can then be approximated as a simple linear motion. By predicting the motion of regions from the previous transmitted frame (motion prediction) and then sending only the prediction error, the transmitted data can be reduced further.

3.2.1 The Need for Compression

A single digital television signal in standard format requires a transmission rate of 256 Mbps, which is unacceptably high for most practical purposes. An analog television signal of comparable visual quality occupies a bandwidth of around 6 to 7 MHz. Digital television in this format therefore compares unfavourably with its analog counterpart, in terms of transmission requirements. This bit rate is too high for most existing communication networks. Most local area networks (LANs, for example, offer data transmission at rates of up to a few tens of megabits per second, and most wide area networks (WANs) support much lower data rates than this. Some of the ATM networks are capable of transmitting at higher bit rates. However, distribution a general video bit stream over these networks would be prohibitively expensive. (Richardson and Riley, 1996)

Thus there is a need to reduce this data rate before digital video communication can be integrated to existing and emerging communication networks. Fortunately, digital video data contains a considerable amount of redundancy. Some compression can be achieved by exploiting the statistical redundancy within the data. For example, video data is often highly correlated both spatially and temporally. This redundancy can be removed by coding the data in a more efficient way. Compression of this nature does not destroy any of the original data and is hence a reversible process (lossless compression). In order to achieve higher levels of compression, it is necessary to remove *subjectively redundant* information. This information is not usually obvious to the viewer and can be removed without severely reducing the quality of the signal. This type of compression destroys some of the original image information, which cannot later be recovered. (lossy compression).

3.3 Image Coding Techniques

A typical photographic quality still image contains a large amount of spatial redundancy. The pixel values are often highly correlated. The redundancy (both statistical and subjective) can be removed to achieve compression of the image data. By removing this redundant information and by representing the data in a more efficient form, the capacity required to store or transmit the image can be greatly reduced

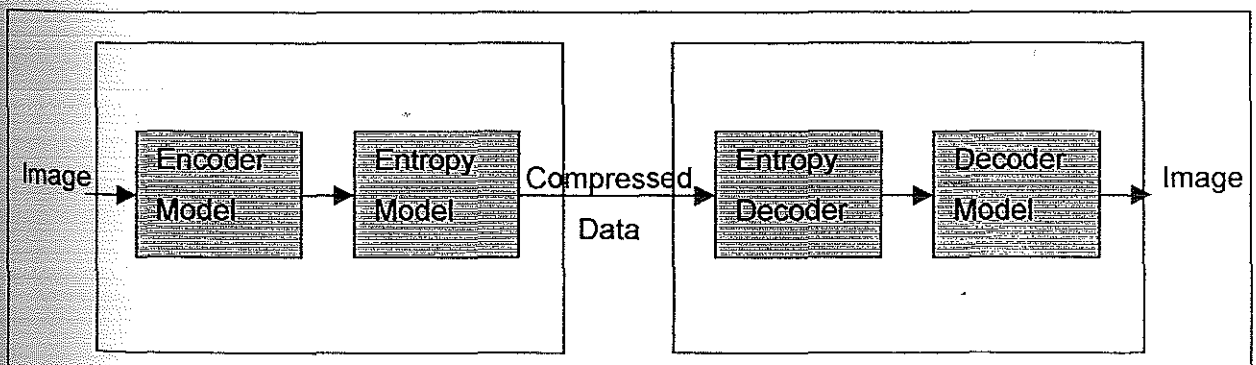


Figure 3.1 Image Compression. Adopted from Riley and Richardson (1996)

The general procedure for compression image information is shown in Figure 3.1 The *encoder model* models the image in some way to exploit its statistical properties and to remove redundancy. The encoder produces symbols that represent the information in the original image. These symbols are then entropy coded (for example using Huffman coding) to code them as efficiently as possible. The decoder carries out the reverse procedure to recreate a copy of the original image. The following image compression techniques were derived from Richardson and Riley (1996) and developed further using other sources.

3.3.1 Predictive Coding

One of the simplest image compression techniques is the differential pulse code modulation (DPCM). The image to be transmitted is scanned in a raster fashion (i.e., the pixels within the image are ordered in lines from left to right and from top to bottom).

Each pixel is represented as a number with a limited precision (eg. 8 bits). In a DPCM system, the pixel value itself is not transmitted. Instead, a prediction of the probable pixel value is made by the encoder based on previously transmitted pixel values. The prediction error between the predicted pixel value and the actual value is quantised and transmitted. This system takes advantage of the fact that most images contain significant amounts of spatial redundancy. Because of this, neighbouring pixels tend to be highly correlated and prediction error tends to be small. Further compression can be achieved by encoding the quantised error values using variable length codes (entropy coding). Since small errors are likely to occur more frequently than large errors, the more common values can be encoded using a shorter code and the less common values using a longer code.

The decoder must use the same prediction method as the encoder. The prediction, based on previous reconstructed pixels, is added to the error value to reconstruct the current pixel.

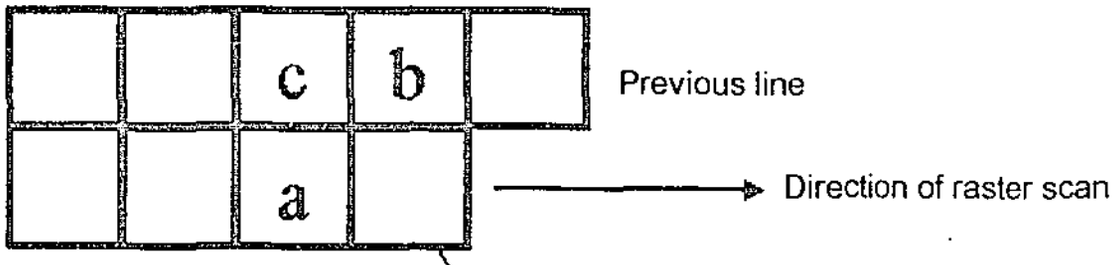


Figure 3.2 Image Compression

The coding efficiency of this system depends on the accuracy of the prediction. The simplest predictor is the previous transmitted pixel value. In the example shown in figure 3.2, the current pixel would be predicted from the value of pixel a. A slightly better prediction can be made by averaging the values of other neighbouring pixels, for example a,b and c in figure 3.2

Better predications still can be made by taking more pixels values into account. The choice of the predictor and the weights of the neighbouring pixel values have a significant effect on the efficiency of the algorithm. Further improvements can be gained by modifying the predictor based on the statistics of the current image.

3.3.2 Discrete Cosine Transform Coding

Understanding of wavelet transform based compression is facilitated by the study of Discrete Cosine Transform (DCT). Thus a study of DCT technique and its application shall create a good foundation for introduction of the wavelet transform and related compression.

Transform coding is a very widely used method of compressing image information. In a transform coding system, pixels are grouped into blocks. A block of pixels is transformed into another domain to produce a set of transform coefficients, which are then coded and transmitted.

An effective transform will *compact* the energy in the block of pixels. This means that the useful information in the block will be concentrated into a few of the coefficients. Compression is achieved by quantising the coefficients so that the "useful" coefficients are transmitted and the remaining coefficients are discarded.

The DCT is a popular transform based coding technique. The popular JPEG image format uses DCT compression. (Ozer, 1994). For most continuous-tone images, the DCT provides energy compaction that is close to optimum. A number of fast algorithms exist for calculating the DCT of a block of pixels. These factors have led to the use of DCT for image and video compression systems.

An image coding system based on DCT shall be set out in the following steps:

Separation of Image into Blocks

An image is made up of a rectangular array of pixel values, which are grouped into blocks. Most existing systems use blocks of a regular size, such as 8 x 8 or 16 x 16 pixels. A larger block size leads to more efficient coding, but requires more computation power.

This block-based segmentation of the source image is a fundamental limitation of the DCT based system. Real-world images do not generally contain features that can be neatly partitioned into regular blocks. The use of uniformly sized blocks amplifies the coding system, but does not take into account the irregular shapes within real images. It is possible to achieve better compression efficiency by using a combination of blocks of different shapes to cover the image, at the expense of increased complexity.

The DCT

The DCT converts a block of pixels into a block of transform coefficients of the same dimensions. These coefficients represent the spatial frequency components that make up an appropriate basis function. The basis functions for an 8 x 8 DCT are shown in figure 3.3. the top-left function is the basis function of the 'dc' coefficient and represents zero spatial frequency. Along the top row, the basis functions have increasing horizontal spatial frequency content. Down the left column, the functions have increasing vertical spatial frequency content, and along the diagonal the functions have combination of horizontal and vertical spatial frequencies. All of the coefficients other than the DC coefficient are known as "ac" coefficients.

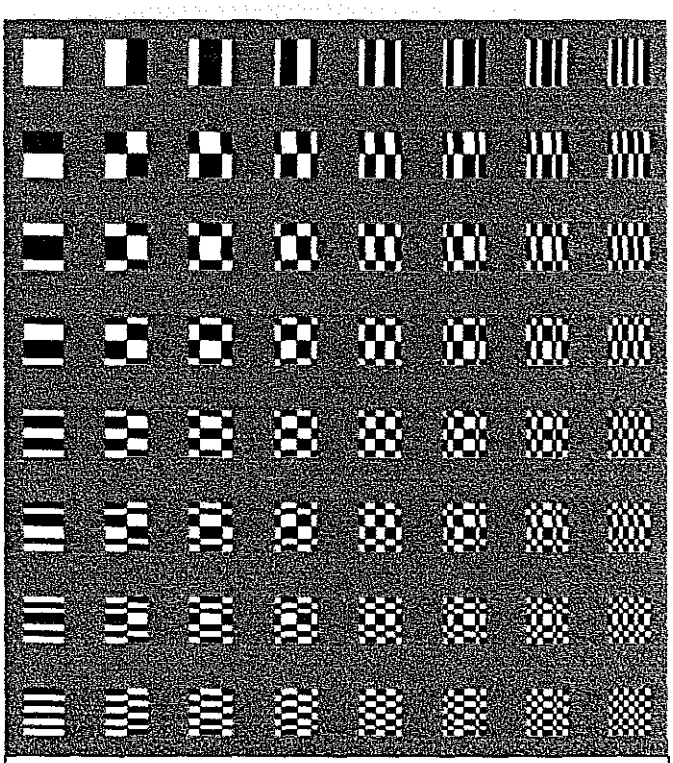


Figure 3.3: DCT basis functions. Adopted from Robert Gordon University (1996).

Any grey-scale 8x8 pixel block can be fully represented by a weighted sum these 64 basis functions. The appropriate weights that are required to produce a particular block are the DCT coefficients for that block. The DCT operation therefore transforms a block of pixels into the set of DCT coefficients that represent the block in the spatial frequency domain. This in itself does not give compression since the information is merely being represented in a different form.

The two dimensional DCT of an NxN block of pixels values is being described by the following formula, where the pixel values are represented as f(I,j) and the transform coefficients are as F(u,v)

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j) \cos\left(\frac{(2i+1)u\pi}{2N}\right) \cos\left(\frac{(2j+1)v\pi}{2N}\right)$$

where $C(x) = \begin{cases} \frac{1}{\sqrt{2}}, & x = 0 \\ 1 \end{cases}$

The inverse DCT of an NxN block of coefficients is given by:

$$f(i,j) = \frac{2}{N} \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\left(\frac{(2i+1)u\pi}{2N}\right) \cos\left(\frac{(2j+1)v\pi}{2N}\right)$$

with C(x) as above.

It can be seen that both forward and inverse transforms require a large number of calculations; each coefficient in the forward transform is calculated by summing 64 separate calculatoins, and so a total of 64x64 = 4096 calculations are needed to transform and 8x8 image.

Forward and Inverse DCTs are separable. Instead of performing a 2-D transform based on the above equations, the same result can be achieved by applying a one-dimensional transform along all the rows of the block and then down all the columns of the block. This reduces the number of calculations required. Each coefficient in the one dimensional tranform requries 8 calculations and so a row or column requires 8x8 = 64 calculations. The

total number calculations is thus 64 x 8 x 2 for the rows and columns, making it 1024. This reduces the computational time.

Quantisation

For a typical block within a photographic image, most coefficients will be clustered near 0. On average, the larger DCT coefficients will be clustered around the dc value, meaning they will be low spatial frequency coefficients. Most blocks within an image tend to contain a little high frequency content, so in general only a few of the DCT coefficients for a block will have significant values.

The DCT coefficients are quantised so that the near-zero coefficients are set to zero and the remaining coefficients are represented with a reduce precision. This can be achieved by dividing each coefficient by a positive integer and rounding the result to the nearest integer. The larger the quantiser scale, the coarser the quantisation. This results in loss of information, but also in compression since most of the values in each block become zero. Increasing the quantiser scale leads to coarser quantisation, which gives high compression and poor decoded image quality. Reducing the quantiser scale has the opposite effect.

DCT-based image coding systems can provide compression of between 10 and 20 times while maintaining reasonably good image quality. The compression efficiency depends to some extent on the content of the image: an image with lots of detail will contain many high-frequency components and will produce more coded data than a similarly sized image with less detail. Compression can be increased by increasing the quantisation scale factor. In general, higher compression can be achieved at the expense of poorer decoded quality. An illustration of this fact is an example of images decoded after performing a DCT on them with different coding levels.

3.3.3 Fractal Coding

Fractal based coding is not very relevant to wavelet transform based compression (described for completeness of discussion), so they shall be described here very briefly.

Fractal coding is one of the "second generation" coding techniques, and can provide very high compression with little loss of image quality. (Torres and Kunt, 1996) The image is described in terms of a *fractal* code, a series of transformations which, when applied to an arbitrary source image, can be used to recreate the original image. The decoder can reconstruct the image using the information contained within the fractal code. This technique has been reported as giving very high compression ratios. However, the process

of finding the optimum fractal code to describe an image is extremely computationally intensive. This processs can be simplified by splitting the image into small blocks and finding the optimum code to represent each block. Decoding is relatively straightforward, but encoding using this technique is much more complex and takes typically several hundred times longer than decoding.

3.4 Wavelet Transforms for Image Compression

Having discussed various image compression techniques, including the DCT, it is appropriate to describe the wavelet transform, its meaning and how it is applied to image compression for the project. The evolution of a filter architecture from the chosen wavelet transform function shall be discussed in the next chapter.

3.4.1 Wavelets

Wavelets are a relatively recent development in applied mathematics. Their name itself was coined approximately a decade ago; in the last ten years interest in them has grown at an explosive rate. There are several reasons for their present success.

On the one hand, the concept of wavelets can be viewed as a synthesis of ideas which originated during the last twenty or thirty years in engineering(subband coding), physics (coherent states , renormalization group), and pure mathematics (study of Calderon-Zygmund operators). As a consequence of these interdisciplinary origins, wavelets are a fairly simple mathematical tool with a great variety of possible applications. (University of California, 1998)

Within recent years, increasingly sophisticated mathematical tools have been used to analyze image compression schemes. Many of these tools were originally developed for the scientific and engineering applications and have been widely used for years. One example is the ubiquitous Fourier Transform (FT), which has become a cornerstone of modern data analysis. A function in the time domain is translated by the FT into a function in the complex (real and imaginary) frequency domain, where it can be analyzed for its frequency content. The FT describes the original function in terms of orthogonal basis functions of sines and cosines of infinite duration. The Fourier coefficients of the transformed function represent the contribution of the sines and cosines at each frequency. The FT is most commonly used

in the form of the Discrete Fourier Transform (DFT), which analyzes discretely sampled time series.

The DFT works under the assumption that the original time-domain function is periodic in nature. As a result, the DFT has difficulty with functions that have transient components, that is, components which are localized in time. This is especially apparent when a signal has sharp transitions. Another problem is that the DFT of a signal does not convey any information pertaining to translation of the signal in time other than the phase of each Fourier coefficient. The phase values are averaged for the input function. Applications that use the DFT often work around the first problem by windowing the input data so that the sampled values converge to zero at the endpoints. Attempts to solve the second problem, such as the development of the short-time Fourier transform (STFT), have met with marginal success.

In recent years, new families of orthogonal basis functions have been discovered that lead to transforms which overcome the problems of the DFT. These basis functions are called wavelets, which, unlike the sine and cosine wave of the FT, do not have to have infinite duration. They can be non-zero for only a small range of the wavelet function. This "compact support" allows the wavelet transform (WT) to translate a time-domain function into a representation that is not only localized in frequency (like the FT) but in time as well. This ability has brought forth new developments in the fields of signal analysis, image processing, and data compression. (Cody, 1994)

3.4.2 Mathematical Representation of a Wavelet

An in-depth mathematical understanding of a wavelet is beyond the scope of the project. However, numerous sources on wavelet theory provide an indepth mathematical analysis in the subject. Some of the basic equations describing wavelets obtained from Cody are described here.

There are two fundamental equations upon which wavelet calculations are based. These are the scaling function (also called the dilation equation or fundamental recursion) and the primary wavelet function:

$$\Phi(t) = \sum_{k \in Z} a_k \Phi(2t - k) \qquad \Psi(t) = \sum_{k \in Z} (-1^k) a_{k+1} \Phi(2t + k),$$

where Z is the set of integers and the a_k are the wavelet coefficients. Both of these functions are two-scale difference equations and are the prototypes of a class of orthonormal basis functions of the form:

$$\Phi_{j,k}(t) = 2^{j/2}\Phi(2^j t - k);\ j,k \in Z \qquad \Psi_{j,k}(t) = 2^{j/2}\Psi(2^j t - k);\ j,k \in Z$$

The parameter j controls the dilation or compression of the function in time scale as well as in amplitude. The parameter k controls the translation of the function in time. The set of basis functions formed by $\Phi(t)$ and $\Psi(t)$ is a system of scaled and translated wavelets.

Wavelet systems can be either real or complex-valued, though most research has used real-valued wavelet systems. Wavelet systems may or may not have compact support. As mentioned earlier, wavelets have compact support if and only if they have a finite number of non-zero coefficients. Since compact support is what gives wavelets the ability to localize in both time and frequency, this paper will be dealing only with wavelets of that type.

Several techniques have been used to create wavelet systems. These include cubic splines, complex exponentials, and parameter space constructions. Parameter space constructions have been used to construct wavelets which are orthonormal bases. The Haar wavelet (the first wavelet basis discovered) and the Daubechies wavelets belong to this class; and are the type used in this paper. Orthonormal wavelets allow perfect reconstruction of a function from its wavelet transform coefficients by the inverse WT. A set of conditions must be satisfied before a set of coefficients can represent a orthonormal wavelet. These conditions are as follows:

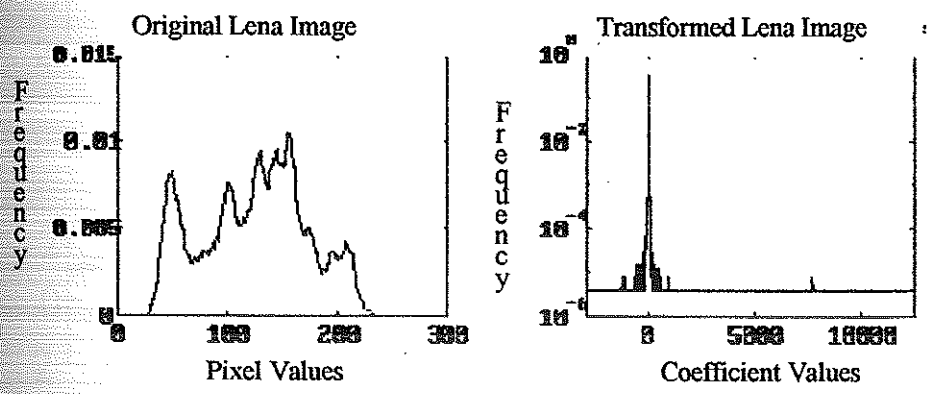
$$\sum_{k\in Z} a_{2k} = 1/\sqrt{2} \qquad \sum_{k\in Z} a_{2k+1} = 1/\sqrt{2} \qquad \sum_{k\in Z} a_k a_{k+2l} = 0 \ \text{for}\ l \neq 0 \qquad \sum_{k\in Z} a_k a_k = 1,$$

Where a_k' is the complex conjugate of a_k.

3.4.3 Wavelet Transformation and Image Compression

Having covered the DCT, the concept of a wavelet transform now becomes relatively easy to digest. A wavelet transform concentrates the original values of a 2-D data set or image into a relatively small number of coefficients. Figure 4 shows a histogram of the pixel values of a 512 X 512 greyscale image, and a histogram of the Daubechies wavelet coefficients for the same image. The information is packed into a relatively small number of large magnitude coefficients. The majority of the coefficients are relatively small in value and do not contribute much information for reconstruction of the original data.

Figure 3.4 Concentration of Wavelet Transform Coefficients in lower sub-bands



The wavelet transform has a property called energy invariance. This property says that the total amount of energy in the image does not change when the wavelet transform is applied. This implies that any change in the wavelet coefficients will result in proportional changes in the reconstructed image. In other words, the low magnitude coefficients can be set to zero without significantly distorting the reconstructed image. In practice, all but a few percent of the wavelet coefficients can be set to zero. (Relue, 1994)

Selection of the coefficients to zero can be done two ways:

- An arbitrary threshold can be established as the cutoff point
- The coefficients can be ranked to allow selecting of an arbitrary percentage of the highest values for retention.

Typically, 5 percent of the values are retained, but good results can be obtained with smaller percentages. It is important to note that the zeroed coefficients cannot be thrown

away. The position of the zeroed coefficients must still be known for reconstruction. This information, however, can be implied by noting the location of the non-zero coefficients.

The compression process has an interesting side effect. Since most of the noise in an image has a low energy value, it will be suppressed when reconstructing the compressed data. Figure 5 shows a sine wave with 50 percent noise added, and the reconstructed sine wave from 3 percent of the original data using Daubechies 2-D transform. The original sine wave is very easy to distinguish in the reconstruction. Relue (1994)

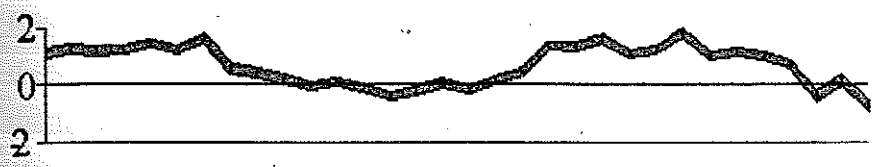
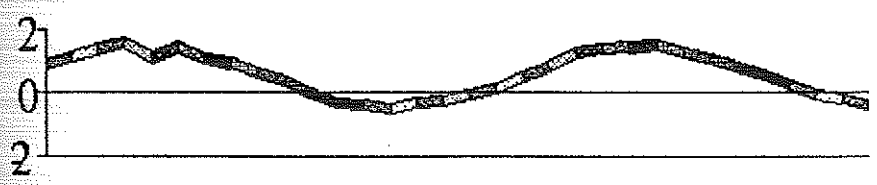


Figure 3.5 The Original (noisy) and Transformed Sine Curve



The reconstruction of low noise signals is generally very good. Simple waveforms such as a sine wave can be done with 3 percent of the data as illustrated in Figure 6.

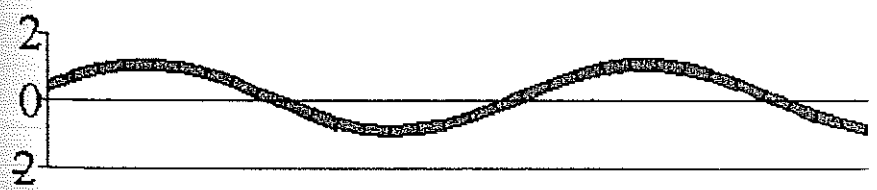


Figure 3.6 The reconstructed low-noise signal.

Once the data has been compressed by the removal of low value coefficients, more compression can be obtained by quantitizing the non-zero wavelet coefficients. The values can be mapped by use of a staircase function to a common value. This will reduce the number of different discrete values in the resulting data, but does result in some loss of

rely on traditional software interfaces, but is entirely contained in the VLSI optics architecture. A critical design constraint for the pixel design is efficient utilisation of chip area. (Lachowicz et al, Paper [5])

The intelligent pixel array emulates a mesh network of processing elements to implement a wavelet transform such that the inherent parallelism and interconnectivity of the architecture is fully exploited. In particular, the use of *triangular (symmetric) filter* demonstrates a marked simplicity for Opto-VLSI without a significant compromise in performance.

4.1.1 The Intelligent Pixel Architecture

Each processing element in the IP array consists of a receiver for image capture and image display (from remote site) as well as local circuitry to perform the necessary operations on the received image. A schematic of he pixel's functionality is shown in figure 4.2. A thin film of ferro-electric liquid crystal (FLC) is placed between the backplane and a front glass electode coated with ITO. A metal pad is then built on the backplance to act as a mirror. Voltages can be determined by a digital driver.

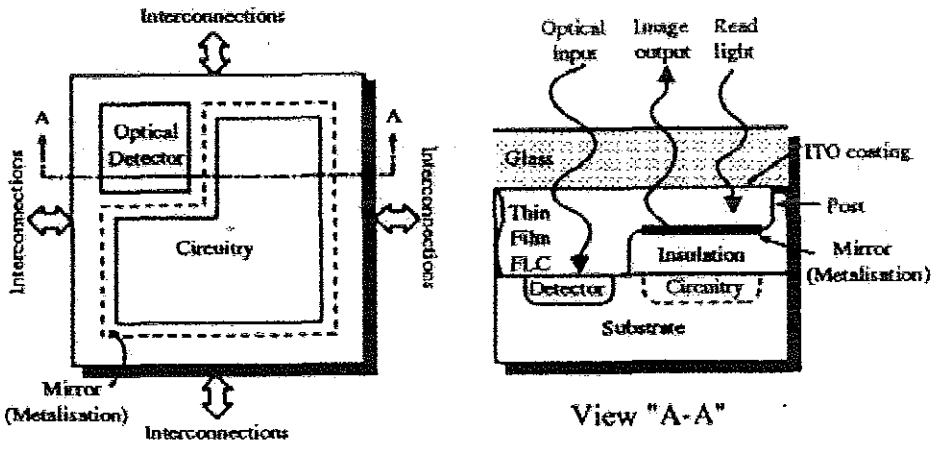


Figure 4.2: Smart Pixel Structure. Adopted from Lachowicz, Alagoda and Ang

Visible to the chip surface is an optical detector for image capture, adjacent to a liquid crystal mirror for image display. It is expected that up to 70% mirror (metal level) fill factor can be assigned for display.

The compression algorithm is implemented within the physical confines of one pixel. The implementation should take advantage of the locality of the operations required. The decomposition and display of images should also take advantage of the relative proximity of the pixel that is used for display. The inverse transform of a data stream to an image, required for receiving images will be obtained via a similar circuit within the pixel. A driver circuit converts the bits into grey-level by space-division multiplexing. The reconstructed image as a whole can then be displayed by the FLC pixel overlay. The response time of the liquid crystal material determines the number of grey levels. As for a mobile communicator display unit, 64 x 64, 6-bit grey level, 10 frames/second appear satisfactory for mobile communication applications.

4.1.2 The Intelligent Pixel Wavelet Transform

Intelligent Pixels perform a sub-band decomposition on an image that segments it into a number of frequency domains at different scales. A complete two-dimensional decomposition of the image can be calculated by sequential convolution of the rows and columns of an image with a pair of suitable chosen quadrature mirror filters (QMFs) followed by half-rate sub-sampling at each scale, as its shown in figure 4.3.

An example of an 8 x 8 image can be observed in figure 4.4, where three scales are needed to get one and only one coefficient of low-low frequency. When any scale is finished, the sample algorithm is applied to the low-low results. The first scale of the algorithm has been illustrated in figure 4.3, where the results are obtained using indirect methods. In the first step, the calculation is applied to the rows, and in the second step elements of the columns are used.

The coefficient, in one dimensional transform, Y_i, can thus be obtained by convoluting the pixel input values, X_i, with the coefficients, W_s, of a FIR filter of length L.

$$Y_i = \sum_{s=1}^{L} X_{i-s} W_s$$

The complete convolution is obtianed through scaling the pixel input by each of the FIR filter coefficients in turn.

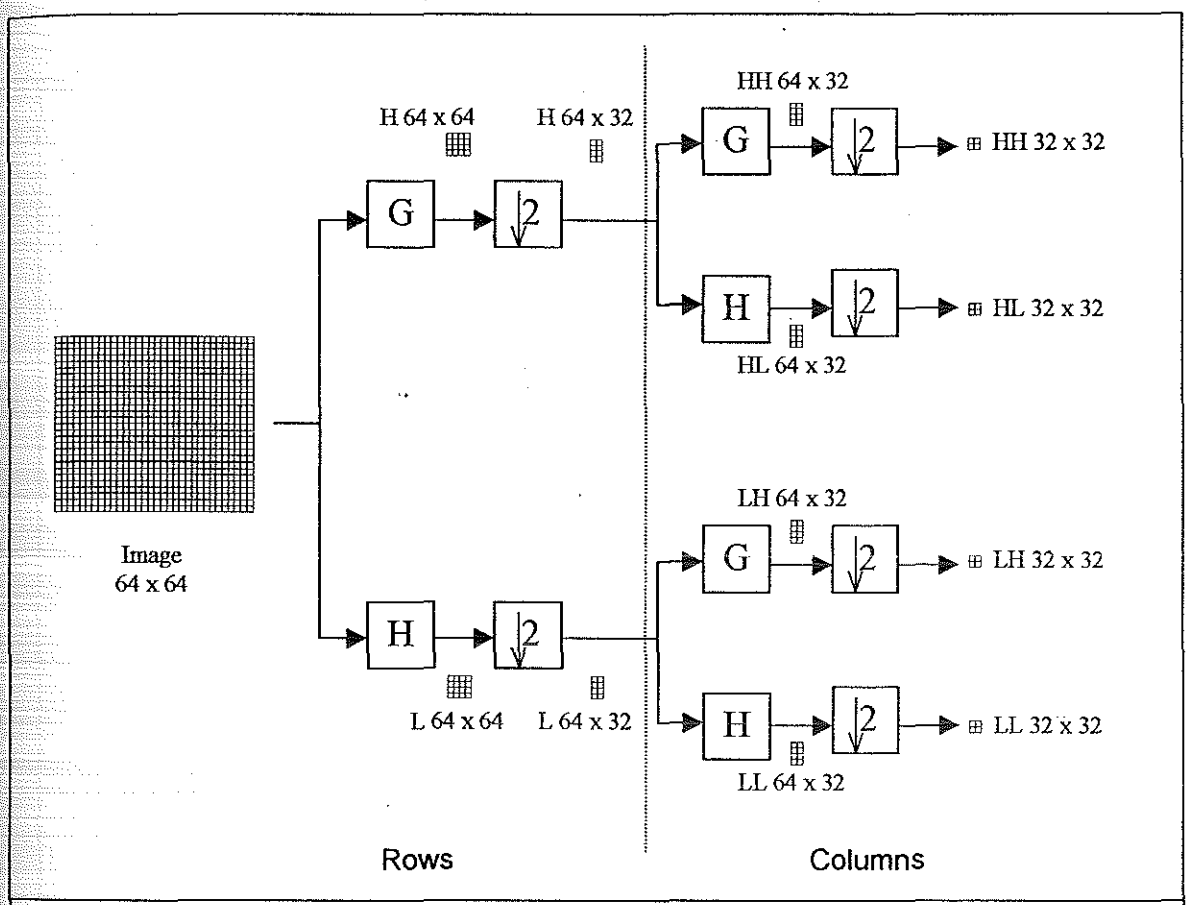


Figure 4.3. Dataflow of the first scales of the 2-D DWT Algorithm. Adopted from Lopez-Sosa (1998).

Due to the space limitations with the IP, simple symmetric QMFs, derived from a triangular wavelet basis, are used, and their coefficients are the following:

QMF = 0, 1, 0 to the low pass filter
DQMF = -0.5, 1, -0.5 to the high pass filter

In the high pass filter, a summation of three numbers is required and the multiplication can be carried out through scaling by a factor of 2. The low pass filter can be reduced to one register without any element of calculation.

The subsample shown in figure 4.4, can be used as an equivalent task, eliminating the calculation of the even coefficients. Figure 4.7 illustrates how the filter is applied to a row of pixels.

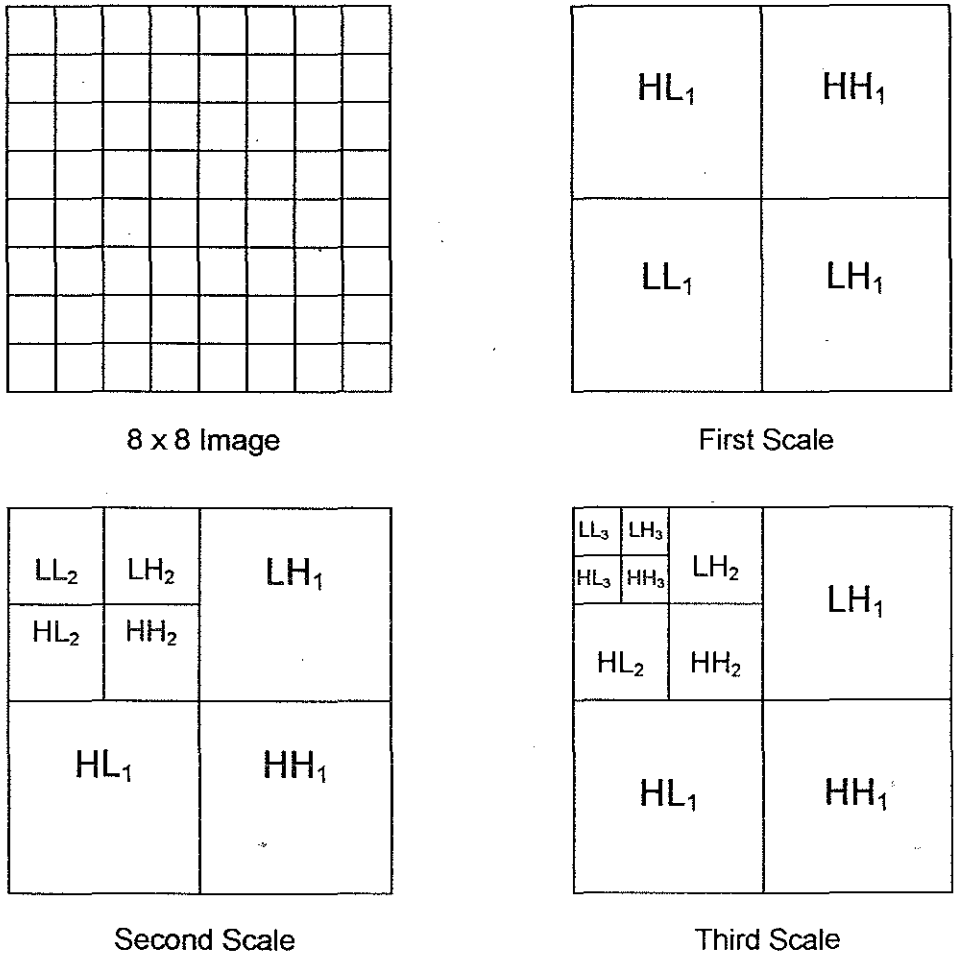


Figure 4.4. Subband decomposition on 8 x 8 image. Adopted from Lopez-Sosa

The values of the coefficients L and H of the first step are;

$L_{11} = X_{11}$, $H_{11} = -(X_{11}/2) + X_{12} - (X_{11}/2)$, $L_{12} = X_{13}$

$L_{21} = X_{21}$, $H_{21} = -(X_{21}/2) + X_{22} - (X_{21}/2)$, $L_{22} = X_{23}$

The inverse transform function can be calculated adding the next values

$X_{11} = L_{11}$, $X_{12} = (L_{11}/2) + H_{11} + (L_{12}/2)$, $X_{13} = X_{12}$

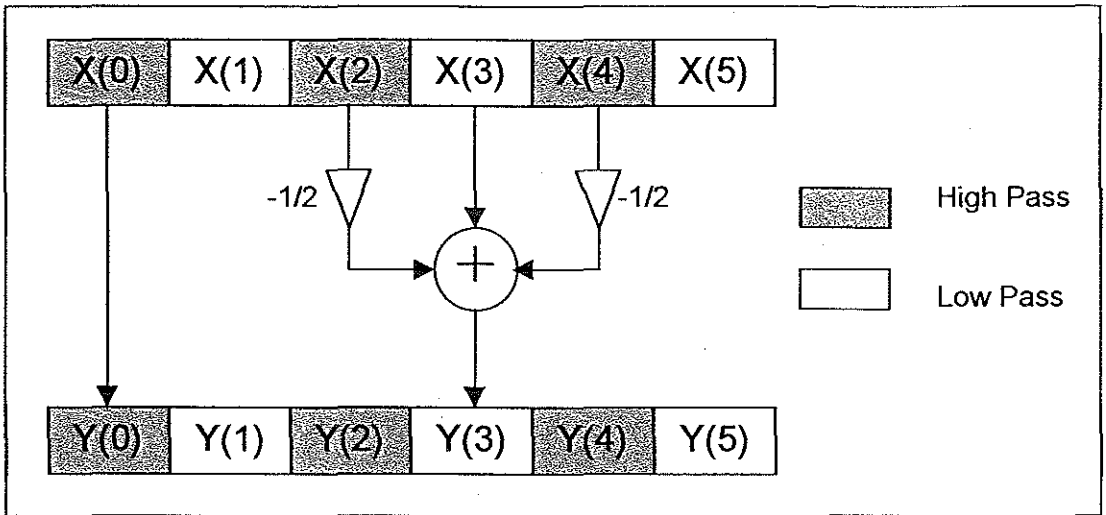


Figure 4.5 Applications of FIR filters to one row array. Adopted from Lopez-Sosa.

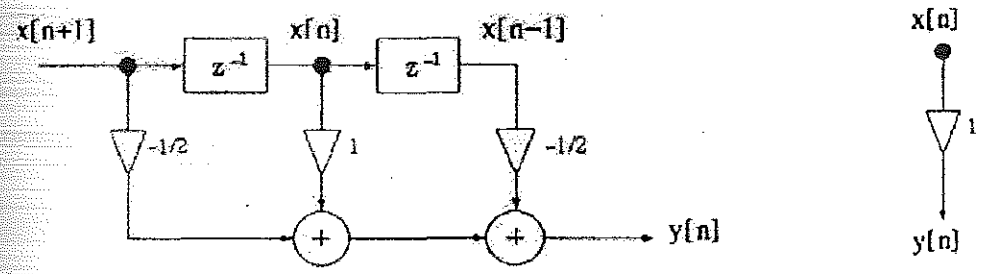


Figure 4.6: High Pass (Left) and Low Pass (Right) FIR filter. Adopted from Lopez-Sosa (1998)

To eliminate the inherent border problem, the coefficient of the ending column is calculated as:

$L_{14} = X_{17}\ H_{14} = -X_{17} + X_{18}$

The figure 4.5 shows the way to implement this part of the algorithms.

Once we have the coefficient L and H, the HH, HL, LH and LL can be obtained using the same method, adding the next values.

$LL_{11} = LL_{11}$, $LH_{11} = H_{11}$

$HL_{11} = -(L_{11}/2) + L_{21} - (L_{31}/2)$, $HH_{11} = -(H_{11}/2) + H_{21} - (H_{31}/2)$

$LL_{21} = L_{31}$, $LH_{21} = H_{31}$

And the calculations for the last rows are

$LL_{41} = L_{71}$, $LH_{41} = H_{71}$

$HL_{41} = -L_{71} + L_{81}$, $HH_{41} = -H_{71} + H_{81}$

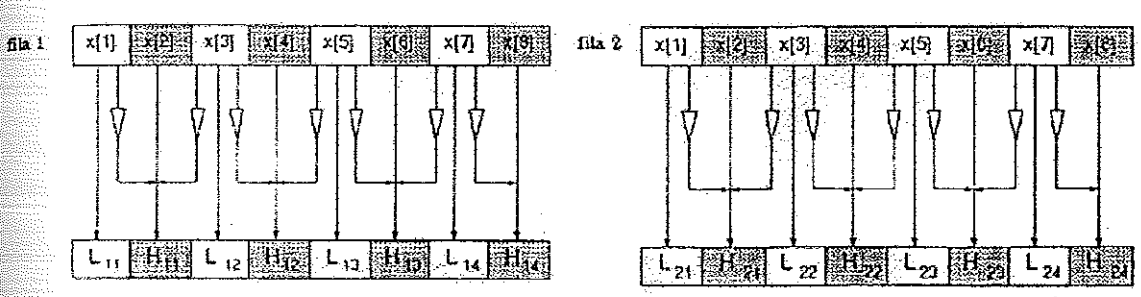


Figure 4.7: Calculation of the first scale row-coefficients. Adopted from Lopez-Sosa (1998).

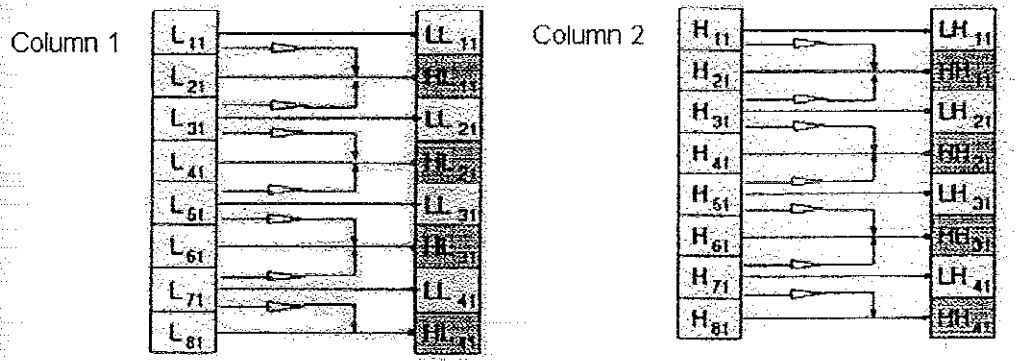


Figure 4.8. Calculation of the first scale column coefficients. Adopted from Lopez-Sosa (1998)

4.1.3 Nucleic Scheme for Data Organisation in IP Arrays

After all the convolutions for a particular scale of the decomposition have been carried out, four frequency subbands are obtained as was shown above. At each subsequent scale it is only the low-low filtered transform coefficients, which are further decomposed. Thus at each scale coefficients from four new subbands must be stored on the array along with all of the coefficients from the three high frequency subbands obtained in the previous scales.

An efficient method to store these subband coefficients with the SP array is to use a nucleic scheme where, as the decomposition progresses to the next scale, only certain nuclei, containing the data required at that scales i.e. the low-low subband coefficients, remain active. The rest of the pixels retain the data from the previous scales but are transparent to

decomposition at the current scale. The resulting arrangement is illustrated in figure 4.9. LL_s^{xy} represents a low-low filtered coefficient at scale s of the decomposition from cell x, y., and the grey cells represent the transparent pixels in that scale.

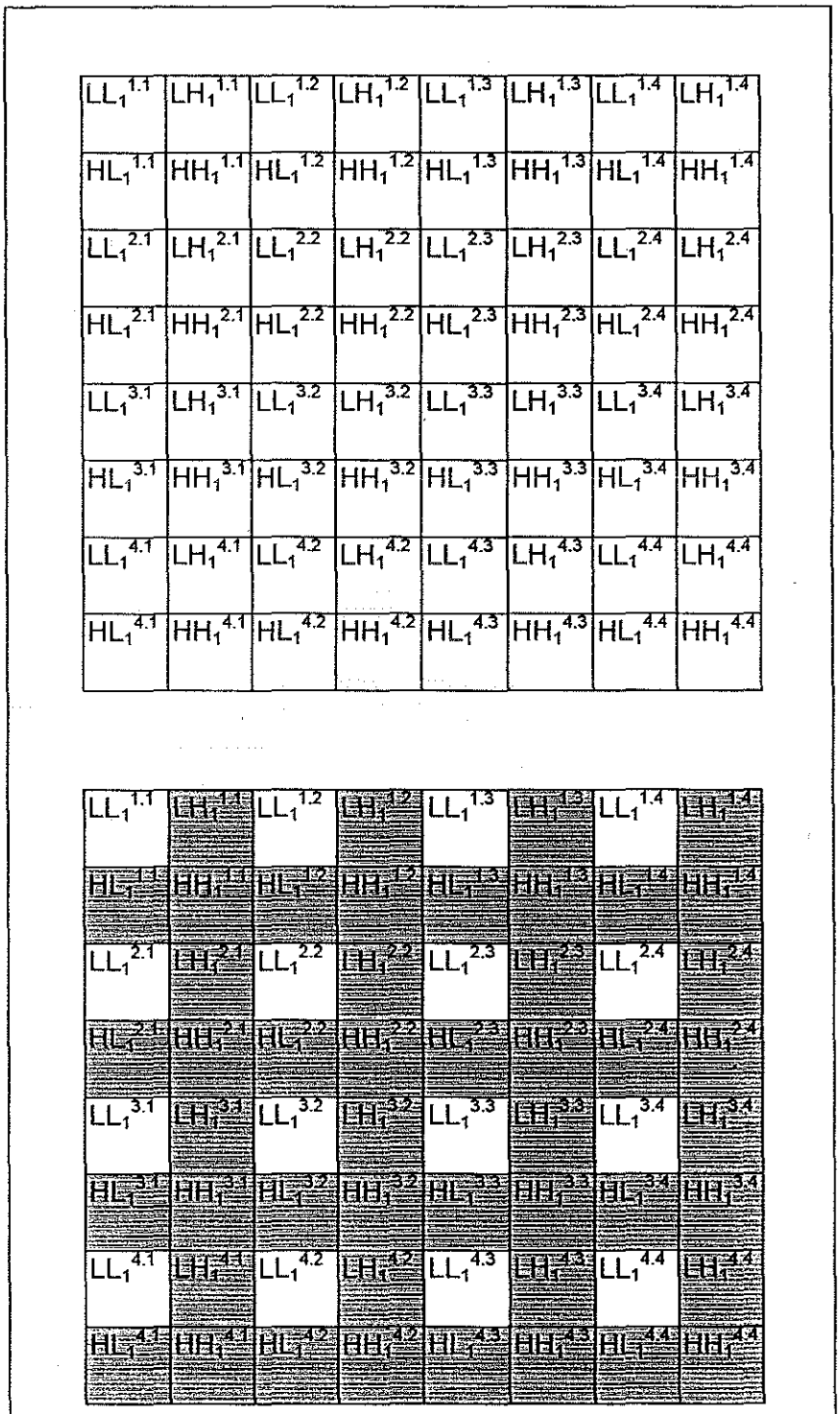


Figure 4.9: Results of first and second scale of algorithm. Adopted from Lopez-Sosa (1998).

Such a nucleic scheme requires no rearrangement of data prior to decomposition at subsequent scales and, assuming that data can pass efficiently through the transparent pixels, allows the convolution algorithm to work with equal effectiveness at any scale. (Lopez-Sosa, 1998)

4.1.4 Algorithmic Analysis of the Intelligent Pixel Array

The IP has to be simple, compact and capable of operation as part of a massively parallel array of processing elements. Figure 4.10 illustrates the tasks that the IP has to perform:

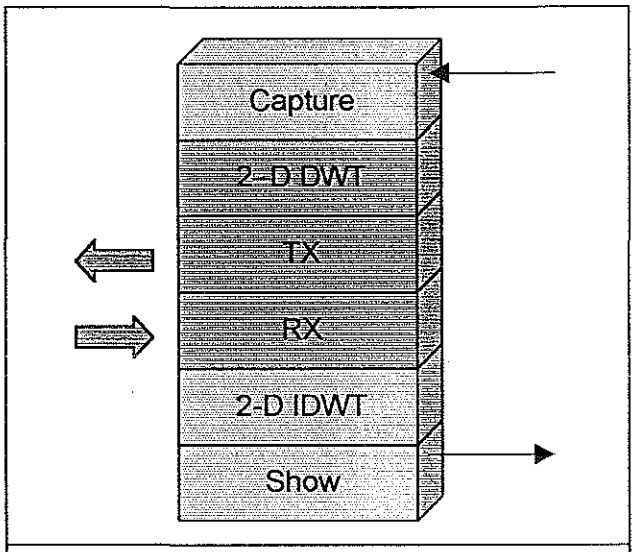


Figure 4.10: Intelligent Pixel Tasks

- To capture the image,
- To compute the 2-D DWT,
- To transmit the coefficients calculated,
- To receive the remote coefficients,
- To compute the inverse 2-D DWT, and
- To show the remote image.

The principal objective is to reduce the DWT algorithm until the minimal implementation is obtained in minimal area. We have to design a pixel architecture that is able to execute the algorithm when it is active in the scale to be transparent in the following scales.

This algorithm and the related pseudocode have been shown in the figures 4.11 and 4.12.

```
To read the pixel value
If it's even in the row (even column)
        To ask for the left pixel value and to invert it
        If it isn't the last column (there is a right pixel)
                To shift it to the right (division by 2)
        To add it to the actual pixel value

If it isn't the last column
        To ask for the left pixel value and to invert it
        To shift it to the right (division by 2)
        To add it to the previous calculated value

If it's even in the column (even row)
        To ask for the left pixel value and to invert it
        If it isn't the last row (there is a down pixel)
                To shift it to the right (division by 2)
        To add it to the actual pixel value

If it isn't the last row
        To ask for the left pixel value and to invert it
        To shift it to the right (division by 2)
        To add it to the previous calculated value

Continue to the next Scale
```

Row Steps

Column Steps

Figure 4.11: Intelligent Pixel Operation Algorithm. Adopted from Lopez-Sosa (1998).

```
Begin
    R1 = Read(Image)
    Scale = 0
    Repeat
        If column is even
          Copy (R2, left)
          R2 = -R2
          If isn't the last column
                Shift (R2, right)
          R1 = R1 + R2
          If isn't the last column
                Copy (R2, right)
                Shift (R2, right)
                R1 = R1 + R2
          End if
        End if
        If row is even
          Copy (R2, up)
          R2 = -R2
          If isn't the last row
                Shift (R2, right)
          R1 = R1 + R2
          If isn't the last row
                Copy (R2, down)
                R2 = -R2
                Shift (R2, right)
                R1 = R1 + R2
          End if
        End if
        To be transparent if it isn't (odd, odd)
        Inc(scale)
    Until scale = 6
    End
```

Figure 4.12: Pseudocode based on IP Algorithm. Adopted from Lopez-Sosa (1998)

4.2 Design of the Filter Bank

Having discussed the algorithmic functionality and the requirements for the IP, the next consideration is the design of the filter bank circuitry that can perform the desired wavelet transforms to scale the input images. As seen in the above section, the architecture of the filter to perform the transform would be something like the one shown in figure 4.13 below.

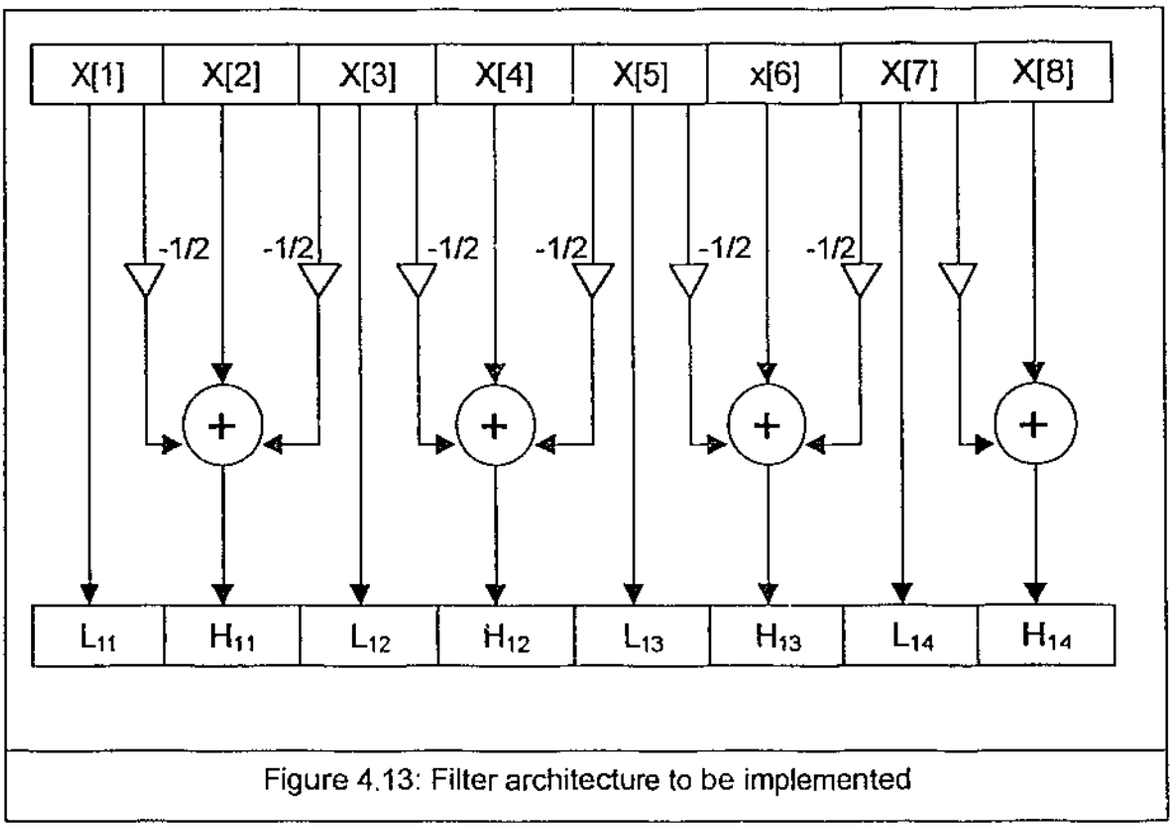


Figure 4.13: Filter architecture to be implemented

This design was derived from the high pass and the low pass functions for the triangular filter. Looking at the schematic presented above, the requirement of various functional blocks such as registers, multiplexers, adders etc was deduced. The description below contains an explanation of the system blocks from a functional point of view.

4.2.1 The Bit Serial Architecture

Before proceeding on to the implementation of the required blocks, it is essential to note that the filter must be designed using a bit-serial architecture. A bit serial architecture is where the transfer and manipulation of data takes place by serial transmission one bit at a

time. For example, the register implemented in a bit serial architecture would be a shift register, so that it can be loaded by shifting the data bits into it from the right or left side of the register. This is in contrast to the parallel architecture, where all the register cells can be loaded in one operation.

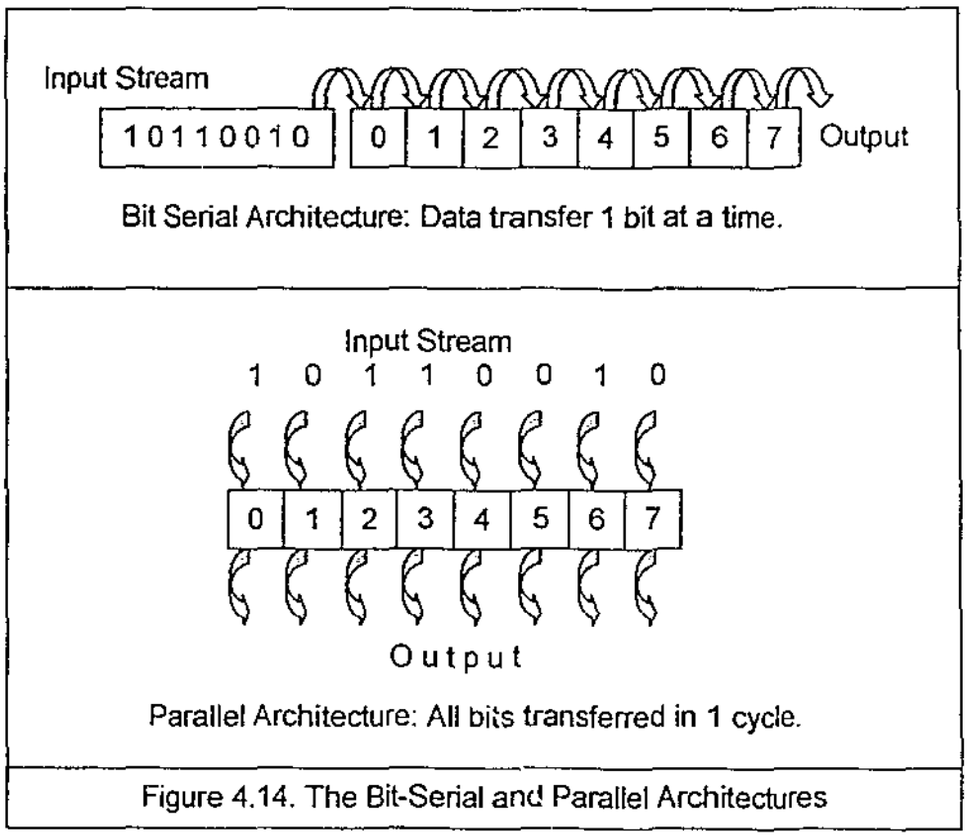


Figure 4.14. The Bit-Serial and Parallel Architectures

- The intelligent pixel construction shows that information about the value of each pixel is obtained from the ADC. These values undergo the wavelet transform as depicted by the above schematic. Thus, to implement the wavelet transform function, we need to examine the value *bit by bit* and perform the appropriate additions/subtractions and store the result in the output.
- We need to implement the filter architecture in a self-timed fashion. This means each operation should be able to produce a 'Done' signal, which can trigger the next operation. The operation can be better co-ordinated if the system were to perform on one bit at a time.
- For a bit-serial system, addition/subtraction etc can be performed by a 1-bit adder and carry circuit. However, to perform similar functions for a parallel system, we need a multiple-bit adder capable of adding all the bits in parallel and producing a cumulative carry and sum output. Such architecture would be more complex and considerably larger than the one-bit adder. Since the intelligent pixel architecture must be very conservative in chip area, the bit serial architecture is a natural choice.

- All the above factors make it necessary for us to implement the filter bank in a bit-serial architecture.

4.2.2 The Register Block

To be able to perform the wavelet transform, we need to store the image value of each pixel and read it one bit at a time and store the outputs. Thus, the system requires a shift register.

The filter bank can receive data from different sources

- The analog to digital convertor, which converts the image value of the pixel into data
- Adjoining pixels; while performing transforms (forward or reverse) values need to be input from adjoining pixels and then multiplied by either ½ or –½. Thus the input is received from adjoining pixels.
- The register output itself. When no operation needs to be performed on a bit (low pass), the data can be looped back to the register.

Thus, to be able to select the source of data being fed to the register (from the left or the right), we need to design a **4 x 1 multiplexer**.

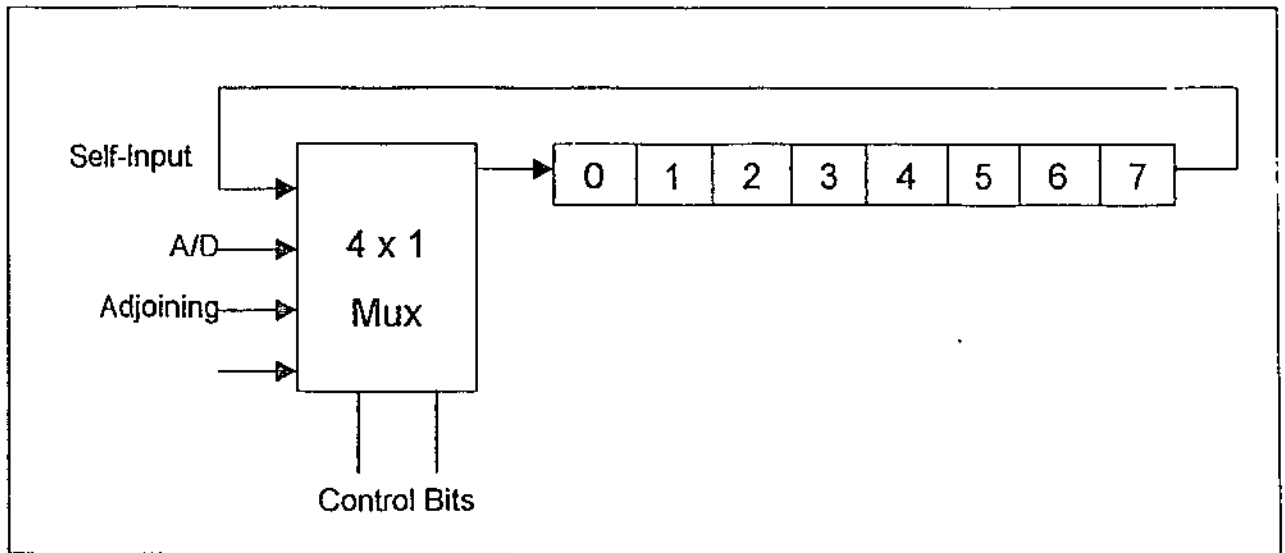
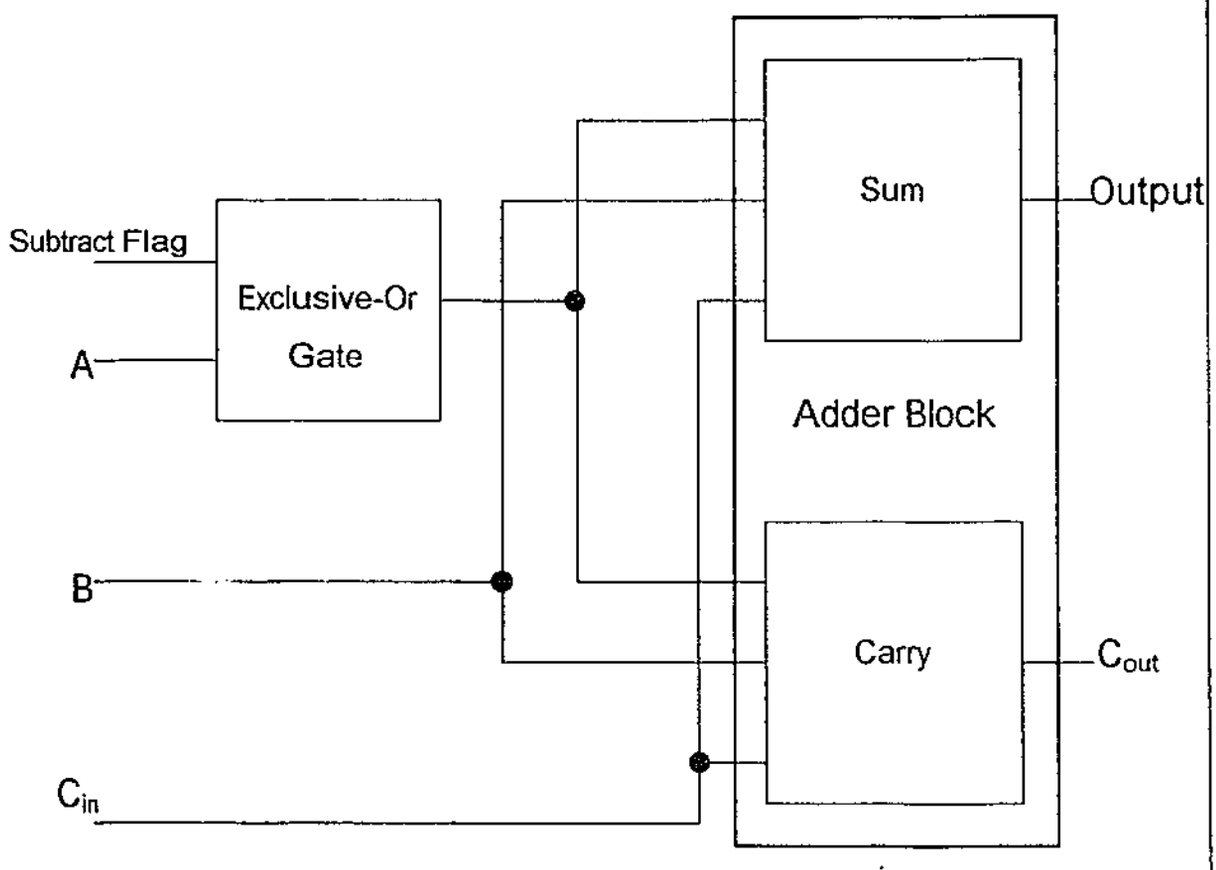


Figure 4.15. Selection of input to the shift register using a 4 x 1 Multiplexer.

The wavelet transforms require multiplication of values by ½ and –½. However these operations do not require a multiplier; an easier way to implement them is to take in 1-bit left shifted output thus effectively halving the value received. The positive or negative sign can be decided later before the value is input to an adder block.

4.2.3 The Adder Block

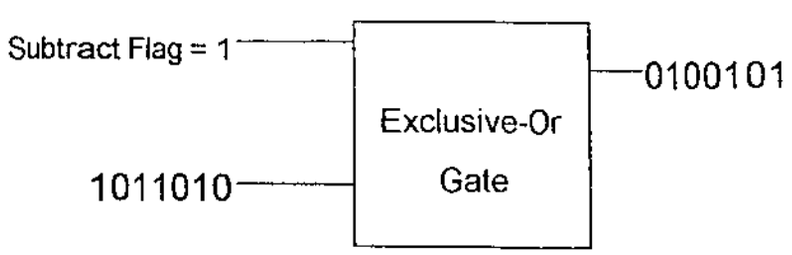
To perform the required wavelet transforms, an adder block required. This constitutes two different operations: adder and carry. To enable selection between addition/subtraction one of the inputs is preceded by an exclusive-or gate. Thus, one of the inputs for this gate is the input stream and the other is a flag signal which is switched to logic '1' when the operation is a subtract. This effectively inverts the input stream being fed into the adder, and thus implements a subtractor.



The Adder Block Implemented with the Subtract Flag

Input	Subtract Flag	Output
0	0	0
1	0	1
0	1	1
1	1	0

Truth Table: Subtract Flag



Block Diagram: Inversion of Output when Subtract = 1

Figure 4.16. Block Diagram of Adder Block: Using the Subtractor Flag

4.2.3.1 Generation of Carry Bit

The carry bit being input into the adder C_{in} is intuitively taken as the C_{out} from the previous bit addition. However, this does not define the value of the first C_{in}. In the addition operation, the carry in for the first bit of the value should always be '1' and then it should be dependent on the value obtained from the previous carry operation. This can be implemented by an OR gate, such that one of its inputs is always '1' for the first bit, thus setting the output to '1'. For the rest of the bits, this input becomes '0', making the output dependent on the second input. The additional control required for determining the first bit of the values can be designed with Finite State Machines.

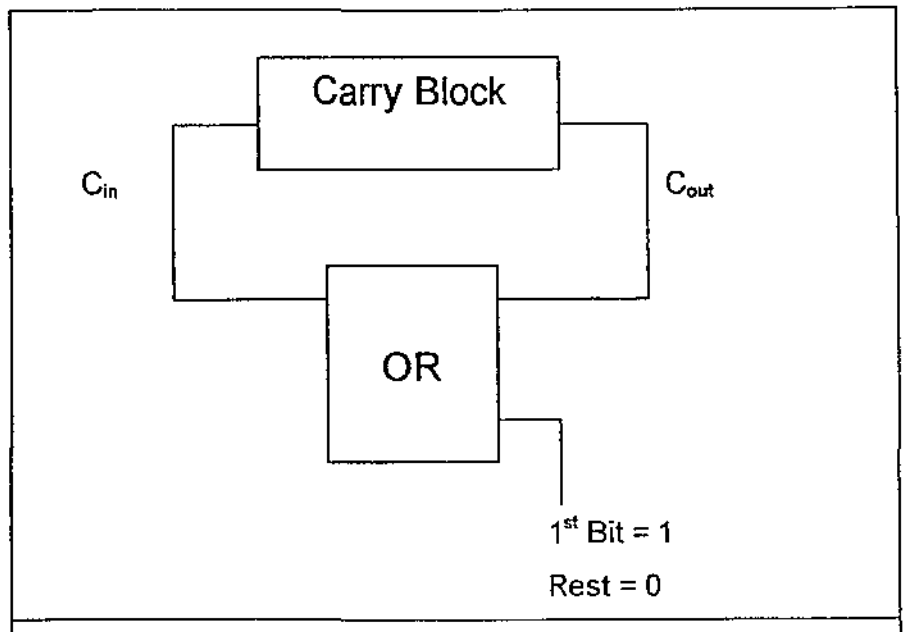


Figure 4.17. Carry Generation

4.2.4 Overall Filter Architecture

The diagram depicted below summarises the above discussed requirements for the system and shows how the data should propagate through the system. This diagram does not include details about the handshake signals required for the timing co-ordination. Details about incorporating the handshake signals and control into the system are discussed in the next chapter.

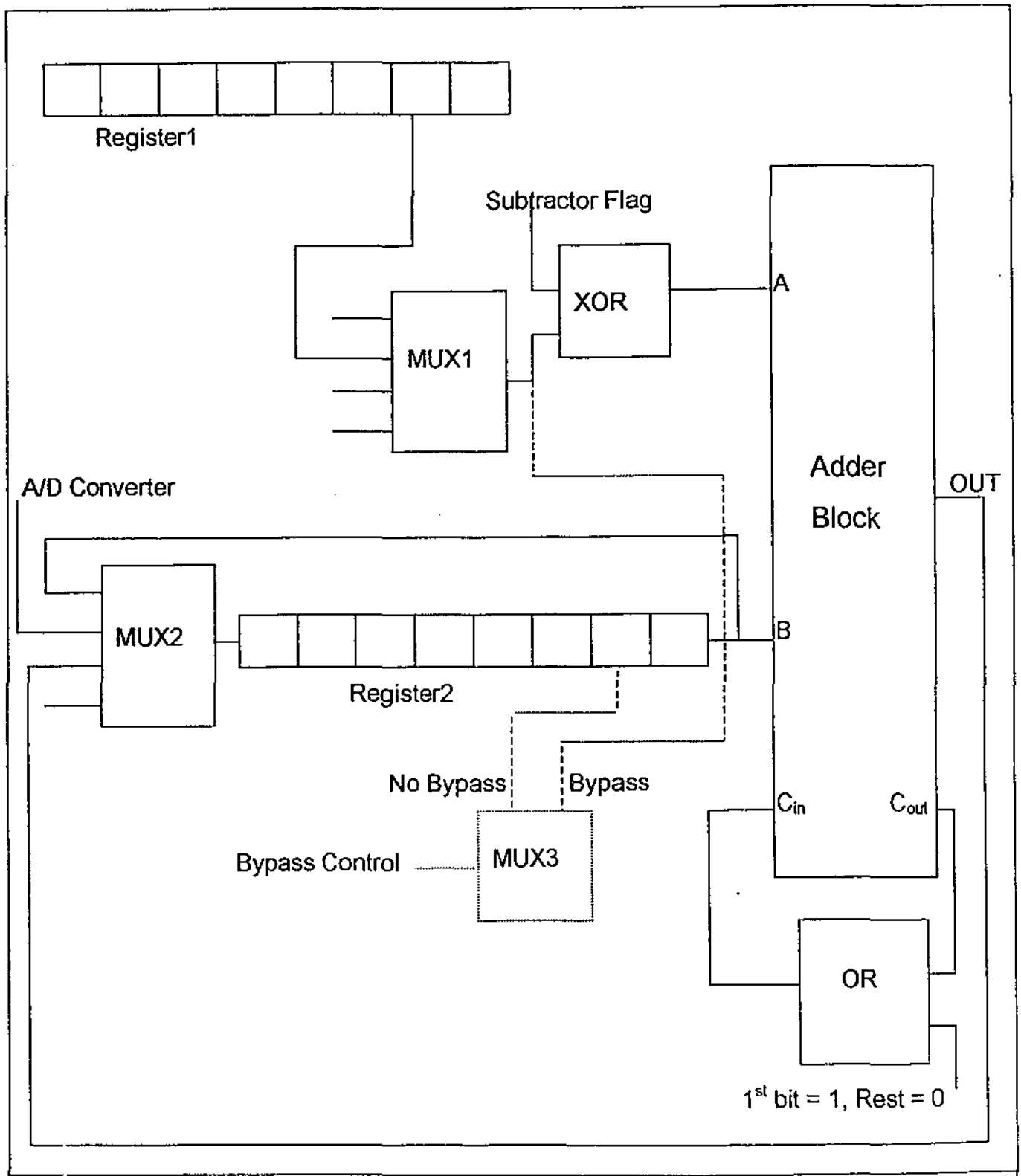


Figure 4.18. Schematic System Diagram of Data Flow through the System

The functionality of the design shown above is described in the following steps.

- The one bit shifted output from Register1 passes through the 4x1 multiplexer Mux1. The Mux1 is incorporated in the design so that the incoming input data stream can be from either of the four directions (up, down, left or right) depending on what pixel value is taken for the addition.
- The data stream is then passed through the exclusive-or gate XOR which enables the operation to be switched between subtraction and addition by controlling the value of the subtract flag.
- The output from the XOR block forms one of the input streams for the adder block. The second input is the data stream received from the register Register2.
- The output from the adder block is fed to the Register2.
- To enable selection of the input stream, a 4x1 multiplexer Mux2 is implemented before the register 2. This multiplexer selects the input stream between the adder output, the ADC, and the register output itself.
- While performing the wavelet transform, sometimes it is necessary to skip the immediately next pixel and obtain input from pixels one step further. In that case, we should be able to bypass the neighbouring pixel. This can be implemented by a 2x1 multiplexer Mux3, which can control the input based on the control signal, which is flag indicating "Bypass". This gate, however, shall not be implemented in the filter architecture and has been postponed to the time when filter design is integrated to form a fully functional intelligent pixel array architecture. Thus it is shown in dotted lines.

5 System Integration and implementation

5.1 Introduction

In the previous chapter, we witnessed how IP arrays could be utilised to perform the desired wavelet transforms on an image. We also developed a model based on various building blocks, and depicted the data flow required for the system. The building blocks required for the triangular wavelet transform were designed and interconnections were shown as required for the communication of data (only) through the system.

In this chapter, we further develop the design to incorporate the handshake and control circuitry for the filter. Then, with a comprehensive circuit design available, we can divide it into logical blocks and be able to implement the layout for each block and simulate its behaviour. An overall schematic for the filter, from the point of view of layout, is also shown.

The layout editor MAGIC was used to implement various cells in the design. The basic design rules for H-GaAs III designing are also shown. HSpice software was used to perform circuit analysis on various blocks. When a cell layout in MAGIC is fully functional, its corresponding electrical circuit can be extracted, and HSpice can be used to perform output analysis. Graphing software called AvanWaves was used to perform graphical analysis on HSpice outputs. As mentioned earlier, the height of the Schottky barrier determines the voltage in GaAs circuits, and thus 0.7V indicates a logic '1'.

5.2 System Integration: Handshake Circuitry

Figure 4.18 in the previous chapter provides us with an overall architecture for a filter cell. However, this diagram considers only the data transfer aspects of the system. For a self-timed circuit, it is required that data transfer between various components of the system is also accompanied by transfer and manipulation of appropriate handshaking signals. Furthermore, special logic is required to handle this incoming handshaking signals for a module and to output appropriate signals for the next block. In this section, the handshaking circuitry is added to the overall system diagram in a step-wise manner. Please refer to figure 4.18 for a clearer understanding of functional blocks discussed.

1. A register design shall be required to implement Register1 and Register2. These registers are required to be able to shift data one bit at a time, and take in new input data from the left and output data at the right. Thus, a 'Start' signal is required to signal the registers to shift one bit of data. In this process, the left-most bit shall be read from the data source and the right-most bit shall be output to the Adder. Thus, after input is ready to be fed into the left-most bit, a 'Start' signal should trigger a shift operation. Similarly, after the output is read in from the right-most bit, a 'Done' signal should be output.

   The register is composed of individual datacells. Thus, within the register itself, it is necessary to have handshaking signals and controls to regulate data transfer between each data cell. The details of these controls have been covered later in the chapter. The architectural blocks required for this are a Muller C cell, a Handshake Block and a basic LCFL cell.

2. The multiplexer MUX1 and exclusive-or gate EXOR will not contain any timing controllers. They shall be implemented using simple combinational circuits. It shall be shown later why it is safe to do so, without taking the timing considerations into account.

3. The multiplexer MUX2 is also implemented using combinational circuits only. It should be made sure, however, that all the inputs of the multiplexer are present and the output has been computed before a 'Start' signal is sent to the Register2.

4. For the adder to start computing the sum of the inputs from Register1 and Register2, it requires a 'Start' signal. This 'Start' signal should be generated only when both the registers have finished execution ie. A done signal has been received from both the registers. However, both the registers might receive their respective 'Start' signals at different times, thus finishing at different times too. It can also be seen that Register1 requires one less bit transfer then Register2 since we read a one bit left-shifted output. On the other hand, output from Register1 must be processed through two combinational circuits, a Mux1 and an XOR.

   All the above factors create uncertainity as to which register shall finish operation first and be ready to provide output to the adder. One solution is to be able to wait for both the 'Done' signals from the registers, and produce a 'Start' for the Adder only when both the 'Done' signals have been received. Thus we need to design a functional block which

is capable to receiving two pulse inputs at different times, and output a pulse after the later input is received. This functional block is called Cell1 in our design and is designed primarily using D-latches.

5. The Adder block receives a 'Start' signals and starts computing the Sum and Carry from the two inputs of the registers and the carry input received from previous operation. For this it requires two different functional blocks, Sum and Carry. Each of these blocks is designed along the lines of a basic LCFL cell, with the required logic for performing Sum and Carry being built into them. Each of the two blocks, Sum and Carry can generate a 'Done' signal when their execution is finished. However, it can be shown that the Sum block based on the LCFL configuration is significantly slower than the Carry block. Thus, a 'Done' signal from the Sum block should suffice to indicate that both the cells have completed execution and the logic values present at Output are correct.
6. The output from the Carry cell must act as input for the Carry operation in the next computation. However, the adder block is required to perform subtraction too. To perform subtraction, the first bit of the input is inverted. The configuration shown in figure 4.18, however, cannot be applied to self-timed system design. This is due to the fact that a looped conducting structure with no break is not good design practice. Thus two new cells are introduced which act as a "subtract-selector", and "data buffer" respectively. The details of these cells shall be given later in the design section. Cell2, as shown in overall system diagram, requires the 'Done' signal from the Adder as its 'Start'. Cell 3 in turn derives its 'Start' from the output of Cell1. The 'Start' signal for the Adder is changed from output of Cell1 to the 'Done' signal of Cell 3.

   Cell2 takes in two signals Sub and Request(Sub). Sub is set to '1' for the first bit of the register in case of the operation being a subtraction. Req(Sub) is also set to '1' around that time to ensure that the cell goes through a precharge and evaluate cycle. Cell 3 is just a data buffer, imitating a 1-bit Register. In this cell, the Output follows the Input value upon receiving the 'Start' signal. This 'Start' signal is the output of Cell1, which pulses when both the registers have finished execution. The 'Done' from this cell indicates that data has been shifted from the buffer to Adder input. Thus, this 'Done' signal acts as the 'Start' for the adder block.

7. The control circuitry to perform the 'Bypass' function, as discussed in the system diagram before, is provided as optional circuitry. It can also be implemented as self-timed circuitry; however such implementation has been left to further development of the design beyond the level achieved in this project.

The above discussed functional blocks required to perform to communicate and process data as well as control signals through the system have been described in the System Diagram shown as figure 5.1.

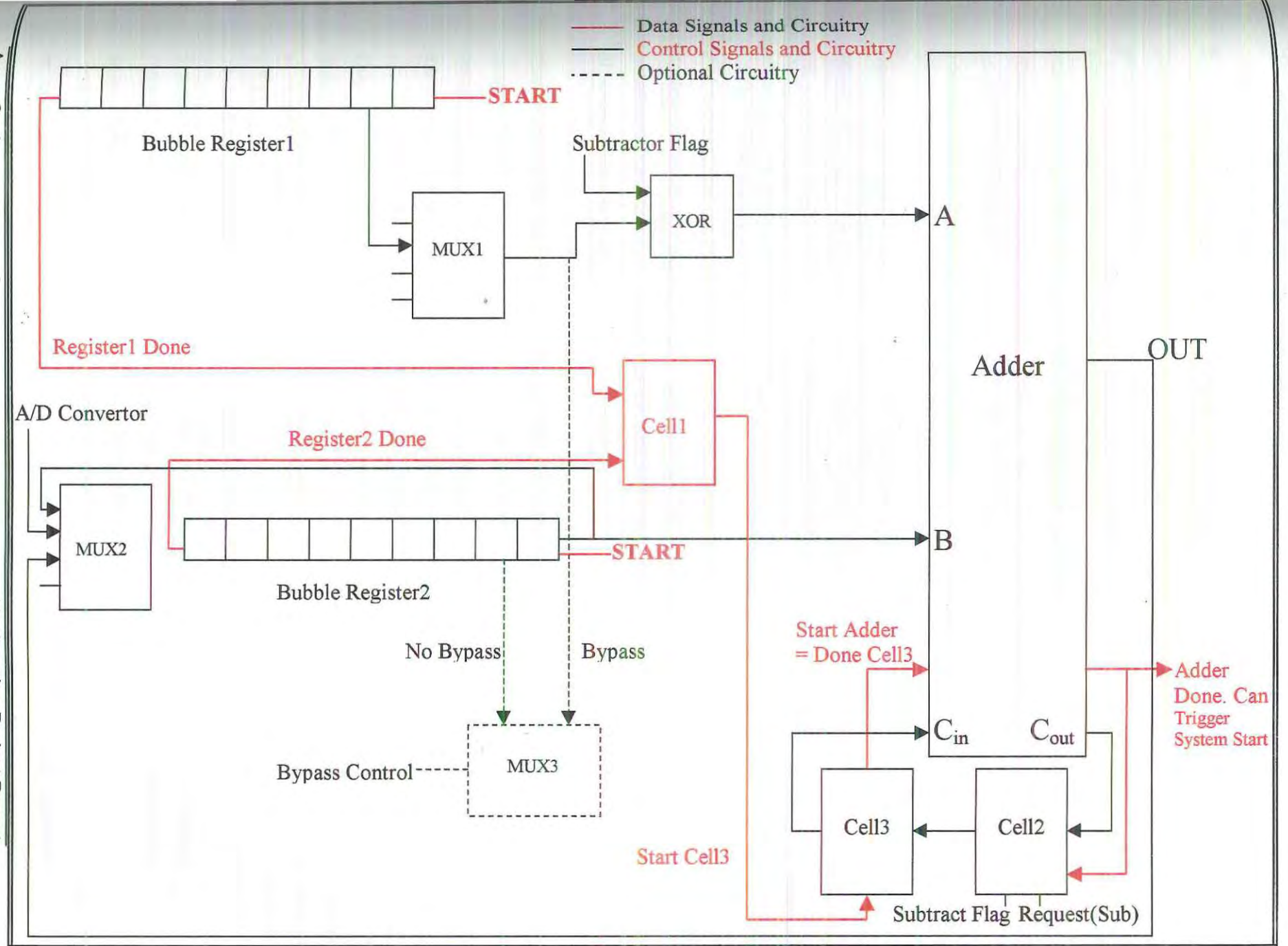


Figure 5.1. Overall System Diagram including Data and Control Signals

5.3 Layout Rules for H-GaAs III Design

Design rules, or Layout rules, can be considered as a prescription for the preparation of the photomasks that are to be used in the fabrication of integrated circuits. The rule set provides the necessary link between the circuit designer and process engineer during the manufacturing phase of an IC. The main objective associated with the design rules is to obtain the circuit with optimum yield in as small a geometry as possible without compromising reliability of the circuit. The design rules for H-GaAs III described here were adopted from Eshraghian (1993).

Usually, the layout rules represent the best possible compromise between yield and performance. In fact, the more conservative the rules are, the more likely it is that the circuit will function. However, the more aggressive the rules are, the greater the probability of improvements in circuit performance. Such an improvement may be at the expense of yield. Design rules specify to the designer certain geometric constraints on the layout artwork so that the patterns on the processed wafer will preserve the topology and geometry of the design. What is significant is that layout rules do not represent some hard boundary between correct and incorrect fabrication, but a tolerance that ensures very high probability of correct fabrication and subsequent operation.

Circuit designers usually want tighter, smaller layouts for improved performance and decreased area. On the other hand, the process engineer calls for rules that result in a controllable and reproducible process. One important factor associated with design rules is the achievable definition of the process line equipment.

Over the years several approaches have been used to describe the design rules. Two major approaches are

- Lamba-based rules.
- Micron-based rules.

The lambda-based rules used earlier in the text were made popular by Mead and Conway (1980). These rules are based on a single parameter, lambda, which characterises the linear features as well as resolution of the complete wafer implementation process.

In contrast, micron based rules provide specification the minimum separation between various layout materials in physical dimensions. Degradation in circuit performance can

make the lambda based design approach unsuitable for GaAs process. However, it makes the layout much simpler to implement and compatible for other technologies as well.

Table 5.1 shows the minimum spacing required between various layers in the layout.

Layer	Rule Feature	Dimension (lambda)
Active (Diffusion)	A1 minimum width	5
	A2 minimum spacing	5
	A3 minimum to n+	5
	A4 minimum E-MESFET width	5
Depletion implant n+	B1 minimum D-MESFET gate overlap	2
	B2 minimum width	7
	B3 minimum spacing	5
	B4 Minimum spacing to E-MESFET	2
Ohmic contact	C1 minimum ohmic contact width	5
	C2 minimum ohmic-metal spacing	5
	C3 minimum cut overlap	2
	C4 minimum ohmic contact size	5 x 5
Gate metal	D1 minimum gate-metal gate extension	2
	D2 minimum gate-metal length	3
	D3 minimum gate-metal width	3
	D4 minimum cut overlap	2
	D5 minimum gate-metal spacing	5
	D6 minimum spacing to ohmic contact	3

Table 5.1. Lambda Based layout rules for GaAs. Adopted from Eshraghian (1993)

Layer	Rule Feature	Dimension (lambda)
Contact	E1 minimum cut size	4 x 4
	E2 minimum cut spacing	4
	E3 minimum spacing to via	4
Metal 1 (Diffusion)	F1 minimum width	4
	F2 minimum spacing	5
	F3 minimum cut overlap	2
	F4 minimum via 1 overlap	2
Via 1	G1 minimum via size	5 x 5
	G2 minimum via spacing	5
Metal 2	H1 minimum width	5
	H2 minimum spacing	5
	H3 minimum overlap of via 1	2

Table 5.2. Lambda Based layout rules for GaAs. Continued from previous page. Adopted from Eshraghian (1993).

5.3.1 Width and Spacing Rules

Although diffusion, metal 1, and metal 2 can cross each other without interaction, in some processes metal 1 is not permitted to cross diffusion.

The width and separation rules given in the table are dependent upon the width of the photoresist. We need to ensure that regions of two unrelated implants do not interact. The separation between implant is determined from

- Width of the depletion region; and
- Width of the photoresist.

Crossing of metal 2 over channel areas of MESFET should be avoided.

5.3.2 Transistor Rules

There are two types of implants used to form the two different MESFETs. A transistor is depletion type if it is inside the n^+ yellow region, otherwise it is enhancement mode.

It is essential for gate-metal (red) to completely cross the implant (green) region, otherwise the transistor that has been created will be shorted by a n^- path between source and drain. To ensure this condition is satisfied, 2λ of gate metal extension is necessary. This is termed the 'Schottky gate extension'.

Orientation is an important consideration during layout. All MESFETs need to be positioned horizontally owing to the anisotropic nature of GaAs, which influences the threshold voltage of the device brought about as a result of variation in both concentration density and channel thickness.

Some processes require isolation between devices to reduce their interaction. This is achieved through lattice damage. The mask is derived from the 'logical' operation of the active layer masks.

5.3.3 Contact Cut and Via Rules

Generally the size of a cut is established from the knowledge of the minimum dimensions necessary to give an acceptable resistance. The ohmic contact has a current capability in the range of 0.5-1.0 mA/μm long. The rules that one may follow are

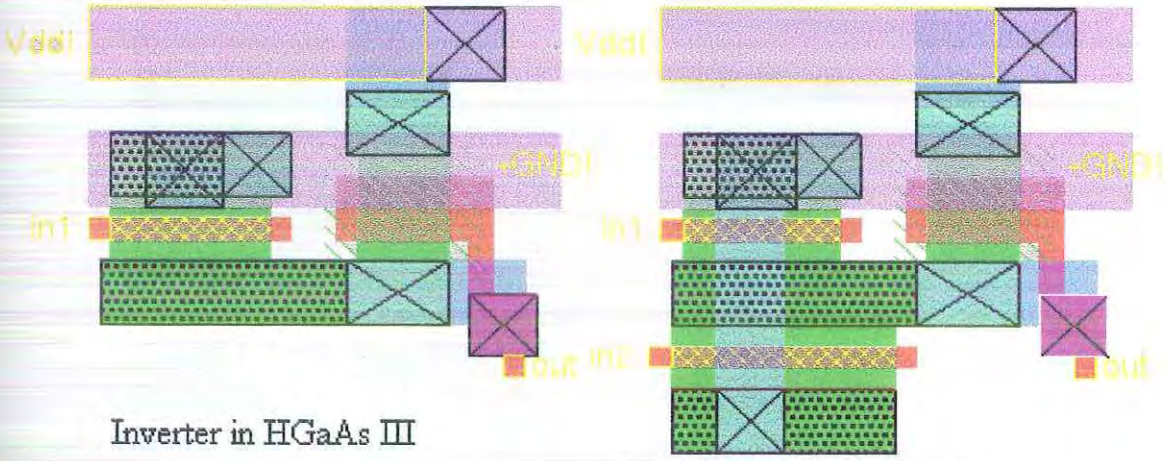
- Minimum dimensions of ohmic cut for source/drain are 5λ x 5λ.
- Minimum dimensions of a cut are 4λ x 4λ.
- Via dimension is 5λ x 5λ.
- Metal 1 overlap of via is 2λ.
- Metal 2 overlap of via is 2λ.

5.4 Basic Gates in HGaAs III

In this section, the two basic gates that can be implemented in GaAs are described. The Modified Ring Notation was described in section 2.3. The NOR gate and the Inverter shown here are implemented in DCFL family using MRN. The Pull Up/Pull Down ratio of 10:1, as derived for the DCFL family, can be observed here.

Inverter		NOR Gate		
Input	Ouput	Input 1	Input 2	Output
0	1	0	0	1
1	0	0	1	0
		1	0	0
		1	1	0

Table 5.3. Truth tables for Inverter and NOR.

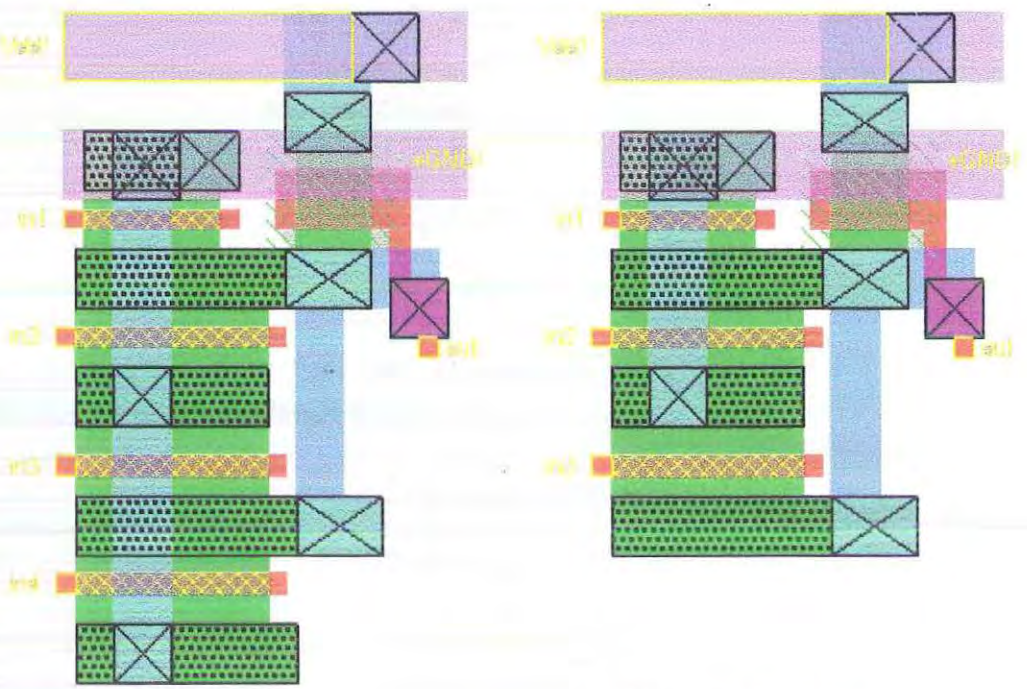


Inverter in HGaAs III

2 Input Nor in HGaAs III

Layout 1: Basic Gates

- As seen above in the layout for Inverter and Nor Gate, the Metal2 (Dark Blue) is used to provide Vdd and Ground for the circuit.
- A Transistor (yellow checkered and green hash) is formed when Gate Metal (Red) crosses oxide layer (Green).
- There are two different types of transistors, Depletion/Pull Up (Yellow Checkered) and Enhancement/Pull Down (Green Hashed) mode.The Metal1 layer (Blue) is used within the cells for interconnections.
- The Pull-Up/Pull-Down ratio of 10:1 is observed.



4 Input Nor Gate 3 Input NOR Gate

Layout 2: 3 and 4 Input NOR Gate

Using the MRN, it is relatively easy to obtain 3-Input and 4-Input NOR gates.

5.5 The Basic LCFL Cell

The LCFL logic family provides us with a basic logic block which can be modified to perform appropriate logic functions. The working of the LCFL cell was described in section 2.4.3. The diagram has been reproduced here for reference. One of the main features of this cell is the latch formed by combining the nor gate and the inverter. This latch can store the value present, and can be reset by setting the input logic such that it the point A can conduct to ground. Alternatively, setting the Request signal to high takes the circuit into Precharge cycle and setting it to low again takes to evaluate phase. As described earlier, the Complete signal goes high only after the output for the circuit has been computed.

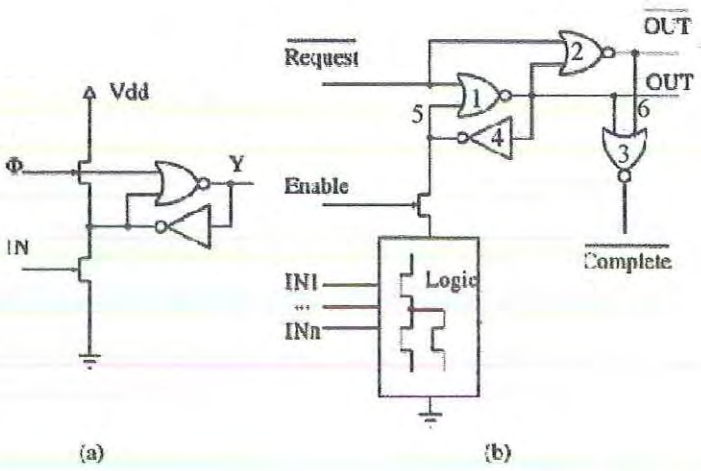
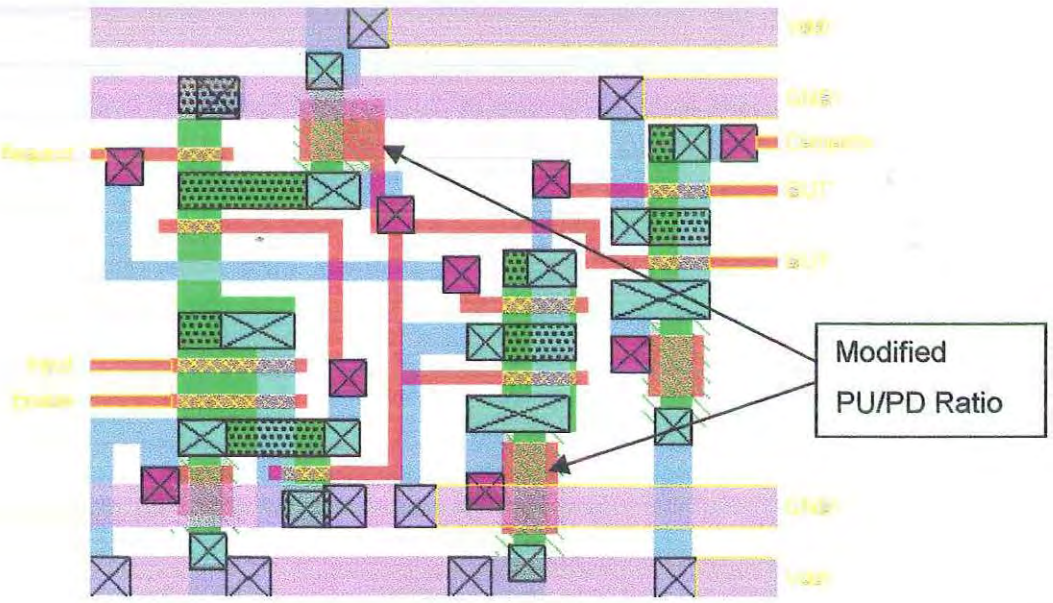


Figure 5.5: PDLL and LCFL logic cells. Adopted from Lachowicz et al (IEEE Transaction [1])



Lavout 2: Basic LCFL Cell

The layout above shows that the PU/PD ratio has been changed (from the normal DCFL value) for two of the transistors. The reason for this is that the sizing must be chosen appropriately so that the inputs reach the Nor Gate 3 at the same time. In the initial state when the request signal is '1' both gates will have an output '0', which is the reset mode. When the request signal goes to '0', and the Enable signal is '1', since output of the gate 1 is initially '0', gate 2 will output a '1' upon receiving two '0' inputs. Now if the input received by gate 1 from gate 5 is '1' then its output will remain '0' and so the output at gate 1 wouldn't change. However, if the input received by gate 1 from 5 is '0', then the output from gate 1 will go to ON, thus taking the output from 2 to OFF. If gate 1 takes more time to evaluate its output than gate 2, and if the output of gate 1 is '1' (after initially being in OFF state), then at the output of 2 there would be a 'glitch' simulating an '1' condition, before the returning to the correct output '0'. This in turn would mean that the gate 3 would receive a glitch of '1' before receiving the correct output of '0'. Note that eventually when the correct signal is received, this gate will still remain at '0', as OUT signal will turn to '1'. But in that case the Complete signal would have been set to '0', before the evaluation is finished. This is not as required by the circuit. Even though we can argue that the output of complete gate would eventually go to other parts of the circuit, such as Muller C cells, which would introduce enough delay anyway, this is not according to the principle of operation of the self-timed systems. Our approach ensures design of robust systems.

Thus a stronger pull down is required in gate 2, while stronger pull up is required in gate 1. According to suggestion from Stephan (sizing used in his designs), the PU/PD ratio of (1/2:4/1) and (1/3:6/1) for the gates 1 and 2 respectively was adopted. Note that this is a departure from the normal DCFL W/L ratio of 10:1

HSpice Simulation:

The HSpice simulation for the LCFL Basic cell has been shown on the next page.

Graph 1 of the simulation shows the Request signal which pulses from logic '1' to logic '0' for a period of 1ns including rising and falling times.

The Complete signal is generated in the circuit about 0.2ns after the request. Thus the computation time is 0.2ns.

Input and Enable are both in logic '1' state.

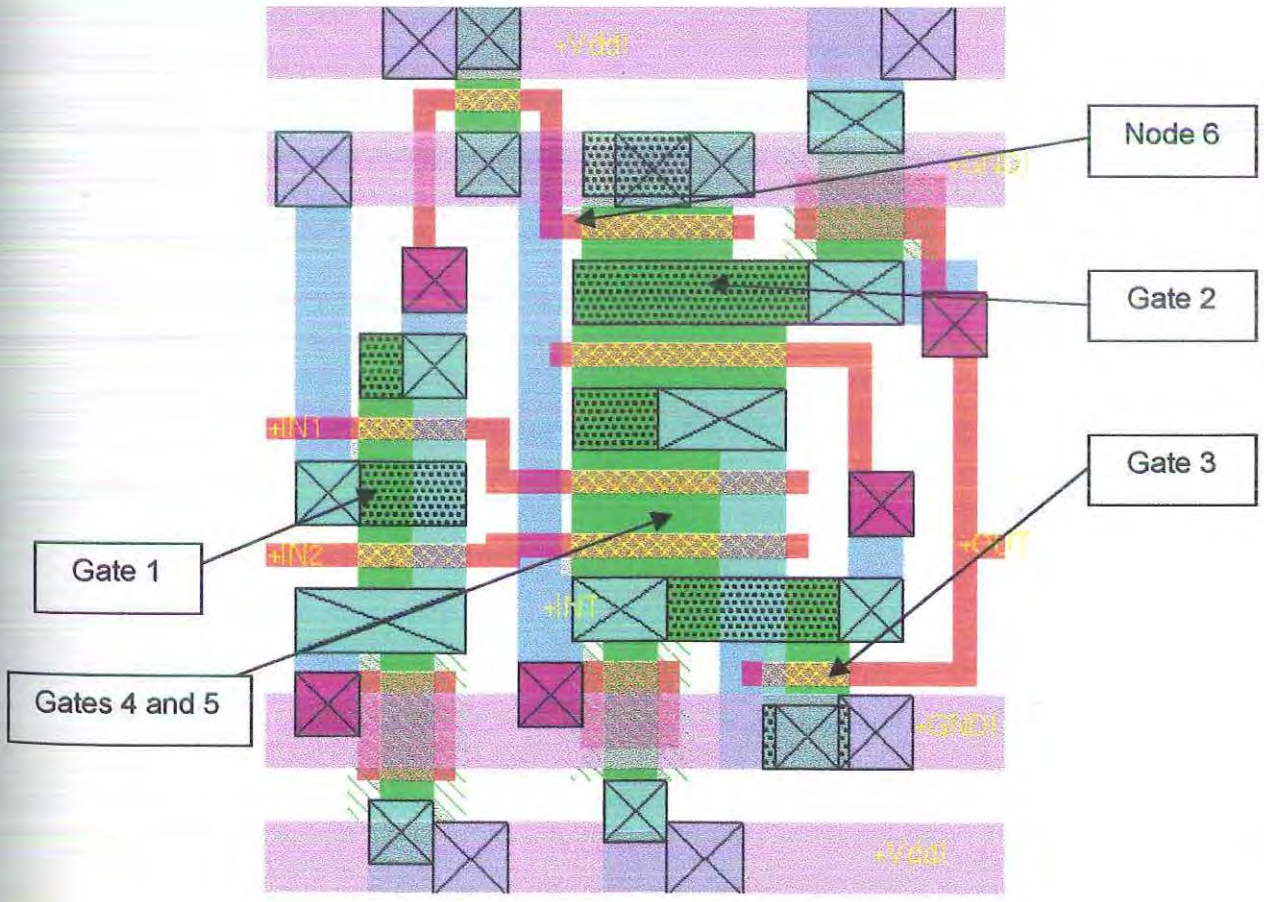
In graph 2, the Out and Out' signals are shown. As seen Out, follows the value of Input when request is low (evaluation phase). The Out' signal is shown to spike. This spike has been smoothened out due to the modified PU/PD ratio, as discussed earlier.

HSPICE SIMULATION FOR HANDSHAKE BLOCK: INPUTS
Wave
Symbol
D5:A0:v(100)
D5:A0:v(106)
D5:A0:v(110)
Voltages (lin)
600m
400m
200m
0
Complete n+1
Enable n
Enable n+1
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
4.5n
5n
5.5n
6n
6.5n
7n
7.5n
8n
Time (lin) (TIME)
OUTPUT: REQUEST N
D5:A0:v(101)
800m
600m
400m
200m

retained. ie If previous Output is '1' then the value at node 6 becomes '0', making the Output 1 (all three inputs to gate2 are '0'). Similarly, if previous Output is '0', then value at node 6 becomes '1', making the Output '0'. Thus, *the value in the circuit is unchanged*. This effect is called the 'memory' effect for the Muller C cell.

- When both IN1 and IN2 are '1', the output from NOT gate 1 is a logic '0'. Besides, since both the pass transistors the conducting, the voltage at node 6 is gounded to '0'. Thus all the inputs for gate 2 become '0', making the output '1'

The above conclusions can be summarised as "When inputs both are ON or OFF, the Ouput is ON or OFF respectively, else Output is unchanged". This is the essence of the operation of Muller C cell.

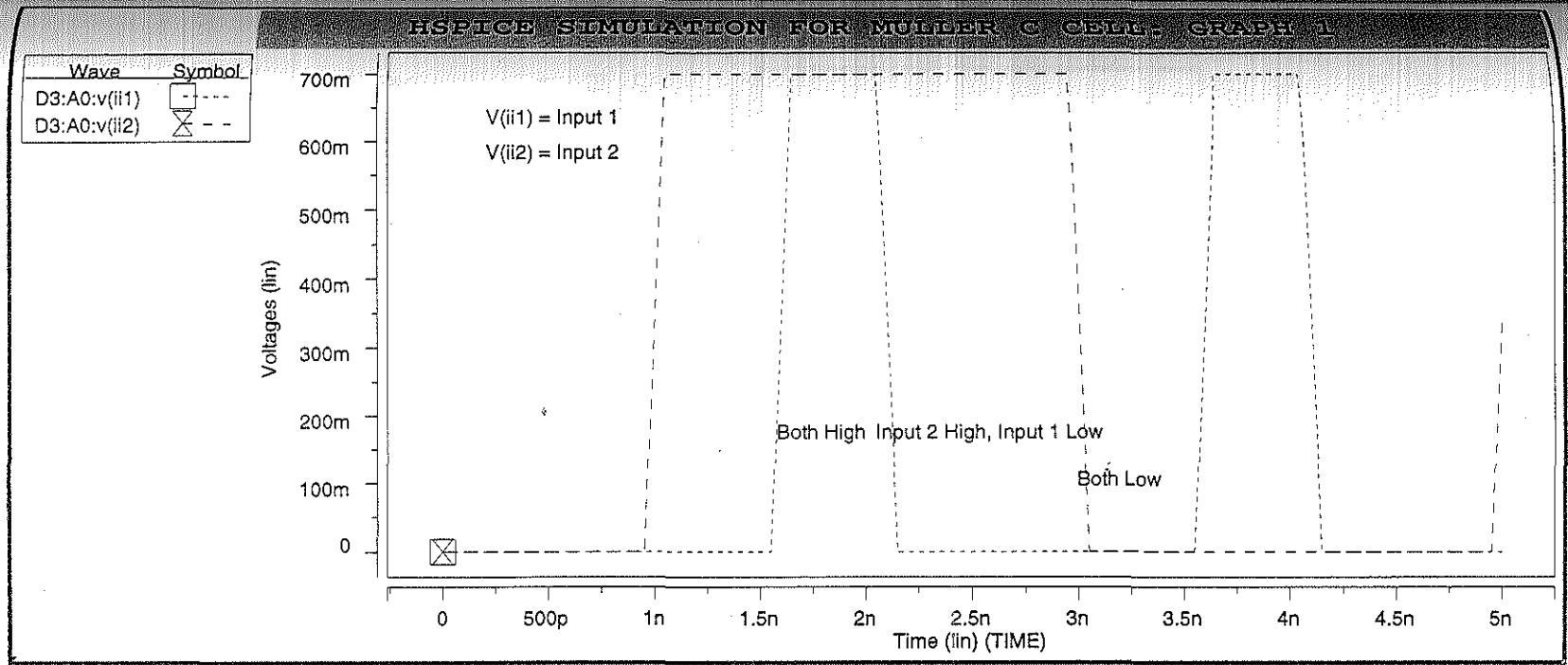


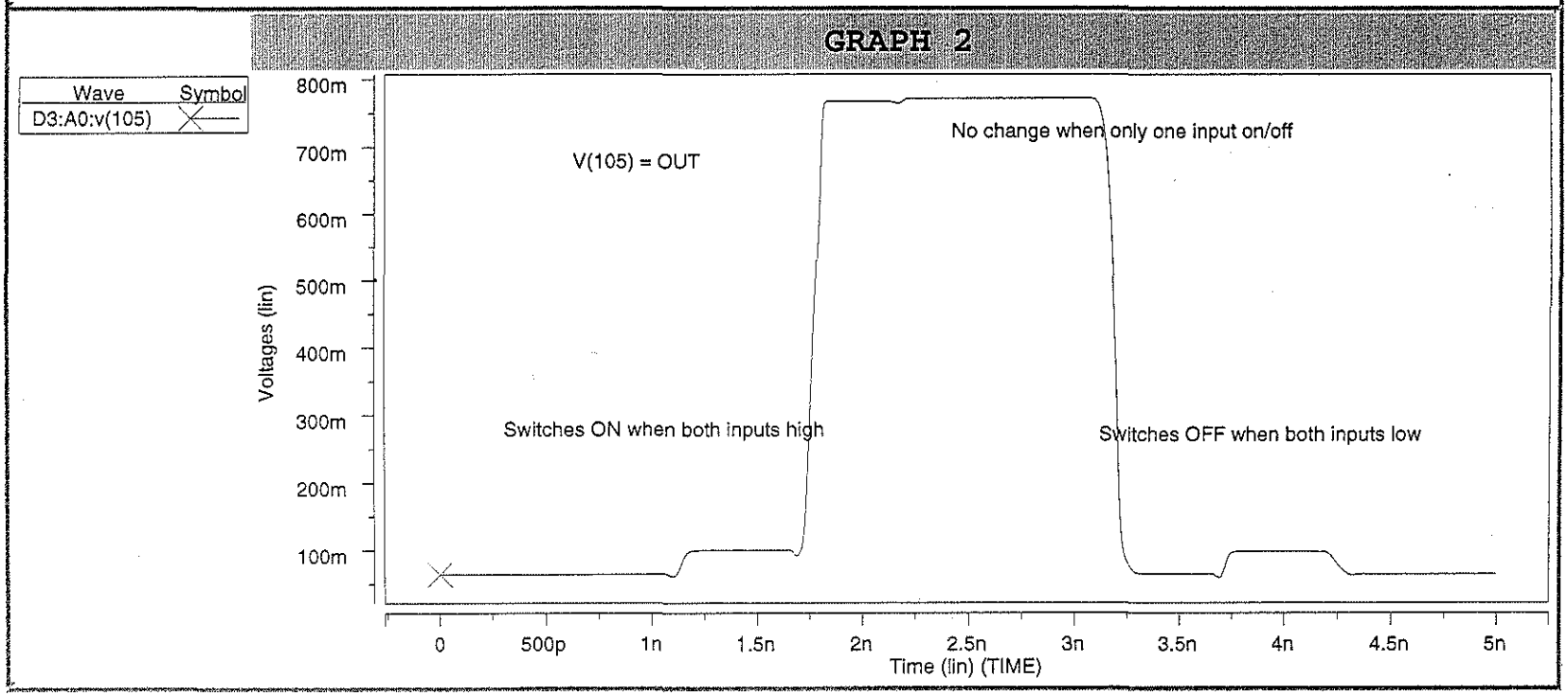
Layout 3: Muller C Cell

HSpice Simulation

The HSpice simulation for the Muller C cell has been shown on the next page.

- Graph 1 shows both the inputs. As seen, there are three different conditions with both inputs '1', both inputs '0' and one-high-one-low condition.
- The Output switches to '1' when both inputs are '1', switches to '0' when both inputs are '0', and remains unchanged when only when of them is in state '1'. This is as expected.
- The delay in the circuit is 0.25ns.
- Prior to simulation, an inverter was attached to the Output to simulate load conditions. Similarly two inverters each were also added to the inputs to 'smoothen' them, making them more realistic. This has been implemented in all the HSpice simulations for various circuits.


HSPICE SIMULATION FOR MULDER C CELL, GRAPH 1
Wave
Symbol
D3:A0:v(ii1)
D3:A0:v(ii2)
V(ii1) = Input 1
V(ii2) = Input 2
Both High
Input 2 High, Input 1 Low
Both Low
Voltages (lin)
700m
600m
500m
400m
300m
200m
100m
0
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
4.5n
5n
Time (lin) (TIME)


GRAPH 2
Wave
Symbol
D3:A0:v(105)
V(105) = OUT
No change when only one input on/off
Switches ON when both inputs high
Switches OFF when both inputs low
Voltages (lin)
800m
700m
600m
500m
400m
300m
200m
100m
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
4.5n
5n
Time (lin) (TIME)

5.7 Handshake Block 1

The handshake block is another important building block for self-timed designs. For our design, it has been utilised in the implementation of the bubble-shift register. Thus, we need to implement and test this cell before implementing a register cell.

Behaviour

The behaviour of the circuit can thus be described by the following truth table.

Enable n+1	Enable n	Complete n+1	Request n
0	0	0	Same as Previous
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Table 5.4. Behaviour of the Handshake Block 1

Design

Figure 5.7 shows the electrical layout for the Handshake cell.

- When Enable n or Complete n+1 signals are in 'logic 1' state, the request n signals goes low. However, when Enable n+1 signal is on and other signals are logic '0', the latch input conducts to ground and the Request n signal goes to logic '1'.
- This functionality of the handshake block is applicable in manipulating the request and complete signals between cells in the bubble register.

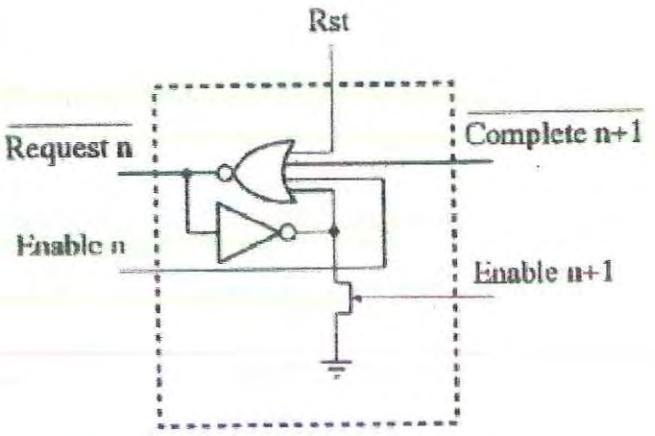
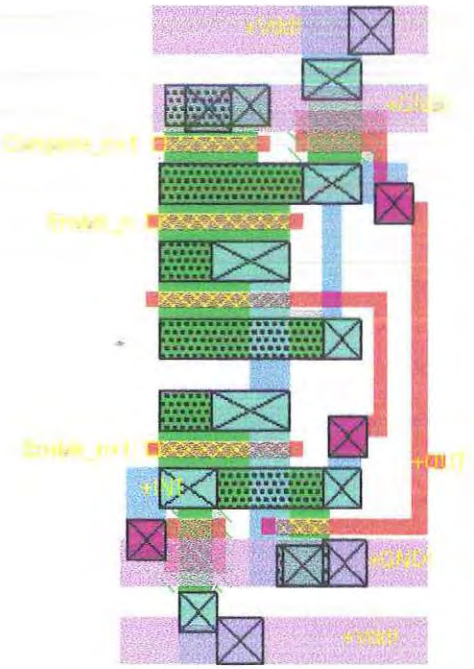


Figure 5.7: Handshake Block



Layout 4: Handshake Block

Hspice Simulation

The Hspice Simulation for the Handshake block has been shown on the next page.

- The Complete n+1, Enable n and Enable n+1 signals are pulsed at different intervals.
- The output switches on and off as per the requirements shown in the behaviour table.
- Response time for the circuit is 0.2ns.

5.8 The Register and Unit Register Cell

The design of a self-timed bubble-shift register is shown in figure 5.8. The Muller C cell, Handshake Block and LCFL basic cell have been implemented in the previous sections. As can be seen from the figure, it is a repetitive design, and each unit cell represents one bit storage on the register. The figure shows how the register is divided into its component unit cells. Thus, to implement this register, we need to design the unit register cell and replicate it by abutting the required number of such cells together.

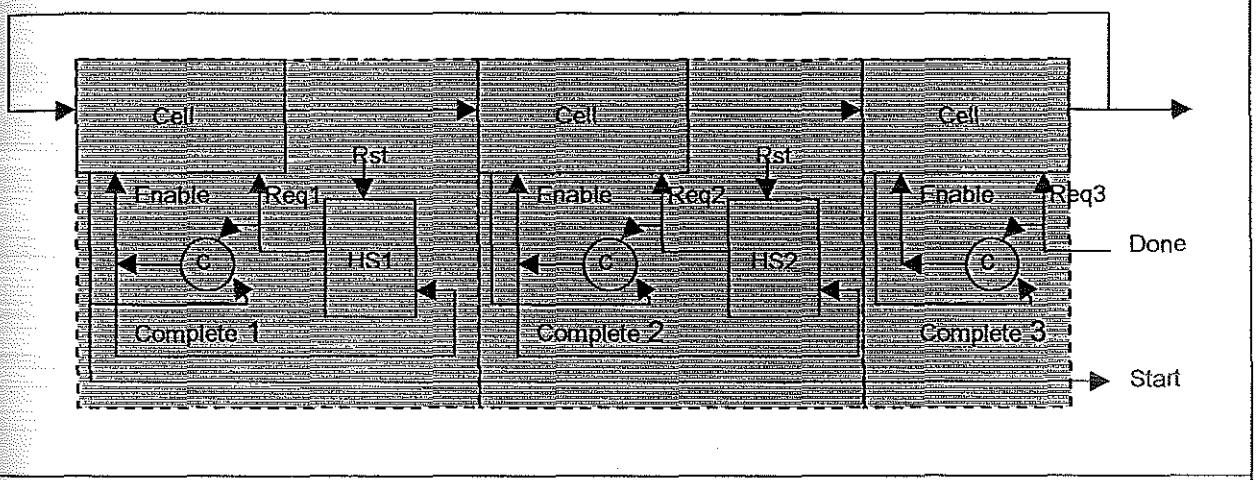


Figure 5.8 Self-timed bubble shift register. Adopted from Lachowicz et al (IEEE Transaction [1])

Details of Data Transfer through the Register

- Initially, the Request n at the right most cell of the register is low.
- After the start signal is received at Cell 3, this is processed through the Muller C element and a request is generated for the LCFL cell, which transfers data from left to right.
- The complete signal from the LCFL cell then interacts with the handshake block of the next cell on the left, thus eventually generating a request signal that cell.
- In this manner, the request is transferred from right to left. Note that the direction of flow of data is opposite to direction of flow of control.
- There is always one 'non-relevant' cell in the register. This is called the bubble. Initially the bubble is at cell 3. As the request signals progress further, the bubble moves from right to left and data moves from left to right. Hence the name bubble shift register.

Design the of Unit Register Cell

All the component cells of the register cell – the Muller C, Handhake Block and a LCFL storage cell – have been implemented. Thus to implement the register cell, it is required to connect these in a correct manner. The figure 5.9 is a schematic representation of the layout, and shows the interconnections as they occur in the actual layout.

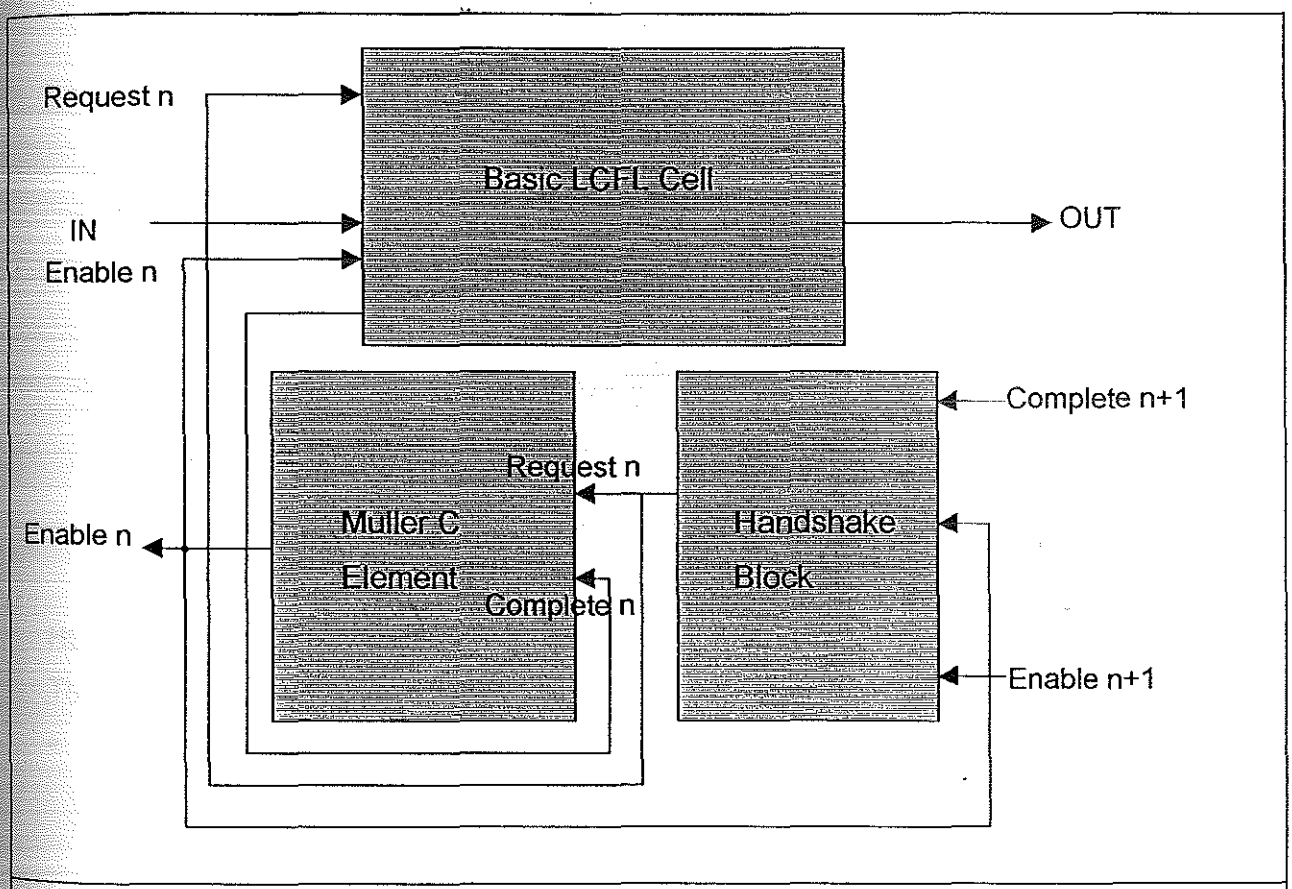
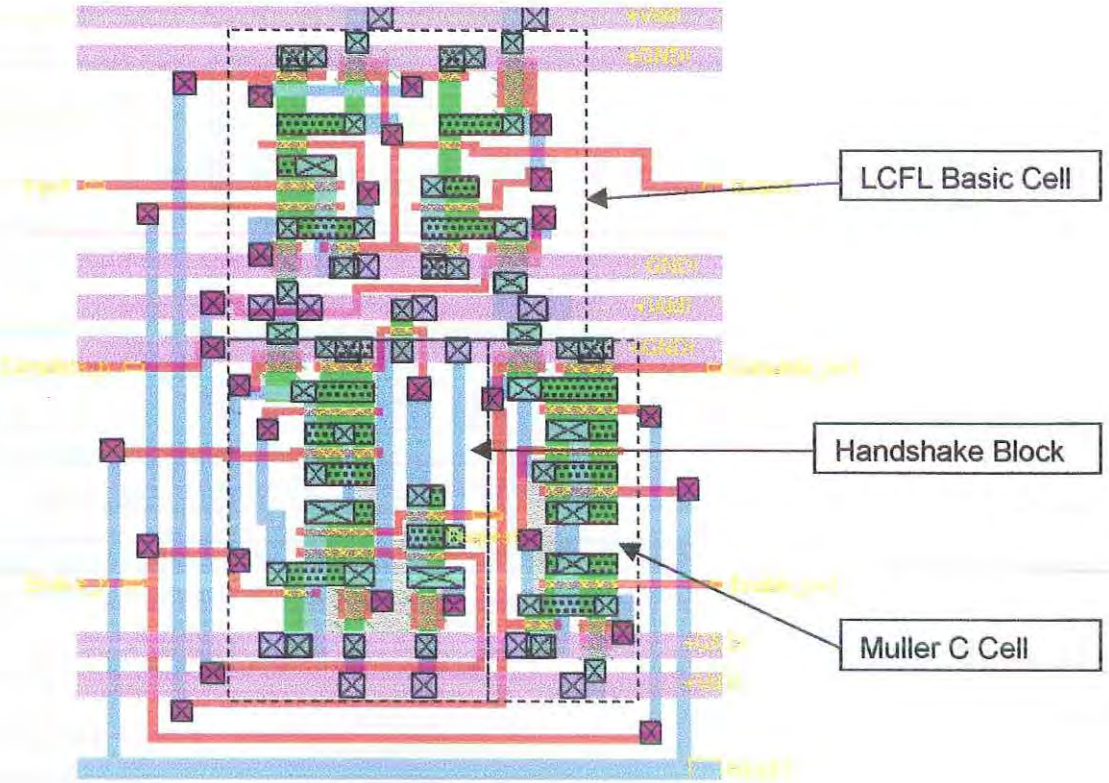


Figure 5.9. Schematic MAGIC Layout for Register Cell



Layout 5: Unit Register Cell

HSpice Output for the Register

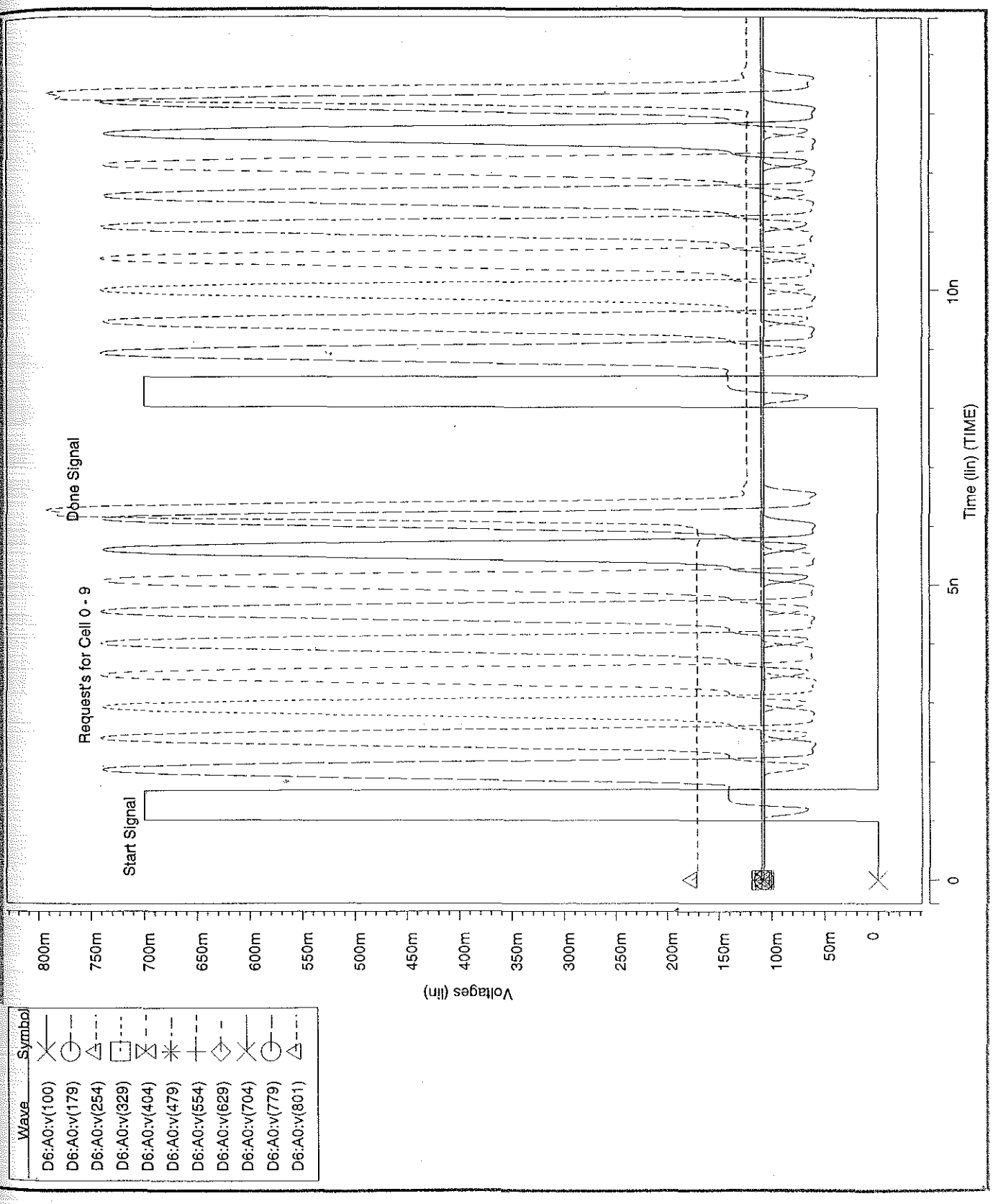
The HSpice simulations for Register are shown on the next two pages.

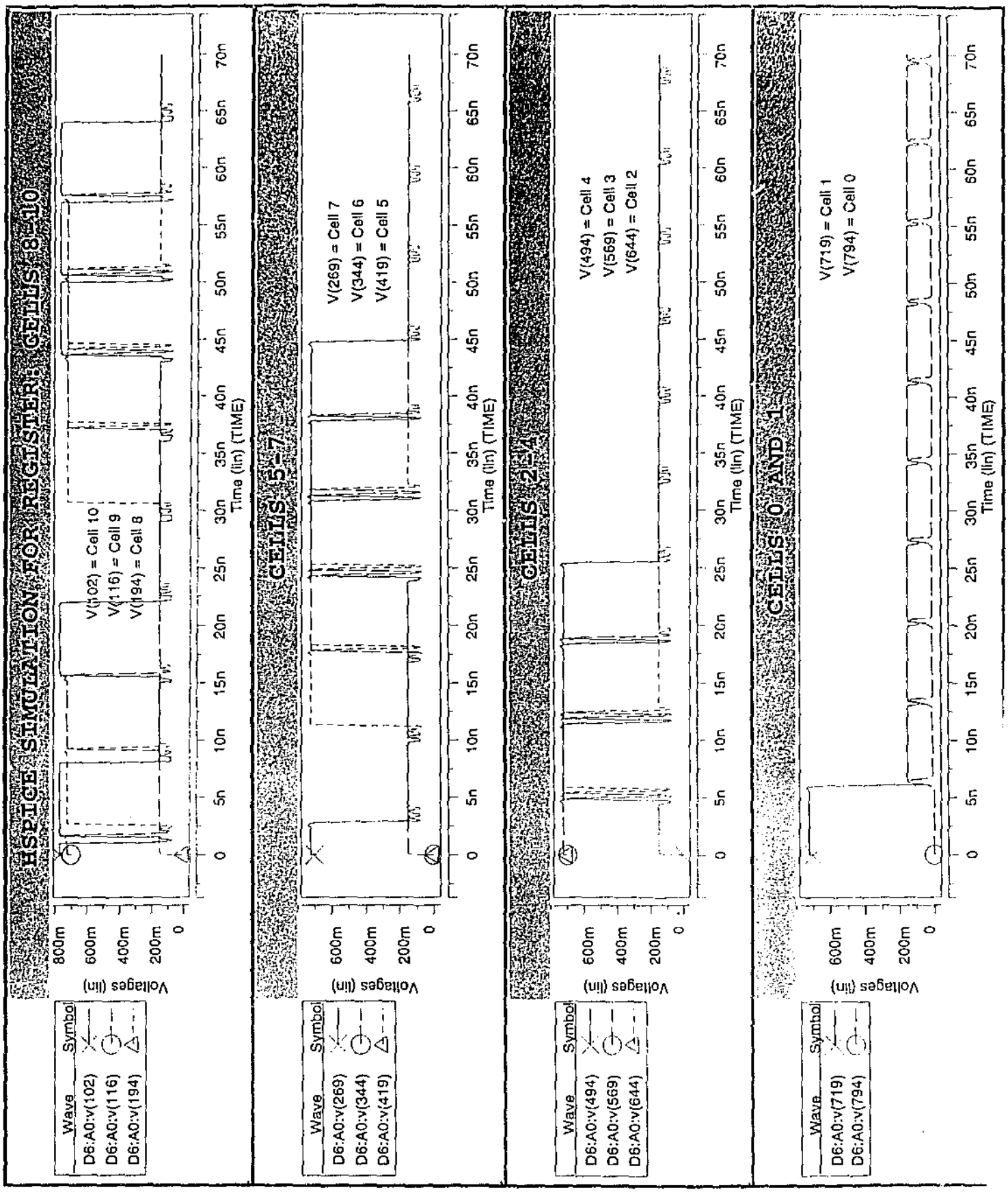
- The first graph shows the control signals. As seen a start signal is generated, which triggers the system start. As cells 0 to 9 each receive request signals and output data, they generate a request signal for the next cell. The request signals for cells 0 – 9 are shown in the graph. The Complete signal generated from the last cell acts as Done for the register.
- The response time, from Start to Done, is 4.6ns.
- The next graph shows how bits are shifted from left to right in the register. As seen, the original contents of the register were from cells 0 – 9 were 0 1 1 1 0 0 0 1 0 1. Thus the cell 9, for example would have the data 1 0 1 0 0 0 1 1 1 0 in it when data is shifted 10 times. This is reflected in the graph for cells 8 – 10. Similarly, other cells would also go through different logic states, as shown in the table below.

	Shift 0	Shift 1	Shift 2	Shift 3	Shift 4	Shift 5	Shift 6	Shift 7	Shift 8	Shift 9
Cell 0	0	0	0	0	0	0	0	0	0	0
Cell 1	1	0	0	0	0	0	0	0	0	0
Cell 2	1	0	1	0	0	0	0	0	0	0
Cell 3	1	0	1	0	0	0	0	0	0	0
Cell 4	1	0	1	0	0	0	0	0	0	0
Cell 5	1	0	1	0	0	0	0	0	0	0
Cell 6	1	0	1	0	0	0	1	0	0	0
Cell 7	1	0	1	0	0	0	1	1	0	0
Cell 8	1	0	1	0	0	0	1	1	0	0
Cell 9	1	0	1	0	0	0	1	1	1	0

Table 5.5. Logic States for different cells after shift operations

These results were reflected in the HSpice graphs.


HSpice Simulation For Register Transfer Timing Signals
Done Signal
Request's for Cell 0 - 9
Start Signal
Voltages (lin)
Time (lin) (TIME)
Wave
Symbol
D6:A0:v(100)
D6:A0:v(179)
D6:A0:v(254)
D6:A0:v(329)
D6:A0:v(404)
D6:A0:v(479)
D6:A0:v(554)
D6:A0:v(629)
D6:A0:v(704)
D6:A0:v(779)
D6:A0:v(801)


HSPICE SIMULATION FOR REGISTER CELLS 8-10
V(102) = Cell 10
V(116) = Cell 9
V(194) = Cell 8
CELLS 5-7
V(269) = Cell 7
V(344) = Cell 6
V(419) = Cell 5
CELLS 2-4
V(494) = Cell 4
V(569) = Cell 3
V(644) = Cell 2
CELLS 0 AND 1
V(719) = Cell 1
V(794) = Cell 0
Wave
Symbol
D6:A0:v(102)
D6:A0:v(116)
D6:A0:v(194)
D6:A0:v(269)
D6:A0:v(344)
D6:A0:v(419)
D6:A0:v(494)
D6:A0:v(569)
D6:A0:v(644)
D6:A0:v(719)
D6:A0:v(794)
Voltages (lin)
Time (lin) (TIME)

5.9 The 2 Input Multiplexer

The 2 Input Multiplexer is a relatively easy design. It is being used in our design to perform the 'Bypass' function. It will also form a building block for the 4 input multiplexer, as we shall see later.

Behaviour

The principle of operation of a 2 input multiplexer is very simple. It selects between two inputs based on the value of another control input. The truth table below shows this function.

IN0	IN1	Control	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Table 5.6. Function of a 2-Mux

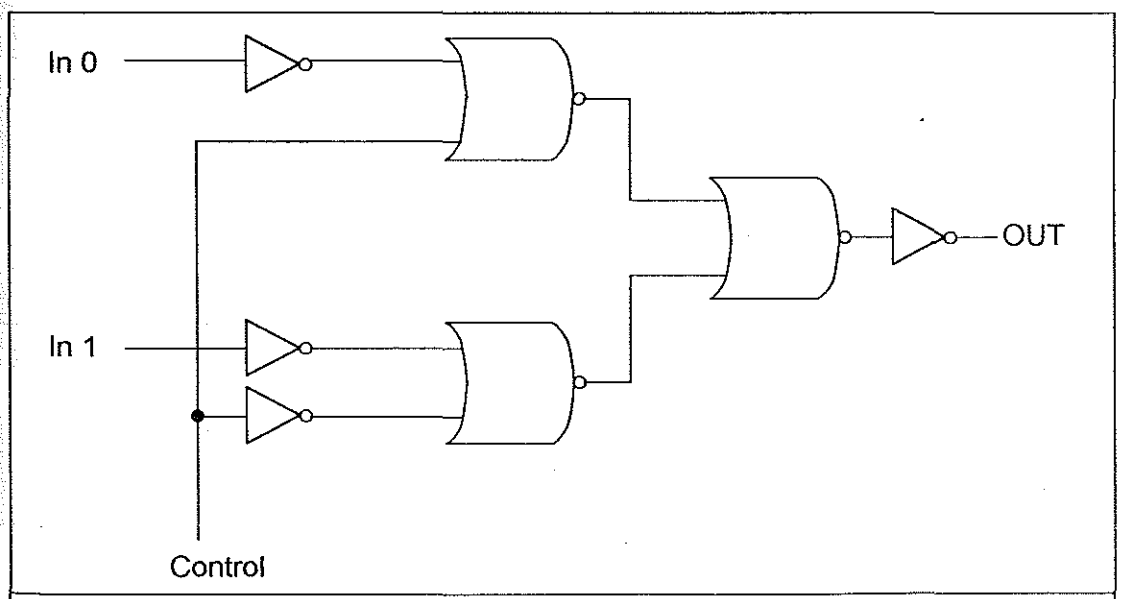


Figure 5.10. Electrical Layout for 2 Input Multiplexer

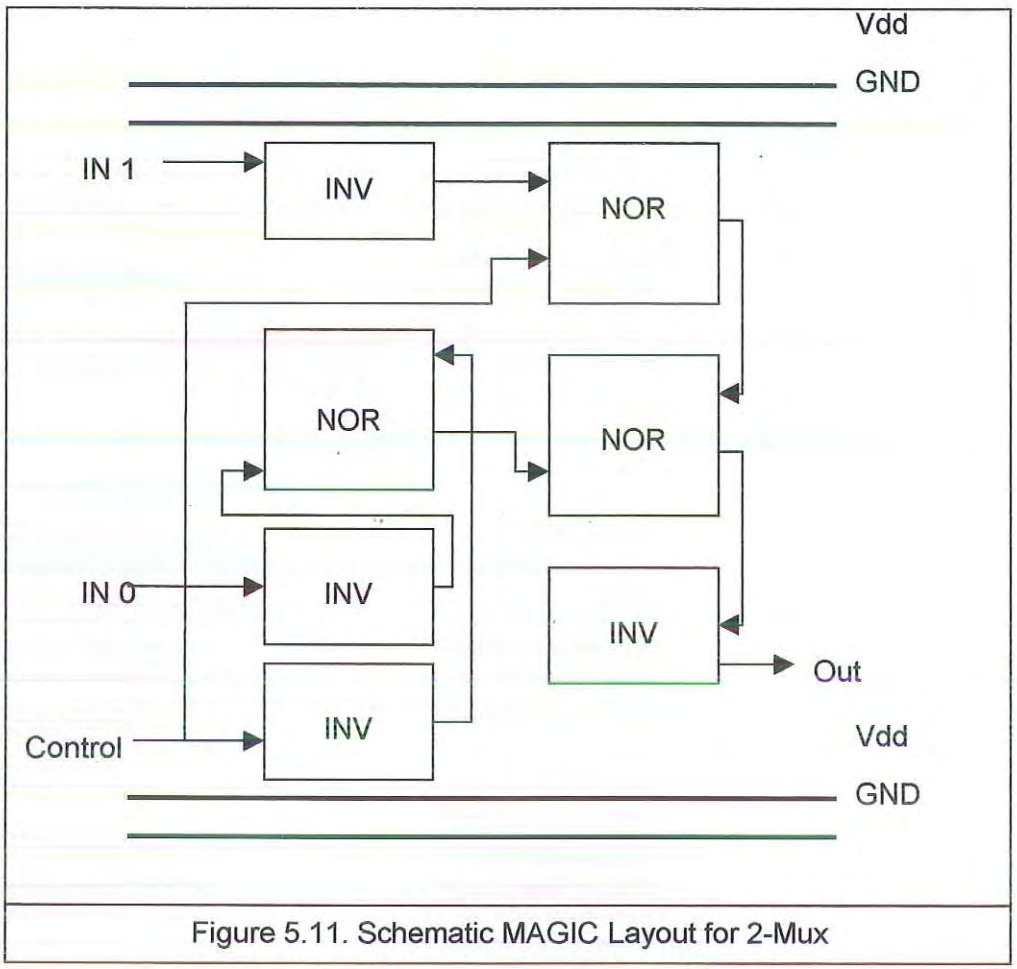
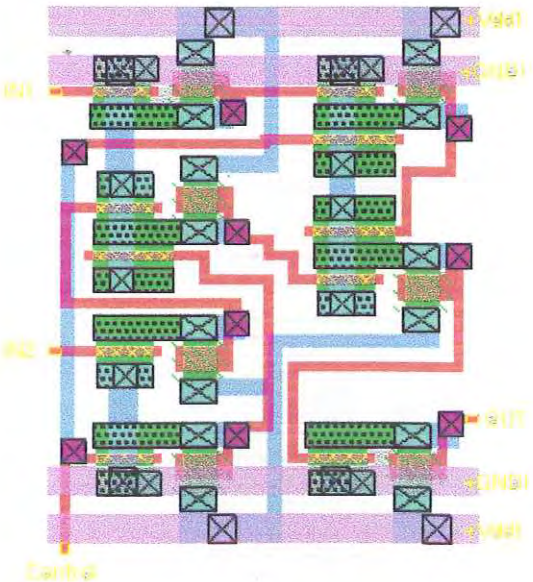


Figure 5.11. Schematic MAGIC Layout for 2-Mux



Layout 6: 2 Input Multiplexer

Design

The design of the 2 input multiplexer, although intuitive, is made complex due to the fact that GaAs designing allows us to use only nor gates and inverters. The electrical layout of the multiplexer is as shown in figure 5.11.

HSpice Outputs

HSpice simulation for the And cell is shown on the next page.

- The control is pulsed between '0' and '1'.
- The output follows one of the inputs 1 and 2, based on the control signal.
- Response time in the circuit is 2ns.

HSPICE SIMULATION FOR 2INP MUX: INPUT AND CONTROL
Wave
Symbol
D3:A0:v(100)
D3:A0:v(129)
D3:A0:v(112)
Voltages (lin)
600m
400m
200m
0
Input 1
Input 2
Control
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
4.5n
5n
Time (lin) (TIME)

OUTPUT
Wave
Symbol
D3:A0:v(139)
Voltages (lin)
800m
700m
600m
500m
400m
300m
200m
100m
Out = Input 2
Out = Input 1
Out = Input 2
Out = Input 1
Out = Input 2
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
4.5n
5n
Time (lin) (TIME)

5.10 The 4 Input Multiplexer

The 4 Input Multiplexer is a very widely used design component. As shown earlier, our design required two 4 Input Multiplexers.

Behaviour

A 4 Input Multiplexer selects between 4 different inputs, based on the value of a 2 bit control signal. The selected input then becomes the output. The truth table below shows the function of the Multiplexer.

IN0	IN1	IN2	IN3	C0	C1	Output
0				0	0	0
1				0	0	1
	0			0	1	0
	1			0	1	1
		0		1	0	0
		1		1	0	1
			0	1	1	0
			1	1	1	1

Shaded Area shows that Value is Inconsequential

Table 5.7. Truth Table for 4 Input Multiplexer.

Design

There were two different approaches taken in implementing the 4-Input Multiplexer. Figures 5.13 and 5.14 show the two different designs. The first design is easier to implement as it utilises the previously designed 2 input multiplexer. However, the second design is faster and more compact. The second design utilises 3-input and 4-input NOR gates.

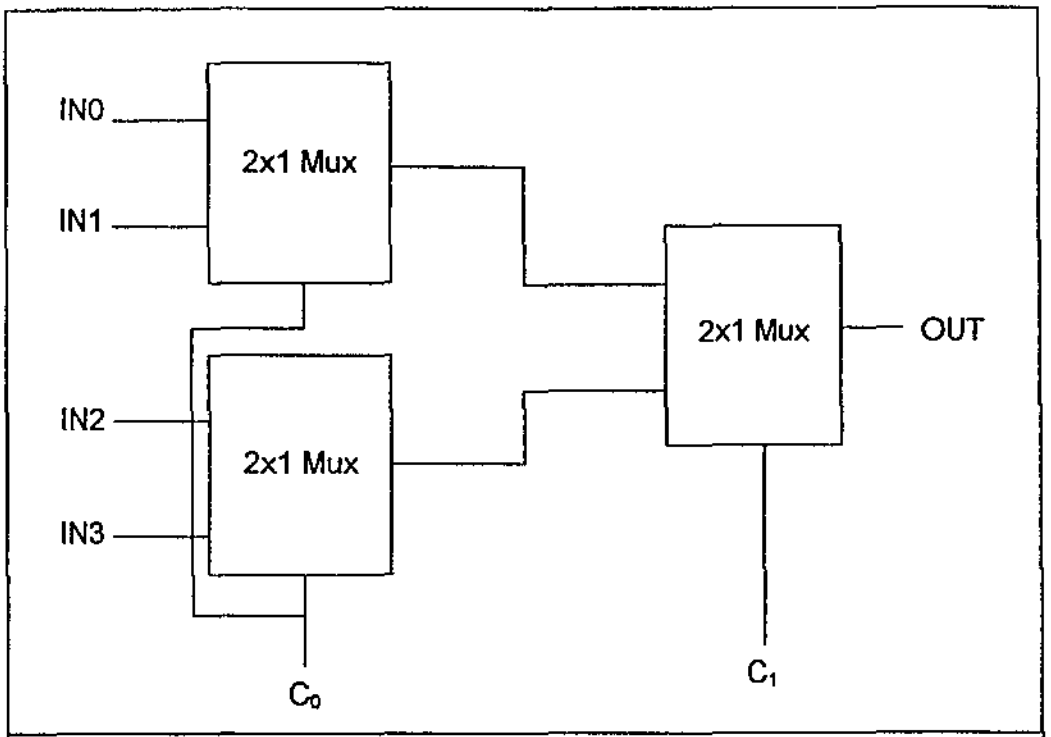


Figure 5.12. Design a 4-Mux using a 2-Mux

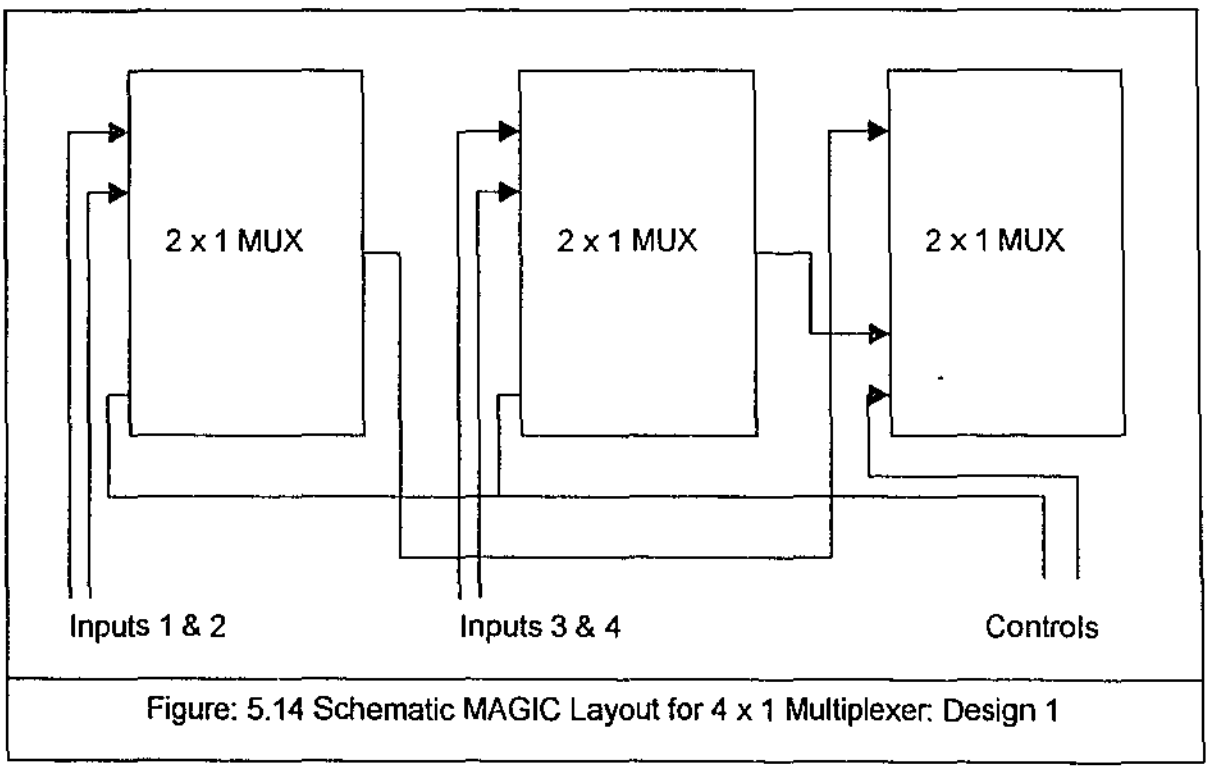


Figure: 5.14 Schematic MAGIC Layout for 4 x 1 Multiplexer: Design 1

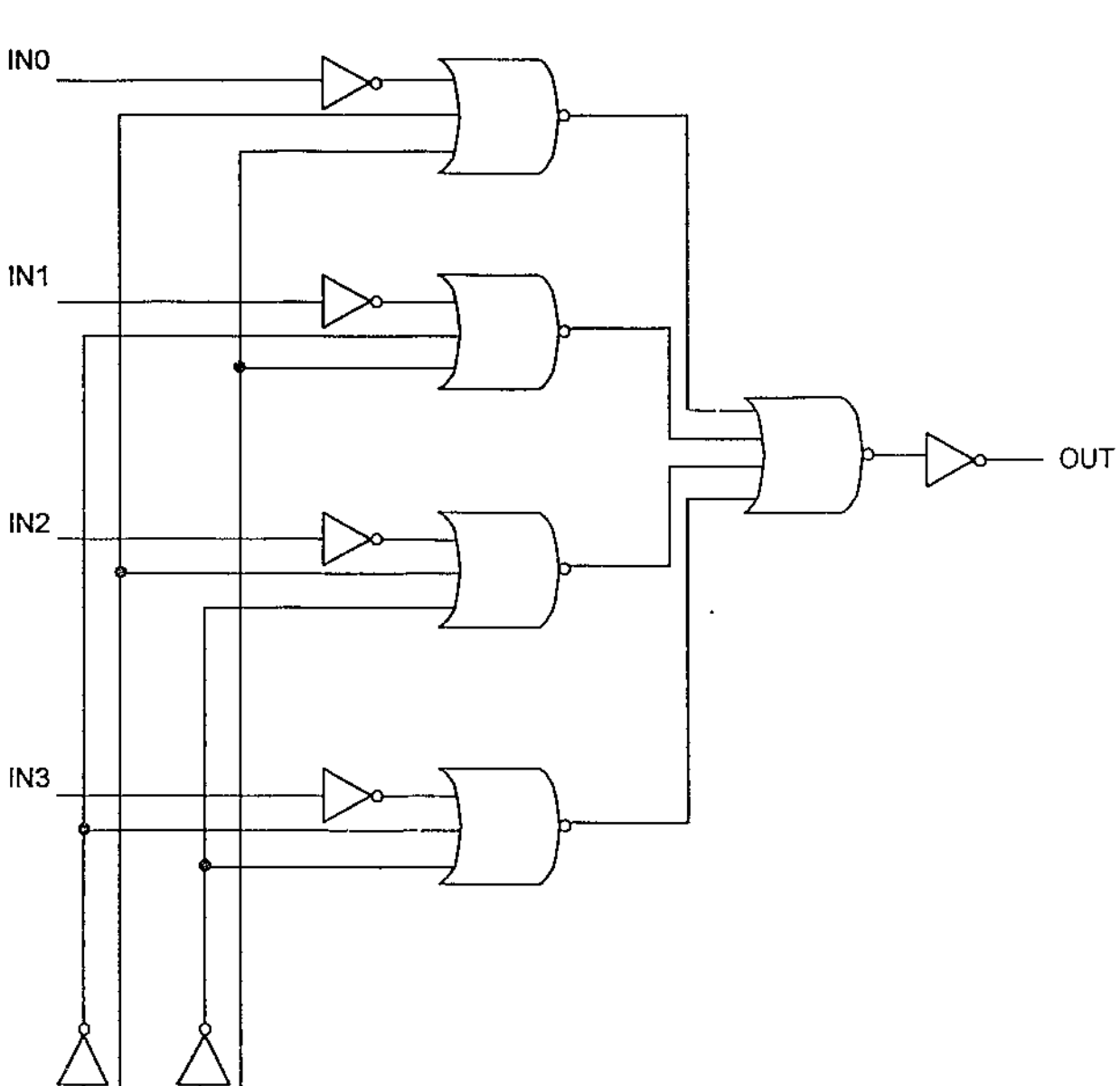


Figure 5.13. Design of 4 Input Multiplexer using 3-Inp and 4-Inp NOR Gates

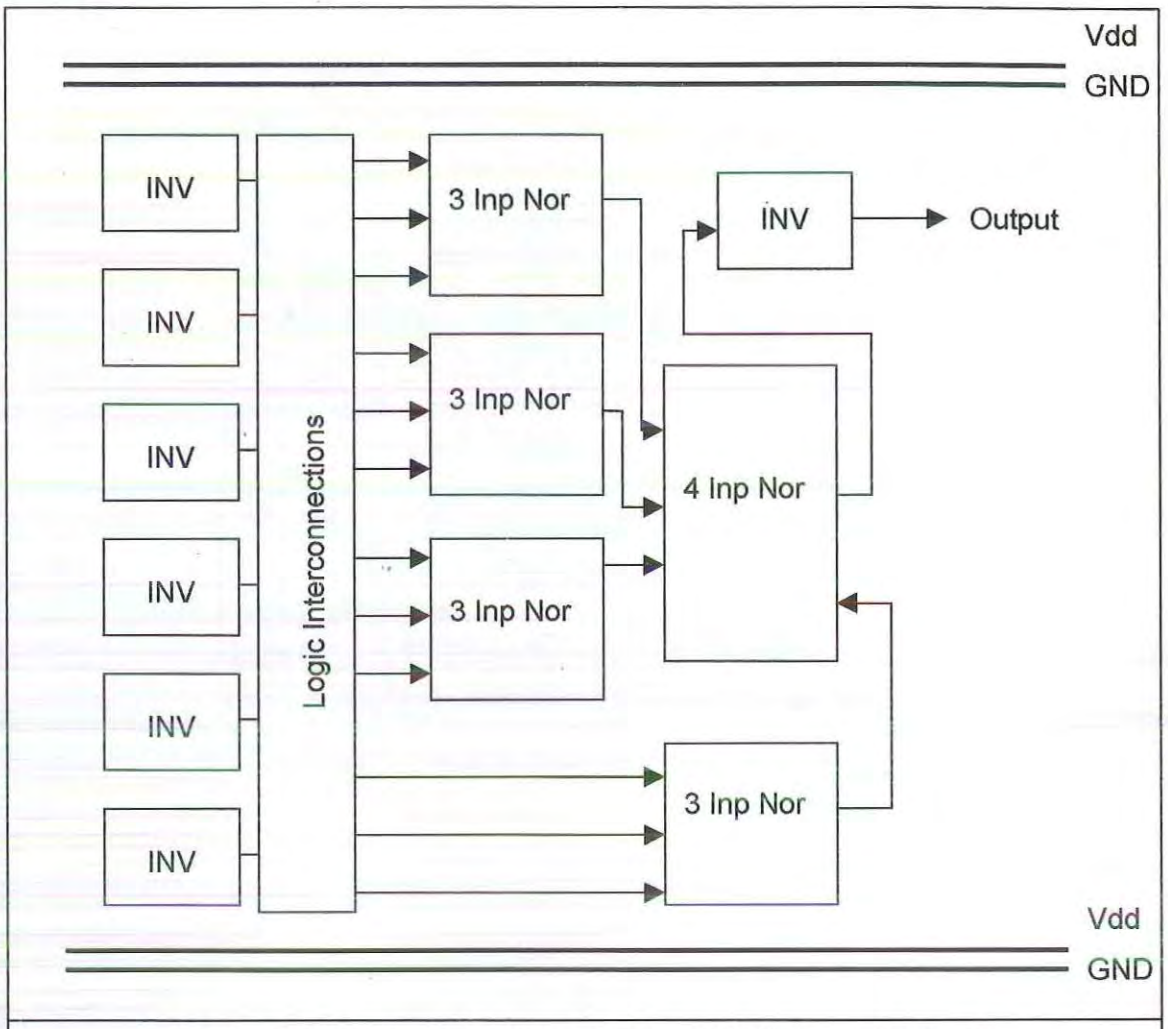
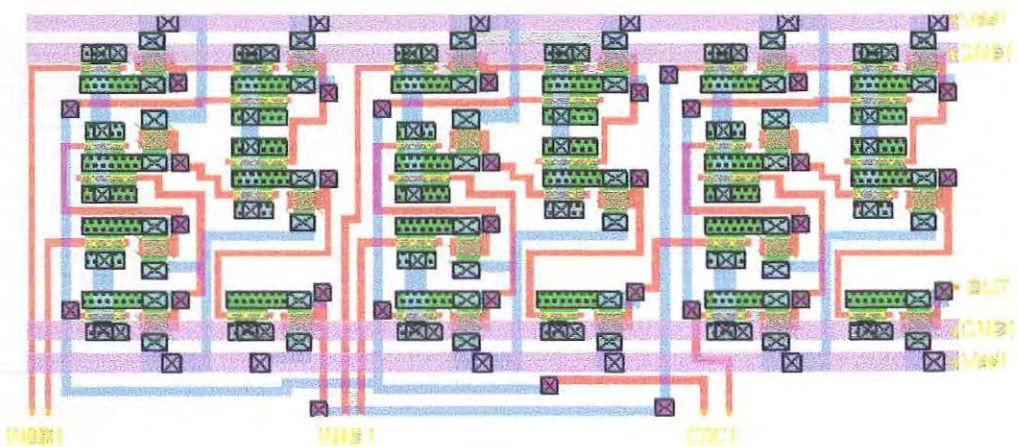
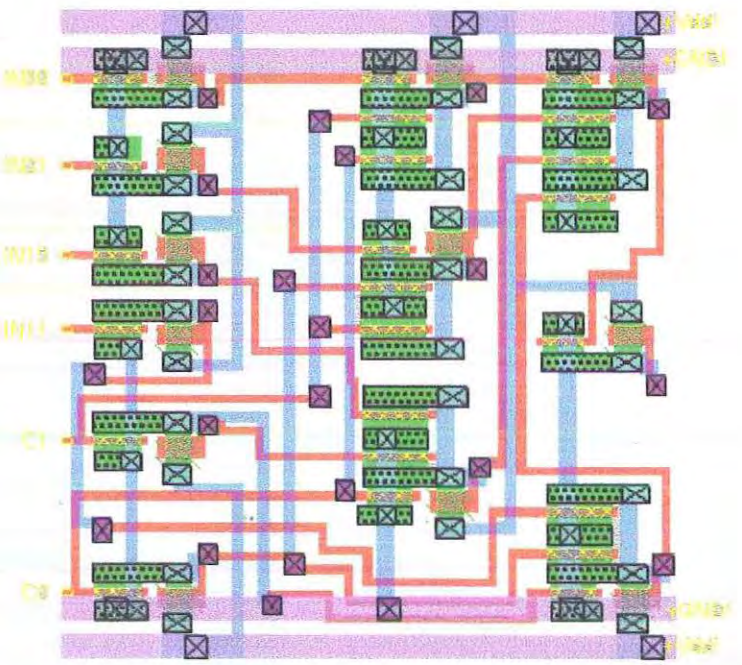


Figure 5.15: Schematic Layout of 4-Input Multiplexer. Design 2



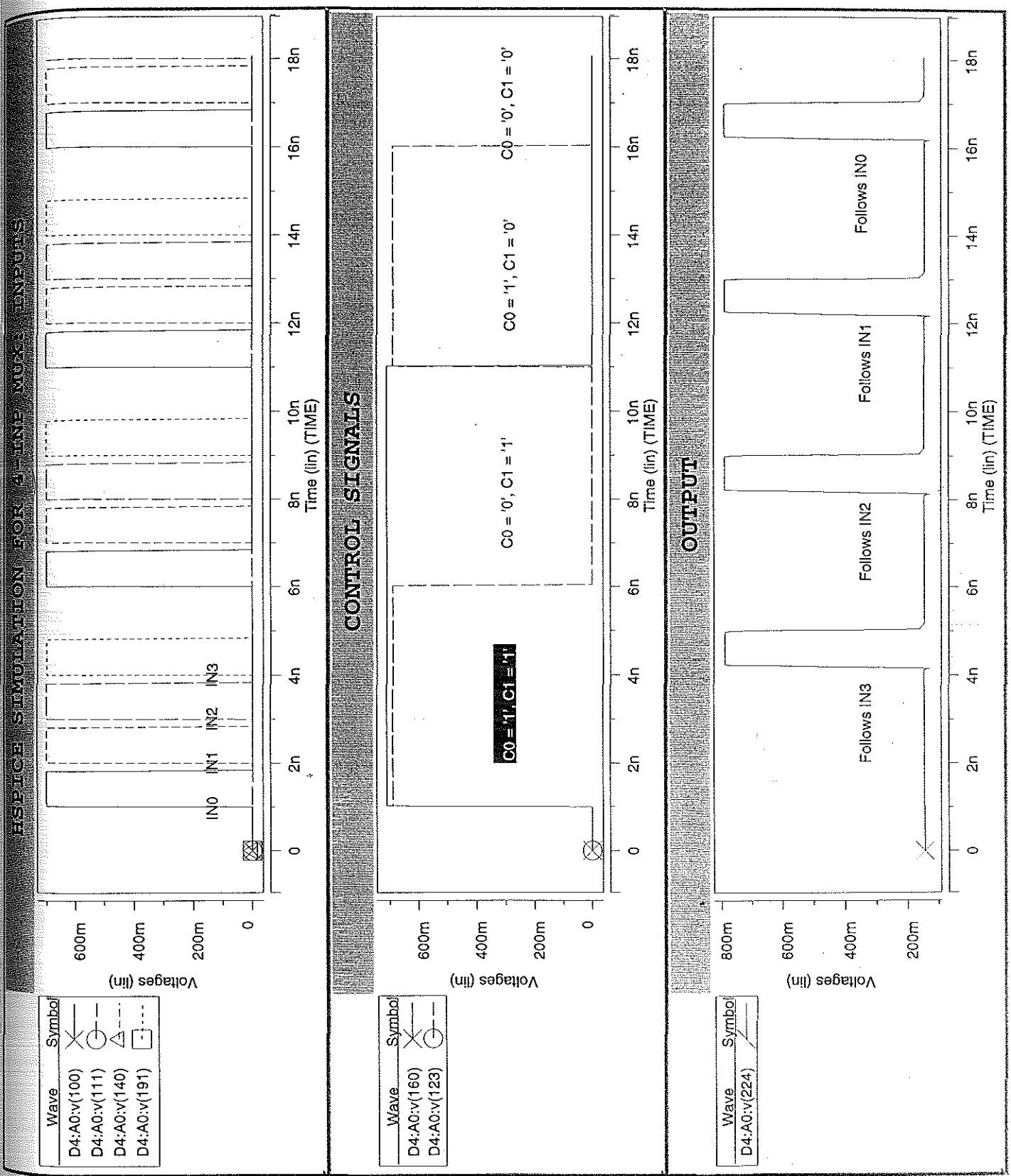
Layout 7: 4 Input Multiplexer: Design 1

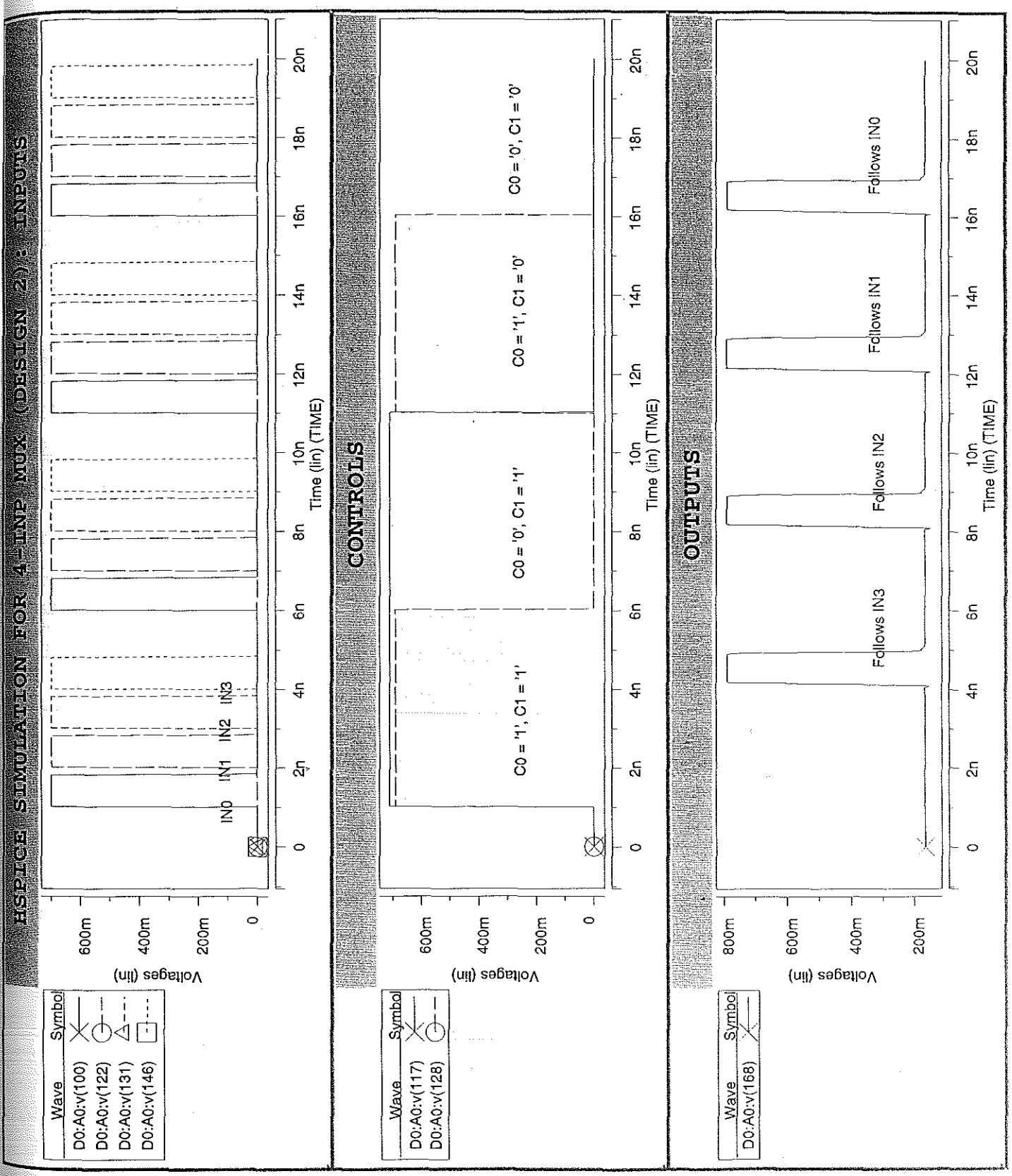


Layout 8: 4 Input Multiplexer: Design 2

The HSpice simulations for both the designs of 4-Input Multiplexer are shown on the next two pages.

- The control signals C0 and C1 are pulsed to achieve four different combinations 00, 01, 10 and 11.
- For each combination of the control signal, the four input signals are pulsed one at a time.
- The output follows the appropriate signal in both the graphs, as shown.


HSPICE SIMULATION FOR 4-IND MUX'S INPUTS
Wave Symbol
D4:A0:v(100)
D4:A0:v(111)
D4:A0:v(140)
D4:A0:v(191)
Voltages (lin)
600m
400m
200m
0
IN0
IN1
IN2
IN3
0
2n
4n
6n
8n
10n
12n
14n
16n
18n
Time (lin) (TIME)
CONTROL SIGNALS
Wave Symbol
D4:A0:v(160)
D4:A0:v(123)
C0 = '1', C1 = '1'
C0 = '0', C1 = '1'
C0 = '1', C1 = '0'
C0 = '0', C1 = '0'
OUTPUT
Wave Symbol
D4:A0:v(224)
800m
Follows IN3
Follows IN2
Follows IN1
Follows IN0


HSPICE SIMULATION FOR 4 I/P MUX (DESIGN 2) - INPUTS
Wave Symbol
D0:A0:v(100)
D0:A0:v(122)
D0:A0:v(131)
D0:A0:v(146)
Voltages (lin)
600m
400m
200m
0
IN0
IN1
IN2
IN3
0
2n
4n
6n
8n
10n
12n
14n
16n
18n
20n
Time (lin) (TIME)
CONTROLS
Wave Symbol
D0:A0:v(117)
D0:A0:v(128)
Voltages (lin)
600m
400m
200m
0
C0 = '1', C1 = '1'
C0 = '0', C1 = '1'
C0 = '1', C1 = '0'
C0 = '0', C1 = '0'
Time (lin) (TIME)
OUTPUTS
Wave Symbol
D0:A0:v(168)
Voltages (lin)
800m
600m
400m
200m
Follows IN3
Follows IN2
Follows IN1
Follows IN0
Time (lin) (TIME)

5.11 The AND Cell

This cells performs the function of a combinational AND gate. This cell needs to be designed explicitly as there is no direct implementation of an AND gate in GaAs.

Behaviour

As per the functional requirements of an AND gate, the output is a logic '1' *only* when both inputs are in logic '1' state.

Design

An AND in GaAs is implemented using two inverters and a nor gate. This design uses the De Morgan's principle: A . B = (A' + B')' . The circuit design follows naturally from the above equation.

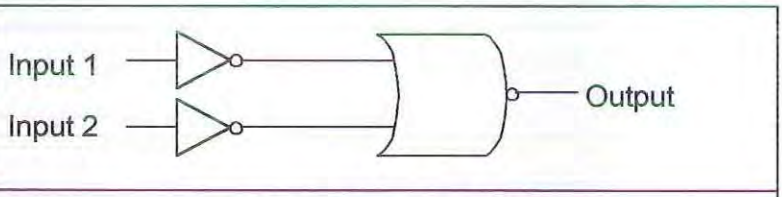
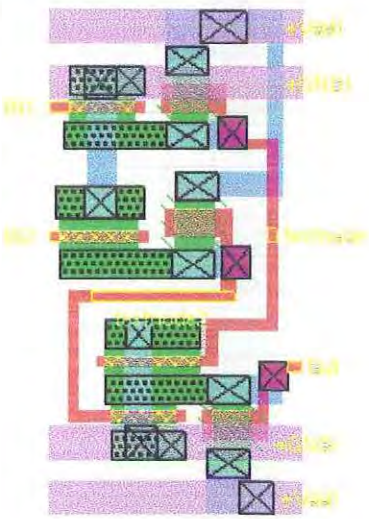


Figure 5.16. An AND Gate

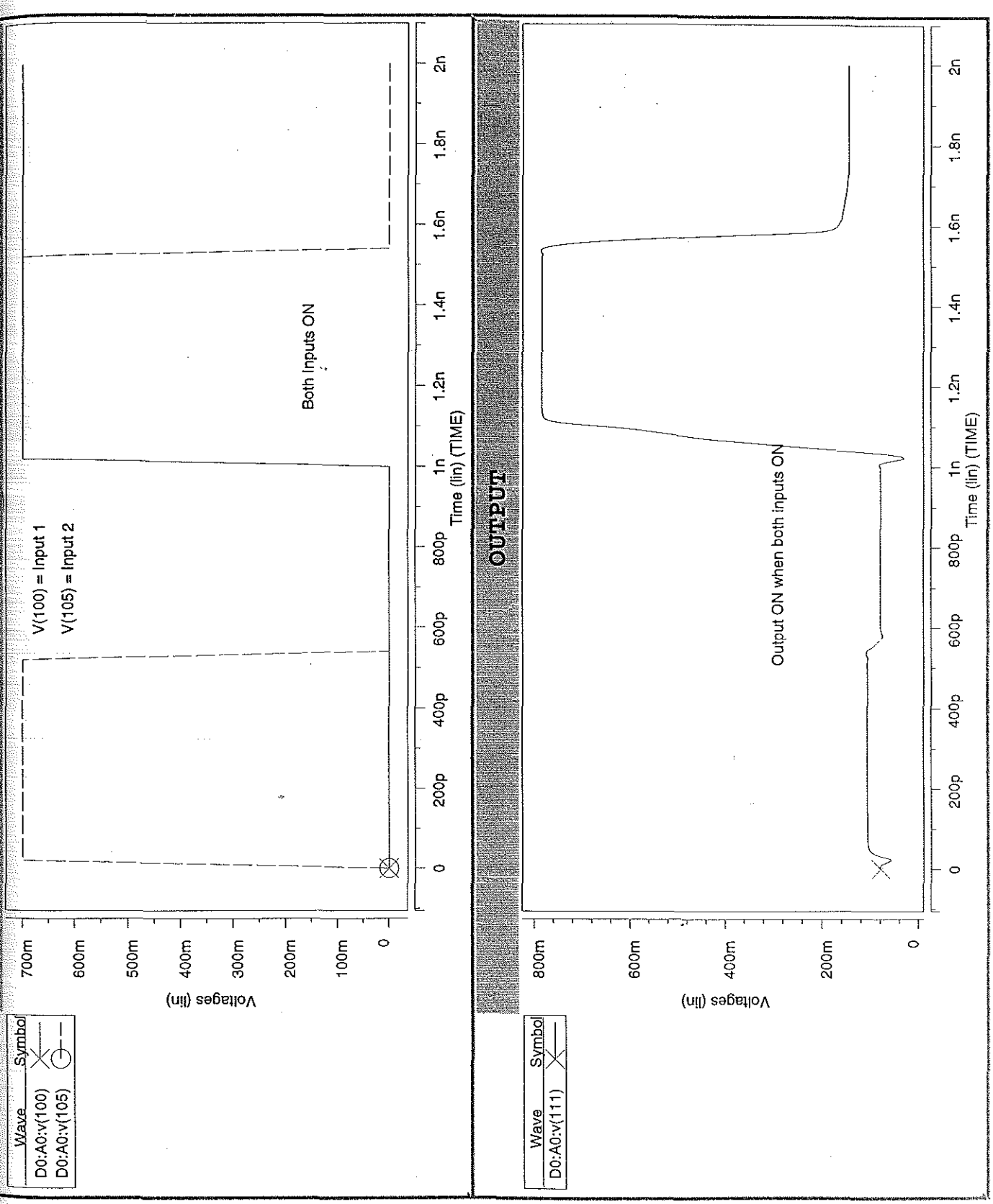


Layout 9: And Cell

HSpice Simulation

HSpice simulation for the And cell is shown on the next page.

- The output is '1' when both the inputs are on.
- The response time in the circuit is 2ns.


HSPICE SIMULATION FOR AND CIRCUIT : INPUTS
Wave Symbol
D0:A0:v(100)
D0:A0:v(105)
Voltages (lin)
700m
600m
500m
400m
300m
200m
100m
0
V(100) = Input 1
V(105) = Input 2
Both Inputs ON
0 200p 400p 600p 800p 1n 1.2n 1.4n 1.6n 1.8n 2n
Time (lin) (TIME)
OUTPUT
Wave Symbol
D0:A0:v(111)
Voltages (lin)
800m
600m
400m
200m
0
Output ON when both inputs ON
0 200p 400p 600p 800p 1n 1.2n 1.4n 1.6n 1.8n 2n
Time (lin) (TIME)

5.12 The XOR Cell

An XOR gate is a very commonly used gate in logic designing. It can conventionally be treated as a standard gate. However, using only nor gates and inverters, the circuit becomes slightly more complex. The XOR cell used for the project was a simple combinational implementation; a self-timed gate was not perceived as necessary.

Behaviour

The function of an XOR cell is very well known. It is shown by the truth table below.

Input 1	Input 2	O/P
0	0	0
0	1	1
1	0	1
1	1	0

Table 5.8. An XOR Gate

Design

The electrical layout for an XOR cell is shown below. Note that the design consists of an AND gate, which has been designed previously.

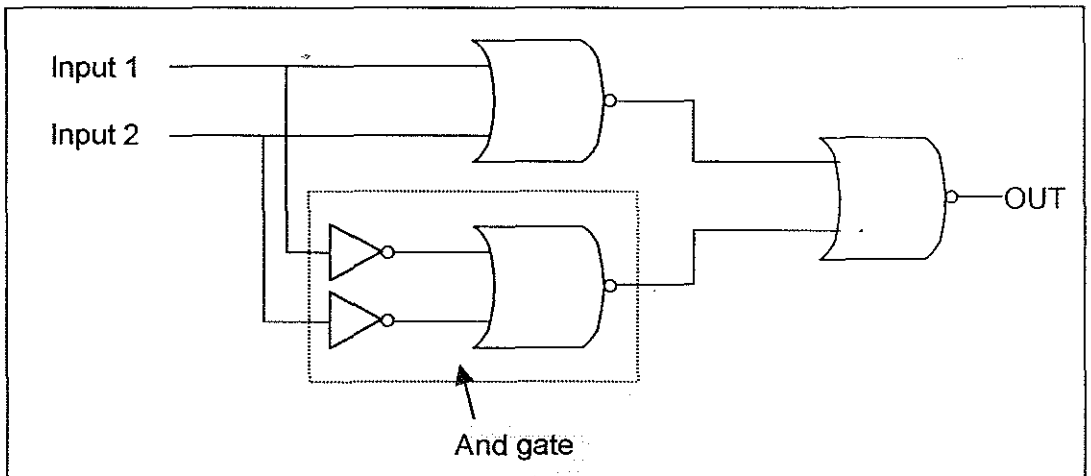


Figure 5.17. Electrical layout of an XOR gate.

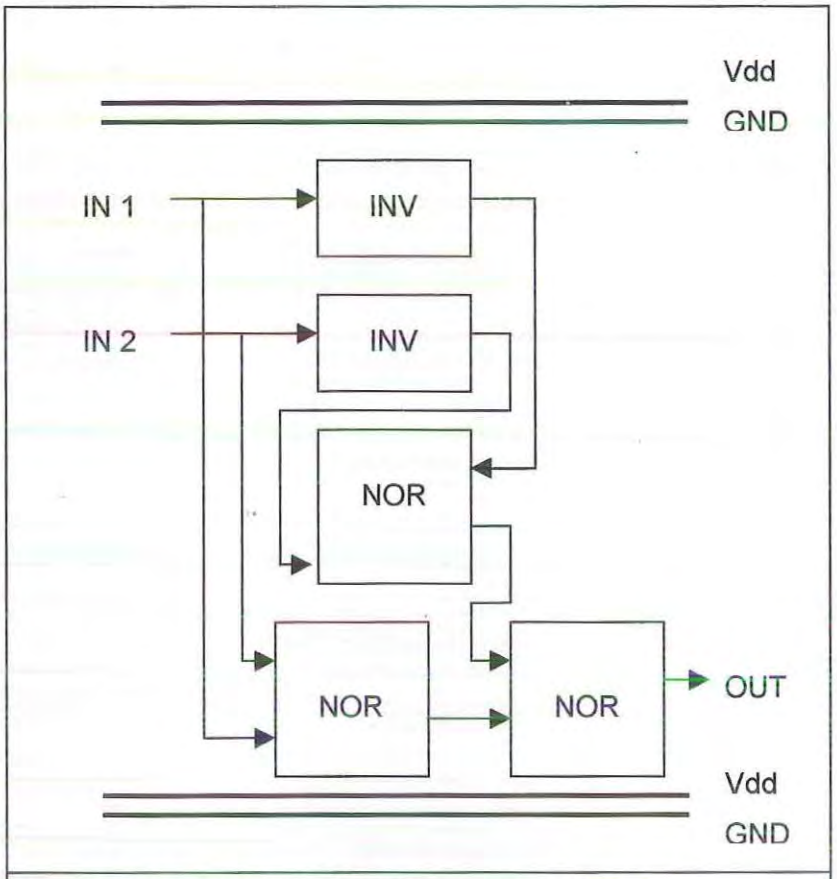
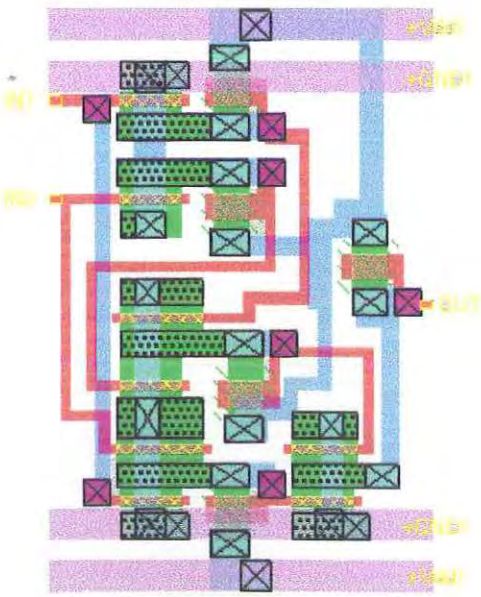


Figure 5.18 Schematic MAGIC Layout of XOR Cell

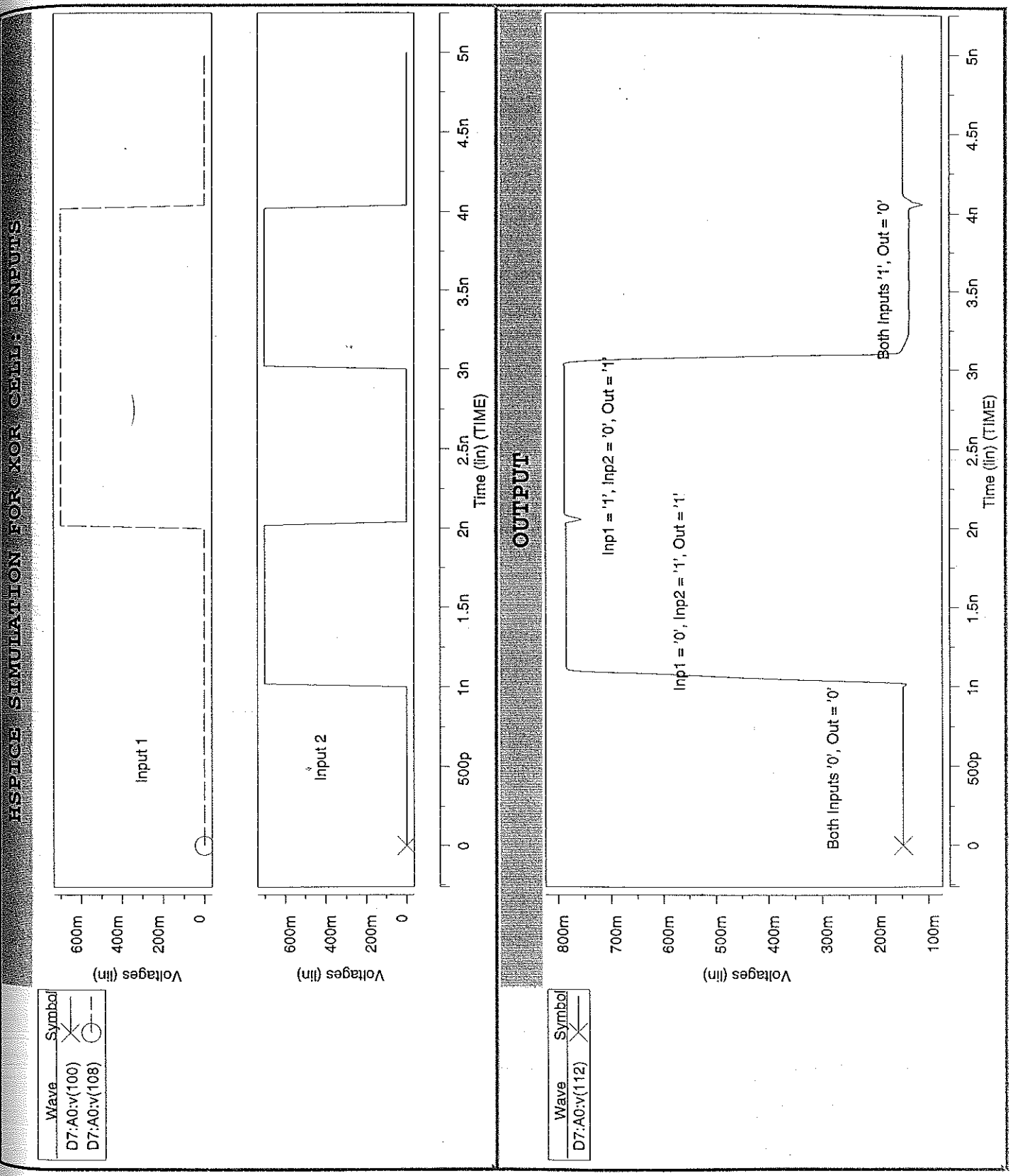


Layout 10: XOR Cell

Hspice Simulation

HSpice simulation for the XOR cell is shown on the next page.

- The output switches between '1' and '0' as per the requirements set out in the behaviour table.
- The response time for the circuit is 0.15ns.


HSPICE SIMULATION FOR XOR CELL: INPUTS
Wave Symbol
D7:A0:v(100)
D7:A0:v(108)
Input 1
Input 2
Voltages (lin)
600m
400m
200m
0
0 500p 1n 1.5n 2n 2.5n 3n 3.5n 4n 4.5n 5n
Time (lin) (TIME)
OUTPUT
Wave Symbol
D7:A0:v(112)
800m
700m
600m
500m
400m
300m
200m
100m
Inp1 = '1', Inp2 = '0', Out = '1'
Inp1 = '0', Inp2 = '1', Out = '1'
Both Inputs '0', Out = '0'
Both Inputs '1', Out = '0'

5.13 The Wait Cell

The wait cell (called Cell 1 in the complete system diagram), as the name suggests, waits for two signal pulses to arrive, and then generates an output pulse. The input pulses arrive at different times, and the cell is to generate an output pulse only after the later pulse has arrived.

Behaviour

The wait cell is to receive (wait for) two input pulses, and in turn output a short pulse when the later of the pulse is received.

5.13.1A D Latch

To understand how the Wait cell performs the required function, it is necessary to understand D-latches and their construction. A layout cell of a D latch has not been designed explicitly; it has been implemented as a part of the Wait cell.

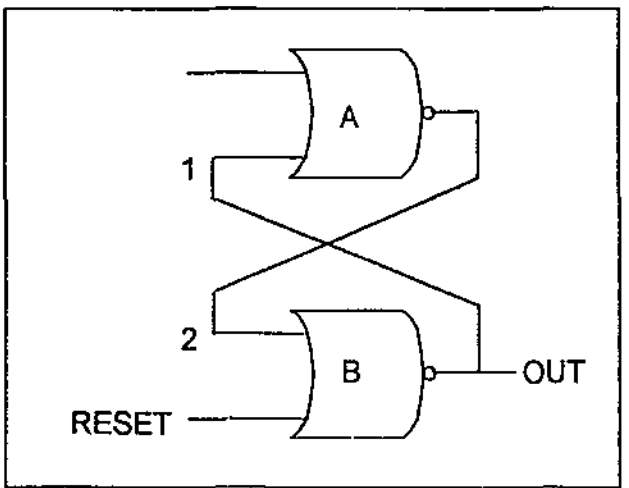


Figure 5.19. Electrical Layout of a D Latch

The figure 5.20, shows the electrical layout of a D-latch. It shows the 2 nor gates and the set and reset signal. By definition, a D-latch outputs a sustained logic '1' signal when the 'Set' pulse is received. Similarly, it outputs a logic '0' when a 'Reset' pulse is received. The following steps describe how this is achieved.

- Assume that initially there is no input being received through the SET or RESET lines. Now if output of nor gate A is a logic '1', then gate B receives logic '1' through input node 2, and its output becomes '0'. This is turn means that a logic '0' is being input back

to A through node 2. This makes both inputs at A '0', thus maintaining its output at '1'. Thus the circuit is in a steady state, with the OUT signal being '0'. Similarly if the output at A were a logic '0', the circuit would be in a steady state with OUT signal being '1'.

- Now if a pulse of '1' is received at gate A through the 'Set' line, its output becomes '0'. Even after the pulse has passed, the circuit maintains a steady state with output from A being '0', as seen before. Thus the OUT signal maintains a logic '1' till further input is received. Thus the latch has been *set.*
- Similarly, if a pulse of '1' is received at the RESET line, the output of B would be '0', and the OUT signal will maintain a logic '0' till further input. Thus the latch has been *reset.*

Design of the Wait Cell

Having understood the function of the D-latch, it is now appropriate to utilise them to achieve the function of the Wait cell.

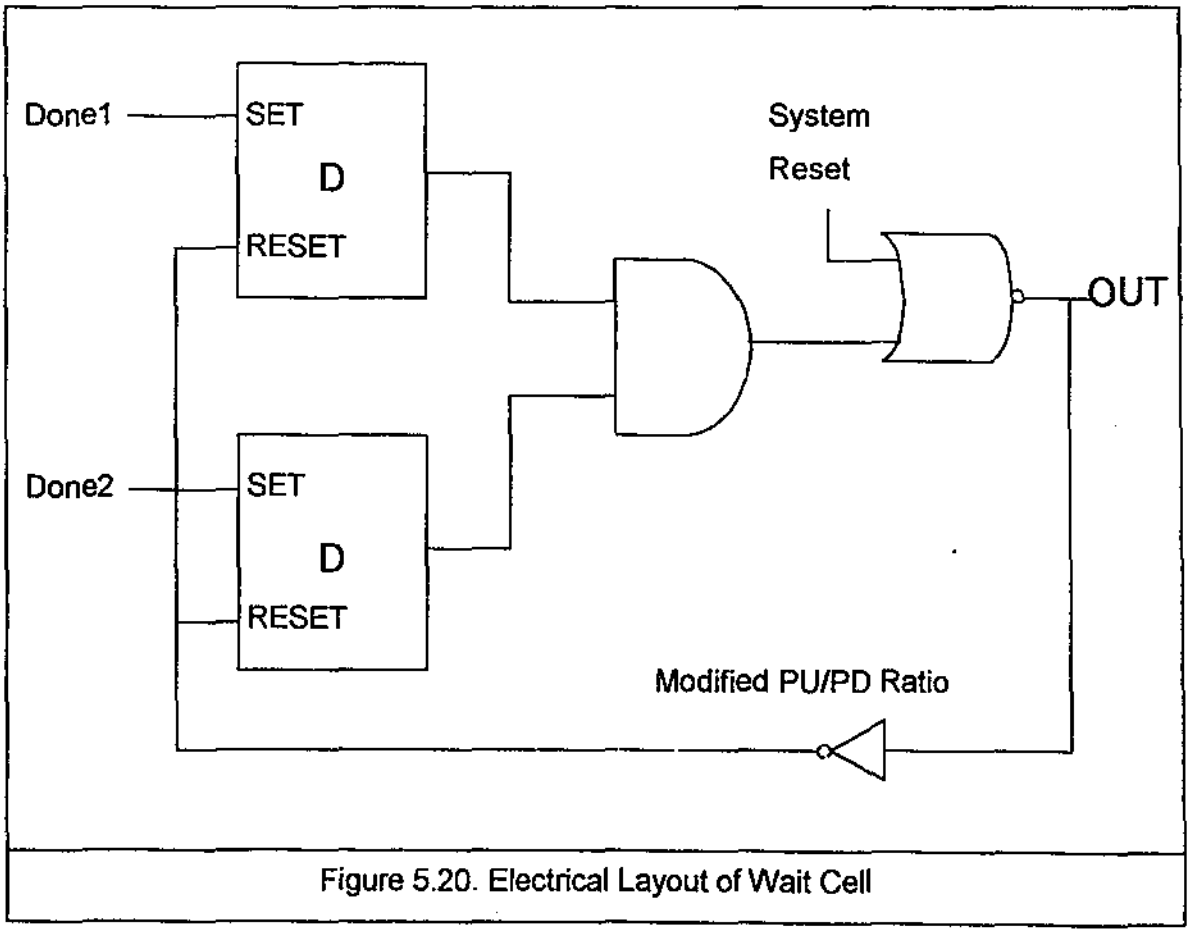
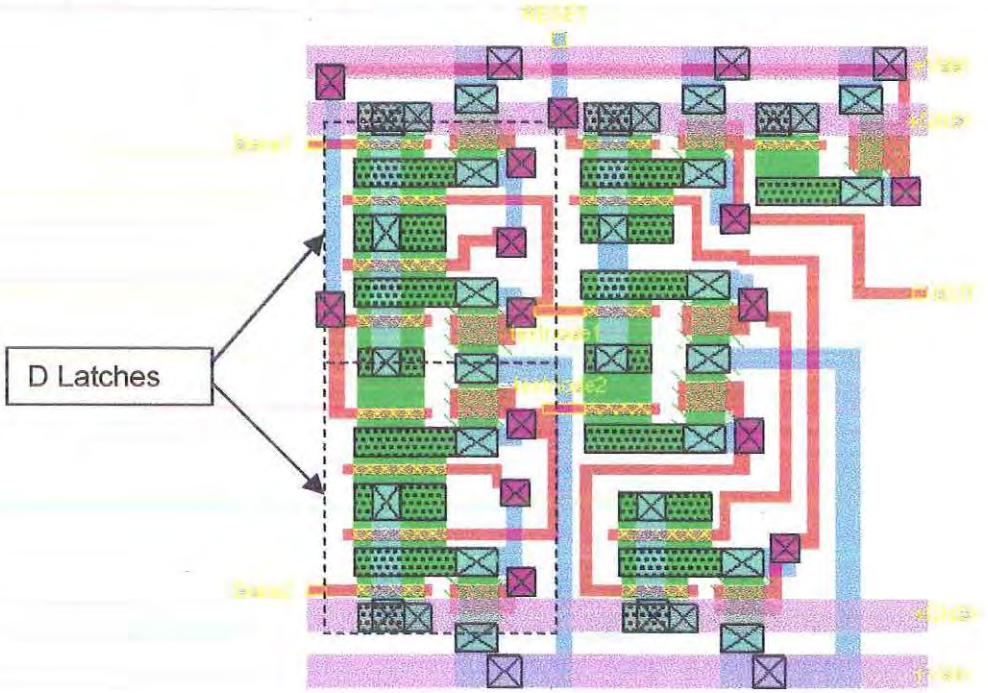


Figure 5.20. Electrical Layout of Wait Cell

Figure 5.21 above shows the electrical layout of the Wait cell. As seen, two D-latches and an And gate are used in the design. The design and implementation of both these cells has been discussed earlier. The operation of the cell has been described in the following cells.

- At start, assume that there is no signal at the Done1 and Done2. A 'reset system' pulse of logic '1' is received at the nor gate, setting its output to logic '0'. Thus the OUT signal pulses from '1' to '0'. This pulse, however, is reversed again by the inverter, and both the RESET lines on the two D-latches receive a pulse. This sets their output to '0'. This is thus the initial state of the circuit.
- In the initial state, both the inputs to the 'And' gate are logic '0'. Thus its output is '0' too. This means both the inputs to the nor gate are logic '1', making its output '1'. Thus, in the initial state (after the reset pulse), the OUT signal is at constant '1' level, and none of the D-latches have been set.
- When the first of the two input signal pulses (Done1 and Done2) arrives, it sets the corresponding D-latch. One of the inputs to the And gate becomes '1'. However, since an 'And' gate requires both its inputs to be logic '1' to output a '1', its output remains unchanged.
- When the second input pulse arrives, both the D-latches are set, and both the inputs for the And gate are in logic '1' state. Thus, there is an output of '1' from the And gate, which in turn acts as input to the nor gate. The output of the nor gate becomes drops to logic '0'. Thus, after arrival of both the input pulses, the OUT signal has changed from logic '1' to logic '0'.
- The logic '0' at OUT signal is again inverted to '1' for the two Reset signals. Thus the D-latches are reset again, and eventually the Output rises back to logic '1'. Thus, the OUT signal pulses to a '0' state for a very short time, depending on the delay in the circuit. To increase the duration of this OUT pulse, more delay should be introduced in the circuit. Thus the PU/PD ratio of the inverter is modified to introduce more delay in the circuit.

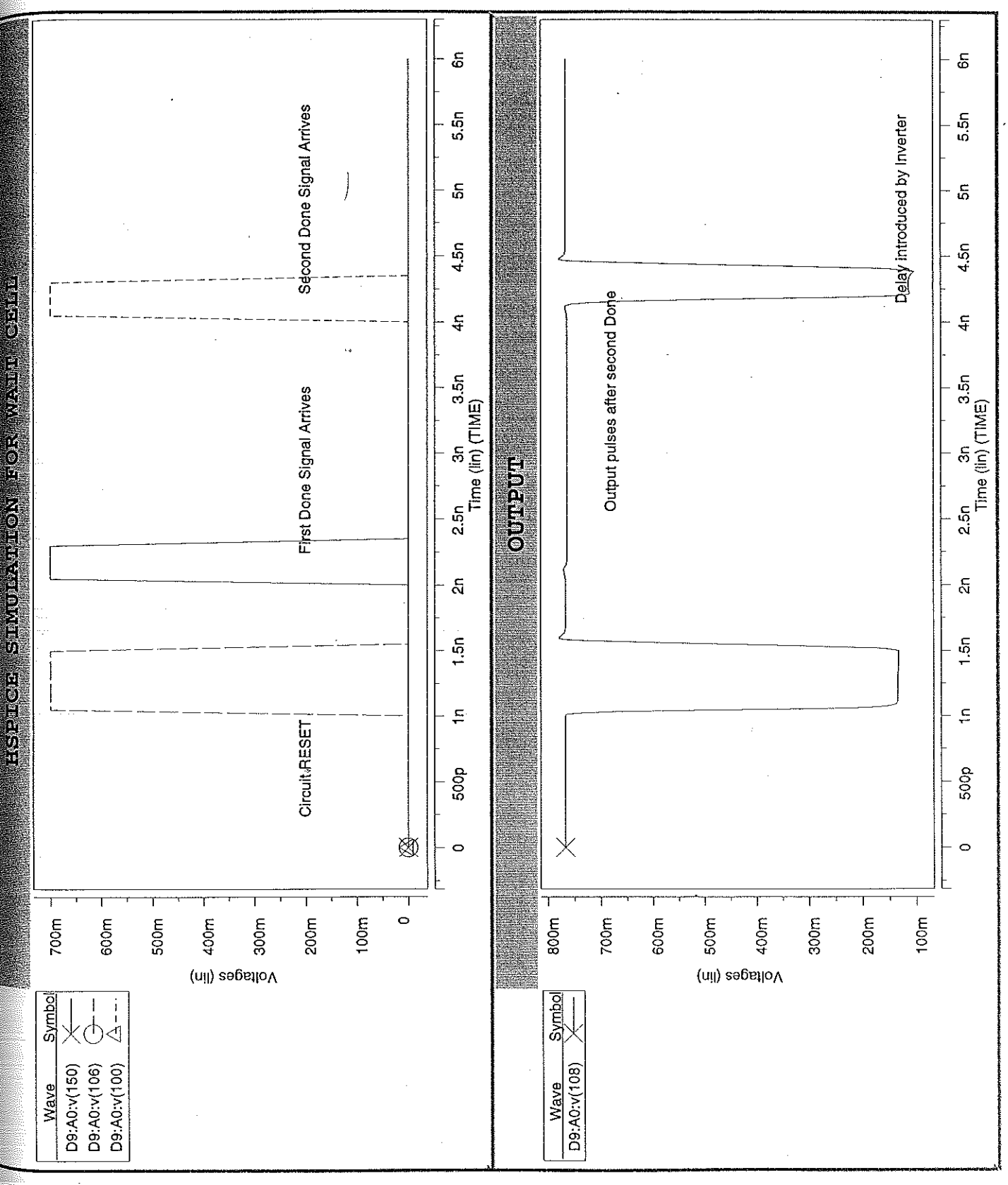


Layout 11: The Wait Cell

Hspice Simulation

HSpice simulation for the Wait cell has been shown on the next page.

- The output goes low initially when the circuit is reset.
- The output then stays high till both the Done signals have arrived, then it pulses from '1' to '0'.
- The pulse width is about 0.3ns. This delay is introduced by the inverter and other elements in the circuits.


HSPICE SIMULATION FOR WAIT CELL
Wave
Symbol
D9:A0:v(150)
D9:A0:v(106)
D9:A0:v(100)
Voltages (lin)
700m
600m
500m
400m
300m
200m
100m
0
Circuit:RESET
First Done Signal Arrives
Second Done Signal Arrives
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
4.5n
5n
5.5n
6n
Time (lin) (TIME)
OUTPUT
Wave
Symbol
D9:A0:v(108)
Voltages (lin)
800m
700m
600m
500m
400m
300m
200m
100m
Output pulses after second Done
Delay introduced by Inverter
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
4.5n
5n
5.5n
6n
Time (lin) (TIME)

5.14 Data Buffer

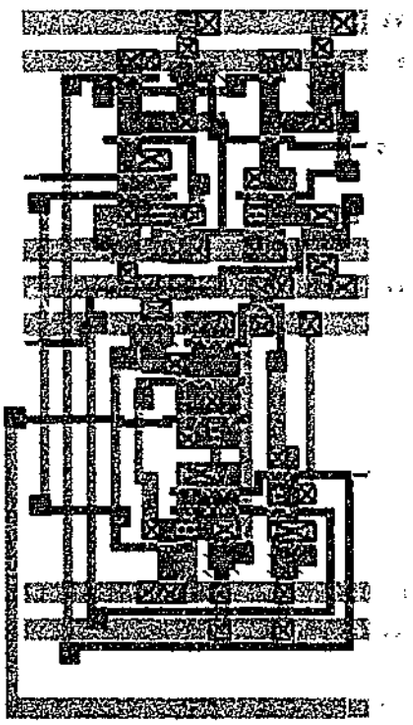
The data buffer (called Cell3 in over system diagram) performs the function of holding one bit of information. It is a self-timed cell, and passes the data along from input to output upon receiving a 'request' signal. This cell is very similar to a unit cell in the bubble-shift register, and thus is implemented by modifying the register unit cell.

Behaviour

As mentioned, the data cell is required to hold one bit of data and transfer it upon receiving a request signal.

Design

The data buffer design is very similar to the unit register cell. It is implemented simply by removing the handshake block (and its corresponding inputs and outputs) from the register cell. The electrical layout is similar to the register cell. Cell layout is as shown below.



Layout 12: The Data Buffer

Since the behaviour of a Data cell is similar to the Register cell, HSpice analysis for the Data cell is not shown.

5.15 The Subtract Selector

The subtract selector (called Cell 2 in the overall system diagram) enables us to set the first bit of each register value to '1', in the event of the operation being a subtraction. This is required to obtain a 2's complement of any binary number, so that subtraction can be performed in the adder block. The cell design was based on the basic LCFL cell structure.

Behaviour

The subtract selector should output a '1' upon receiving either an input of logic '1' or alternatively if the subtract flag is set. Thus it should perform an 'OR' function. Since it is a self-timed circuit, a request signal should be received at an appropriate time. For the first bit of the addition/subtraction it should receive a request signal as soon as the system starts operation. For the remaining bits, it should receive a request signal that coincides with the 'done' signal from the previous block, ie the Adder.

Design

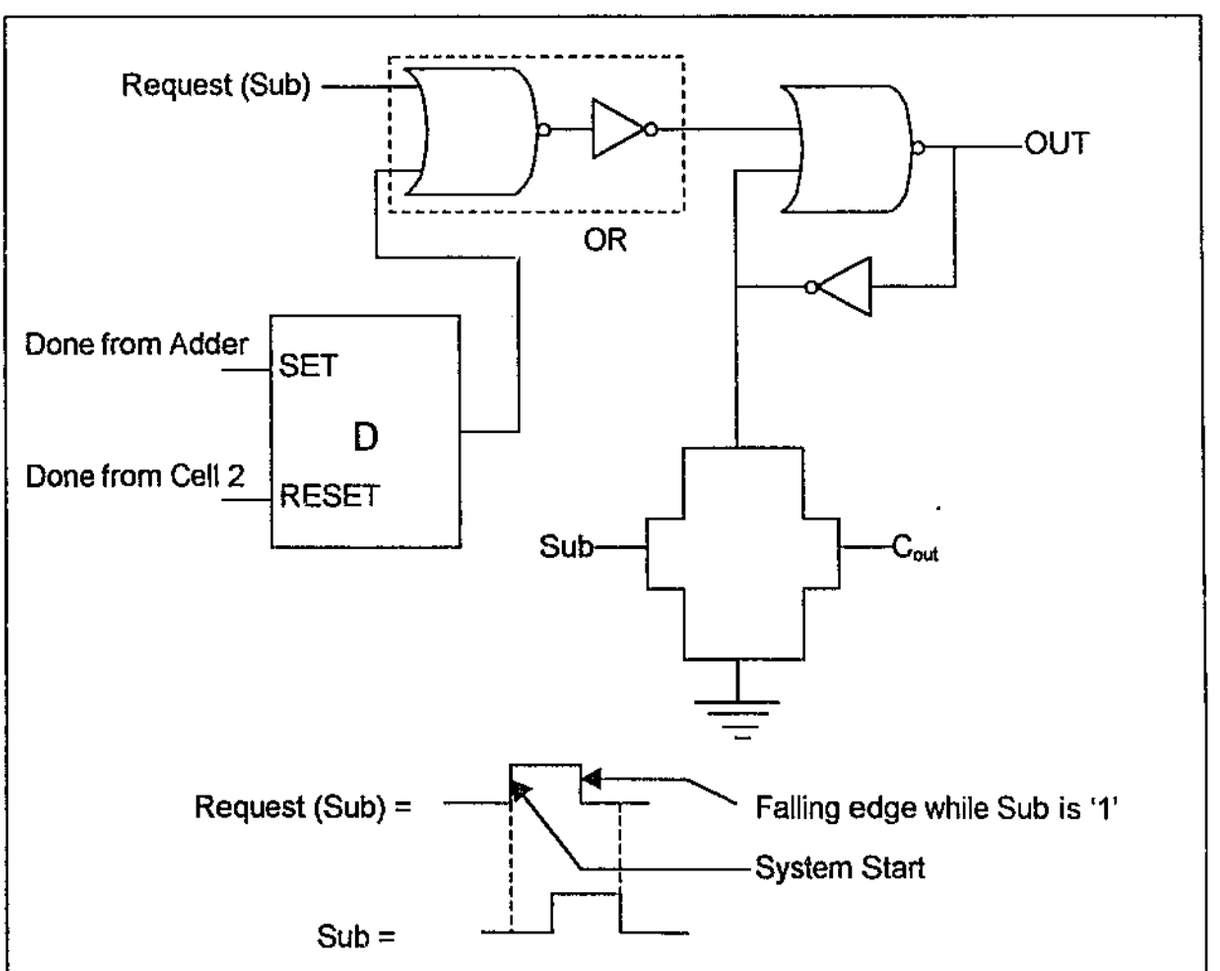
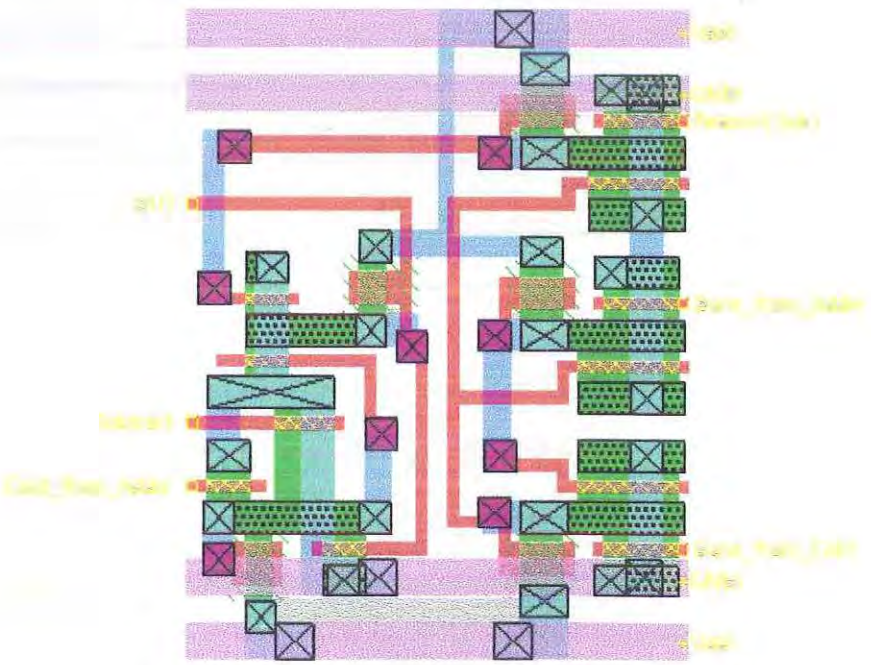


Figure 5.21. Electrical Layout of the Subtract Selector Cell

Figure 5.21 above shows the electrical layout of the cell. As seen, a D latch has been utilised to configure the Request signal appropriately. The cell has been designed based on the LCFL structure. Unlike the other LCFL cells, however, the OUT' and the Complete signals are not generated, as they are not necessary. The corresponding gates have thus been excluded from the design.

- The functioning of the circuit is similar to the LCFL basic cell, as described before. The OR function has been implemented by two pass transistors conducting to ground parallel to each other. The two inputs to these pass transistors are the Subtract flag and the carry bit from the previous operation. Thus, if the Subtract flag is set, the output is set to 1 regardless of the input.
- The 'request' signal is formed by two different inputs going into an OR gate, meaning any one of them can trigger the system. As seen, the 'Done' signal from the Adder sets the D-latch and thus triggers the request signal. The circuit remains in the evaluate stage till the 'Done' signal from the Data Buffer cell arrives. Since the output of this cell is no longer required, the request signal turns off.



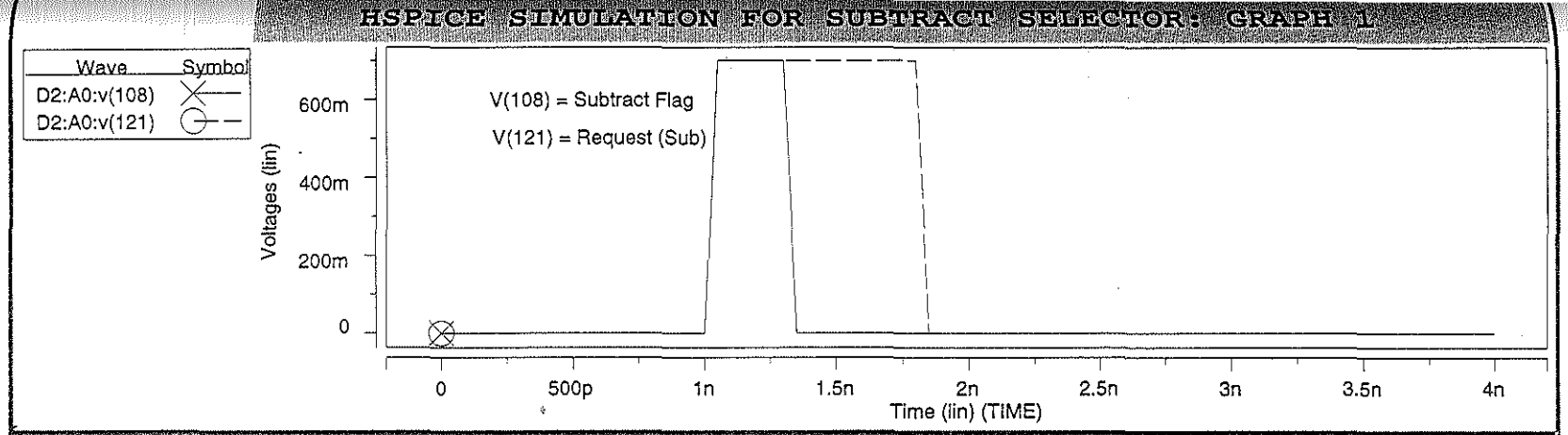
Layout 13: Subtract Selector Cell

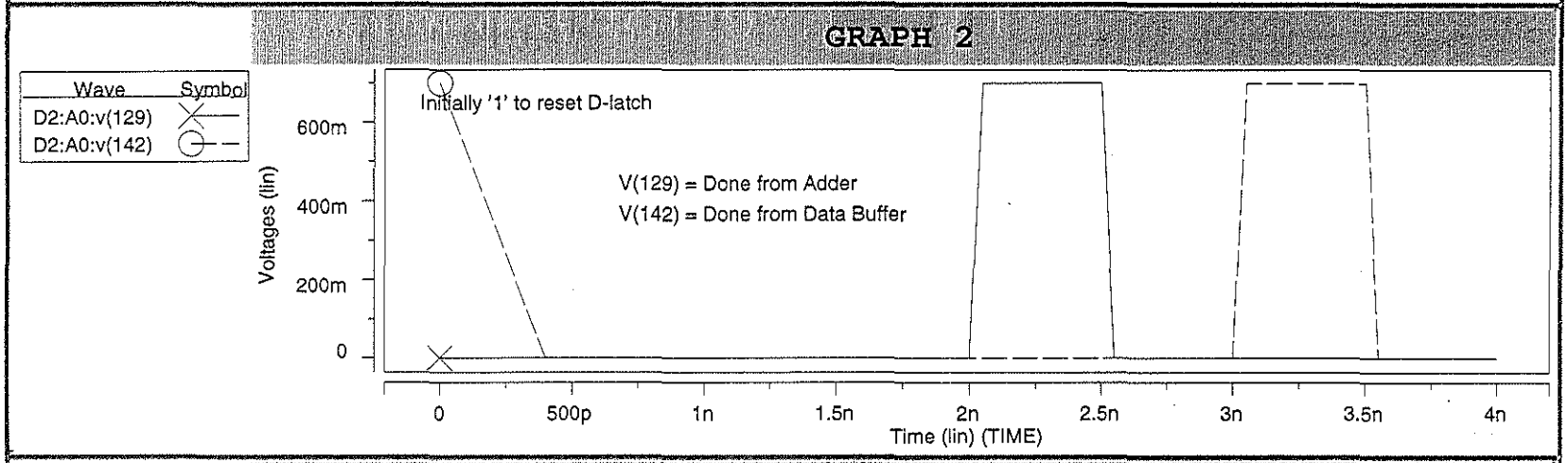
- When the system is in its first cycle of operation, however, the cell does not receive any 'Done' signal from the adder. It should be able to generate an appropriate C_{in} for the first bit addition. For this purpose, the Request(sub) signal triggers the cell. The start of this signal coincides with the system Start signal.
- The subtract flag should be programmed so that it switches to logic '1' during the first bit of a new pixel value where a subtraction is required. This can be achieved using a Finite State Machine (FSM). This, however, is not encompassed in the scope of this project.

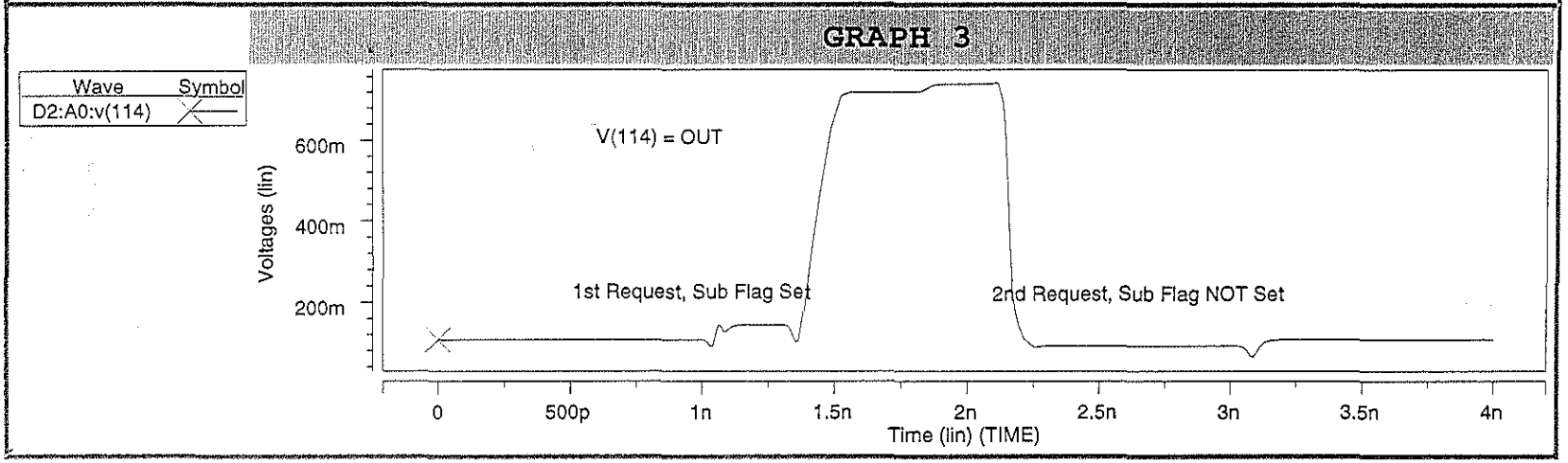
HSpice Simulation

The HSpice simulation for the Subtract Selector is shown on the next page.

- Graph 1 shows the Request (Sub) signal. It is pulsed from '0' to '1'. The subtractor flag has been set meanwhile. Thus the output shifts from '0' to '1' around 1.5ns.
- The Graph 2 shows the two Done signals from Adder and Data buffer. Initially, the Done from data buffer is high to ensure that the circuit is reset. When a pulse of Done arrives from the Adder, the request for the circuit goes to high. It remains high till the Done from data buffer arrives. The Output switches off, as the subtractor flag and input are switched off (graph 1).
- Response time for the circuit is 0.35ns


HSPICE SIMULATION FOR SUBTRACT SELECTOR: GRAPH 1
Wave
Symbol
D2:A0:v(108)
D2:A0:v(121)
V(108) = Subtract Flag
V(121) = Request (Sub)
Voltages (lin)
600m
400m
200m
0
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
Time (lin) (TIME)


GRAPH 2
Wave
Symbol
D2:A0:v(129)
D2:A0:v(142)
Initially '1' to reset D-latch
V(129) = Done from Adder
V(142) = Done from Data Buffer
Voltages (lin)
600m
400m
200m
0
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
Time (lin) (TIME)


GRAPH 3
Wave
Symbol
D2:A0:v(114)
V(114) = OUT
1st Request, Sub Flag Set
2nd Request, Sub Flag NOT Set
Voltages (lin)
600m
400m
200m
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
Time (lin) (TIME)

5.16 The Sum Cell

The sum cell is a part of the Adder, and it provides a sum of the two input values and carry input. It is designed based on the LCFL cell architecture.

Behaviour

The sum cell is required to output a sum of three binary numbers, two inputs from registers and the carry from previous summation. The function is described by truth table below.

Input 1	Input 2	Carry In	Sum Out
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 5.9. Truth Table for Sum Operation

Let Input 1 = A, Input 2 = B and Carry in = C.

Then Sum = A' B' C + A' B C' + A B' C' + A B C

The Karnaugh map for the above boolean equation is shown below

CVAB	OO	O1	11	1O
O		1		1
1	1		1	

Table 5.10. Karnaugh map for Sum function

As seen from the Karnaugh map, there is no simplification possible for the logic equation stated above. This must be implemented in our circuit.

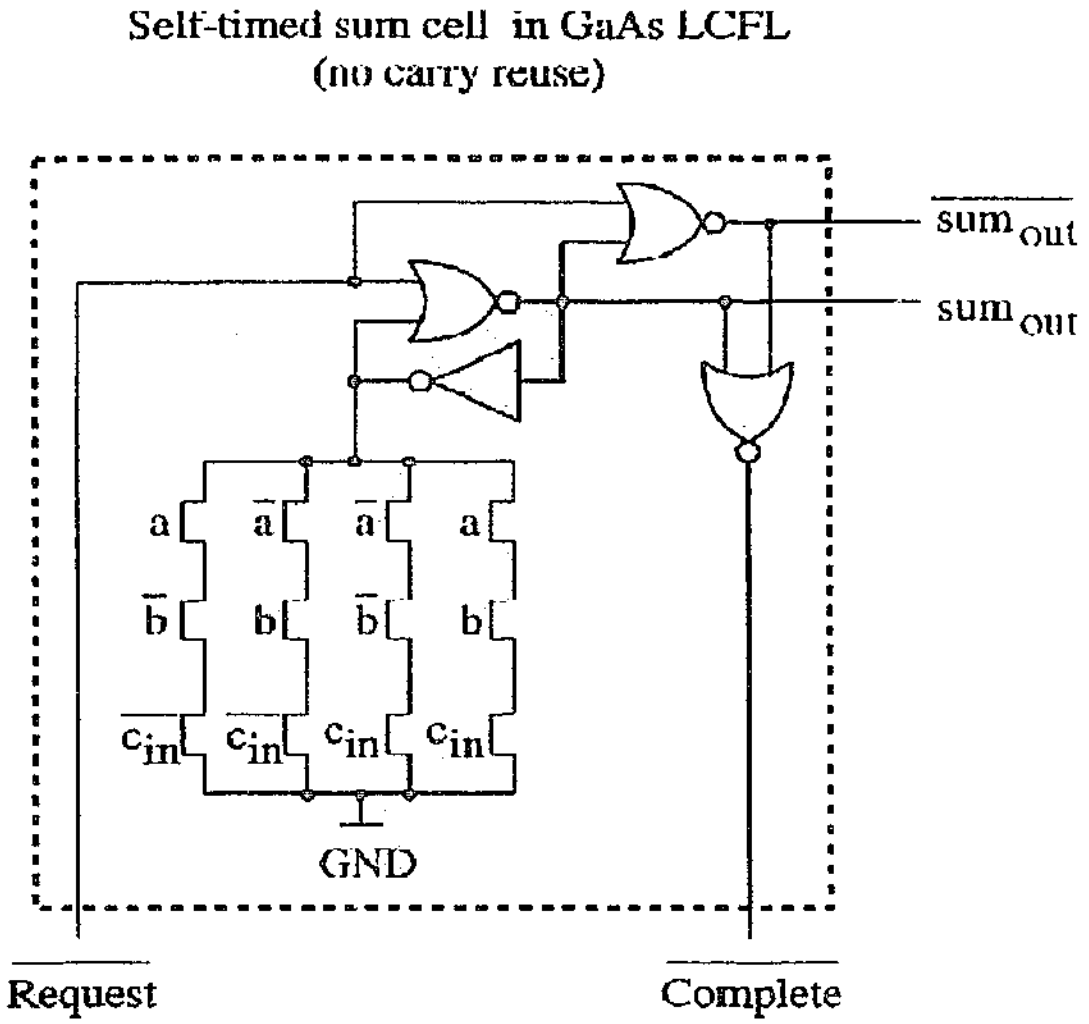
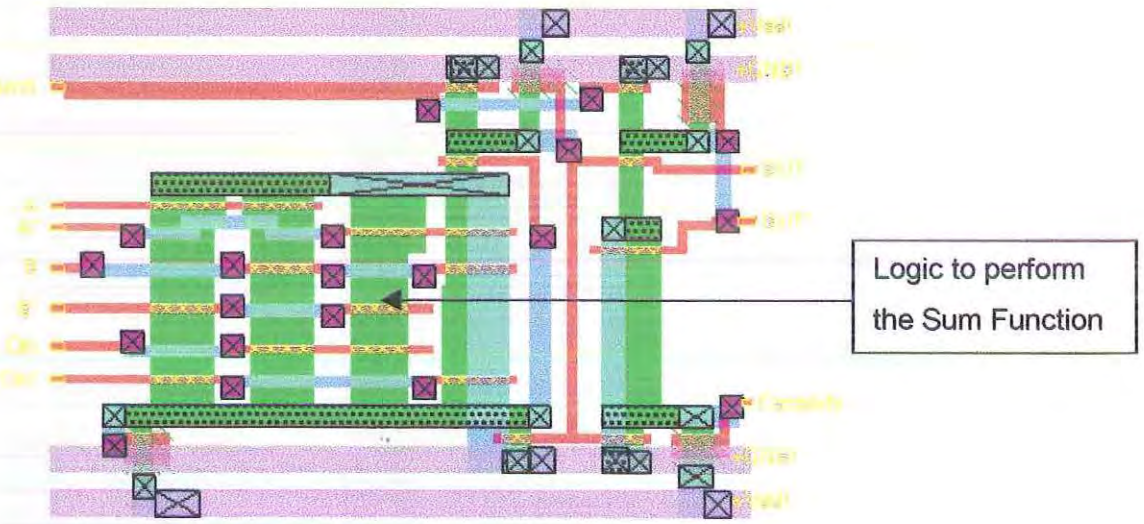


Figure 5.22: Electrical Layout of LCFL Sum Cell.

Design

Figure 5.22 above shows the electrical layout of the Sum cell. As seen it is based on the LCFL cell architecture. The logic is implemented by parallel branches of pass transistors conducting to ground. These transistors are then controlled by appropriate inputs to implement the function above.The figure shows that there are upto 4 pass transistors in series in each of the branches conducting to ground. For the logic function to be implemented correctly, the latch should be grounded to logic '0' through the conducting branches, if any. This is not possible using a normal PU/PD ratio as pull-down is not strong enough to remain lower than threshold voltage through 4 transistors. Thus, the pull-down has to be made stronger. So the ratio for pull-down transistors has been changed to 10:1 as seen in the layout below. This makes the circuit 'area-expensive'.

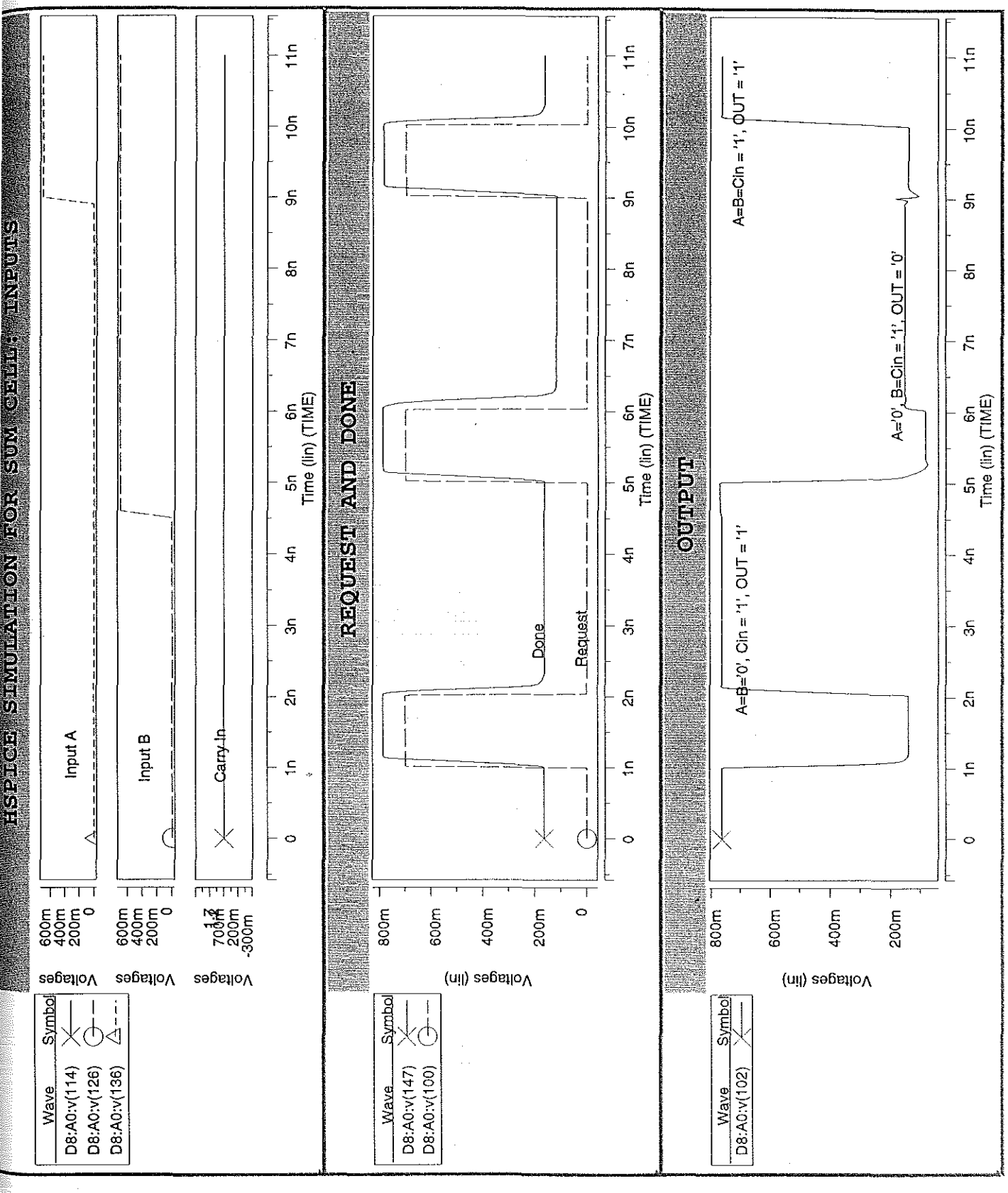


Layout 15: Self-timed Sum Cell

Hspice Simulations

HSpice simulation for the Sum cell has been shown on the next page.

- The inputs are switched on at different times to simulate three different conditions (0+0+1), (0+1+1) and (1+1+1).
- The Request signal triggers the cell for the three conditions.
- The Output changes sets to 1, then 0, then 1 for the three conditions. This is as per the requirements set out in the behaviour table.
- Response time of the circuit is 0.2ns.


HSPICE SIMULATION FOR SUM CELL: INPUTS
Input A
Input B
Carry-In
Voltages
600m
400m
200m
0
700m
200m
-300m
0
1n
2n
3n
4n
5n
6n
7n
8n
9n
10n
11n
Time (lin) (TIME)
Wave
Symbol
D8:A0:v(114)
D8:A0:v(126)
D8:A0:v(136)
REQUEST AND DONE
Done
Request
800m
600m
400m
200m
0
Voltages (lin)
D8:A0:v(147)
D8:A0:v(100)
OUTPUT
A=B=Cin = '1', OUT = '1'
A=B='0', Cin = '1', OUT = '1'
A='0', B=Cin = '1', OUT = '0'
D8:A0:v(102)

5.17 The Carry Cell

The carry cell is required to compute the carry generated in the Adder through its operation. It receives two input values and carry from previous operation, and computes the carry accordingly. This cell is also designed along the lines of LCFL architecture.

Behaviour

The carry cell should take in three inputs, two from the registers and the carry generated in previous operation. It should then be able to output the appropriate carry out signal. The function can be described by the truth table below.

Input 1	Input 2	Carry In	Carry Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 5.11. Truth Table for Carry Operation

Let Input 1 = A, Input B, Carry In = C
Then the logic equation for Carry becomes

Carry Out = A' B C + A B' C + A B C' + ABC

The karnaugh map for the above logic equation is shown below.

	C\AB	OO	O1	11	1O
C	O			1	
	1		1	1	1

Table 5.12. Karnaugh map for Carry function

The function thus simplifies to
Carry Out = AB + BC + AC

Design

The figure 5.23 above shows the electrical layout of the carry cell. It is derived from the logic equation for the function. The cell layout is shown below

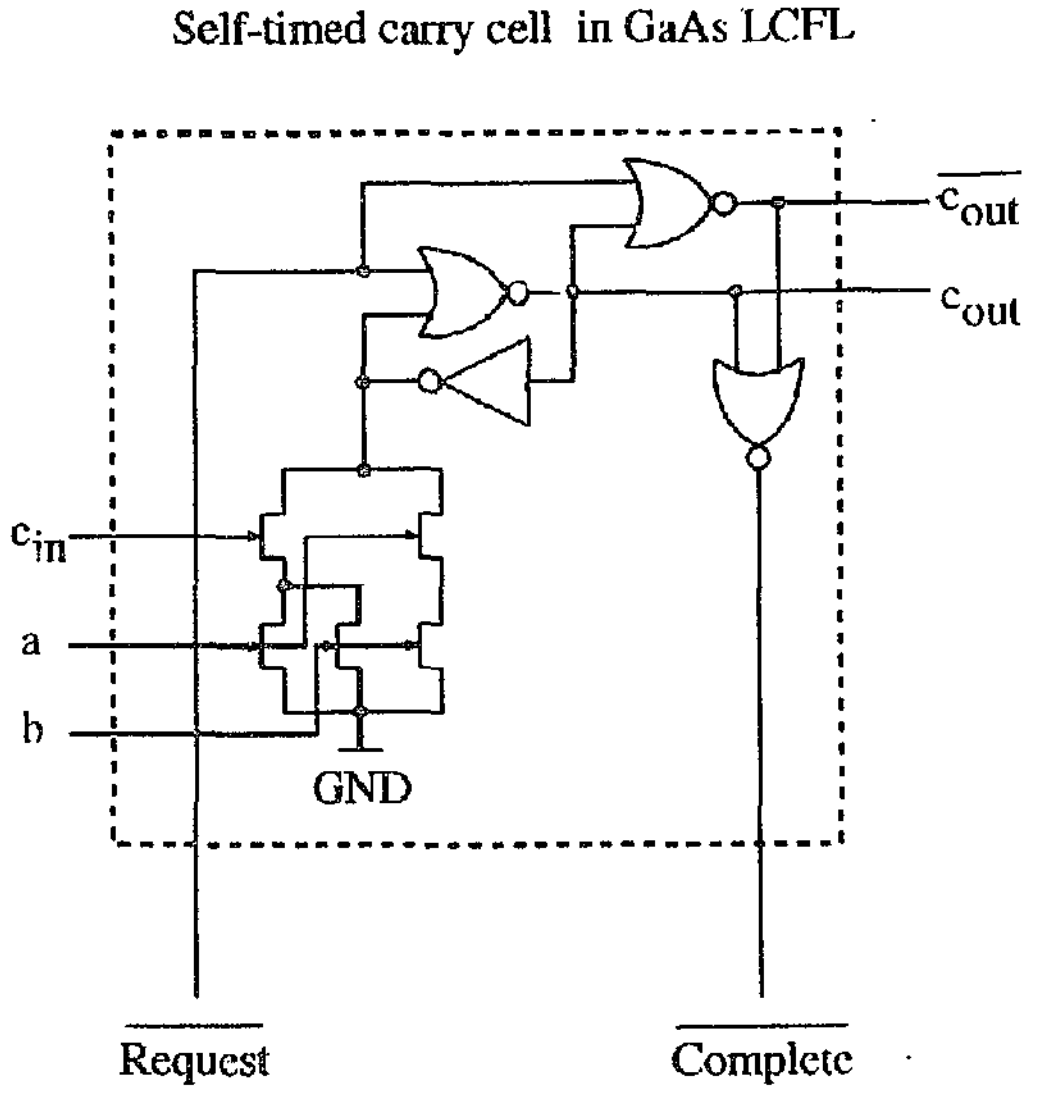
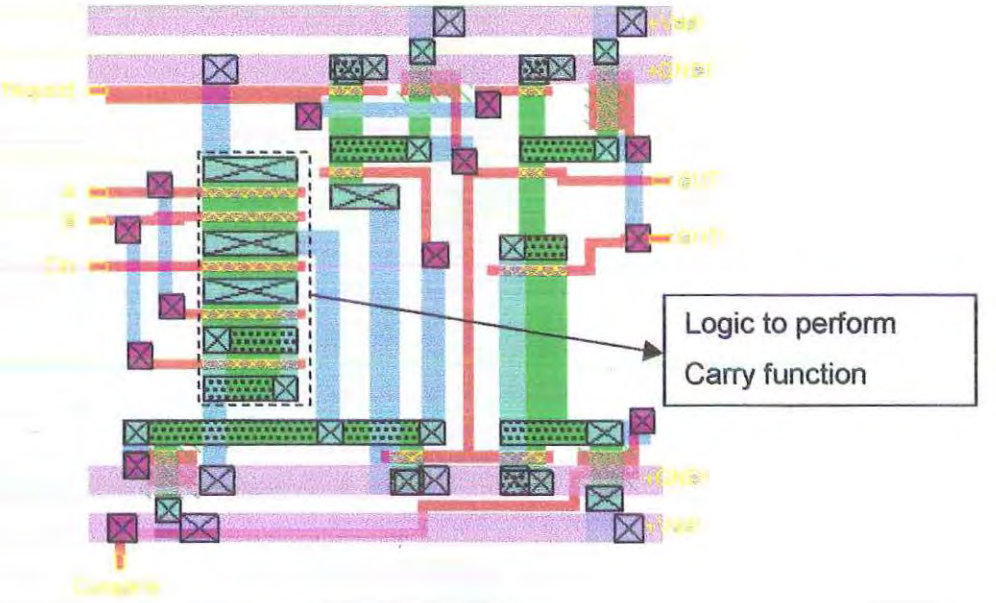


Figure 5.23: Electrical Layout of LCFL Carry Cell

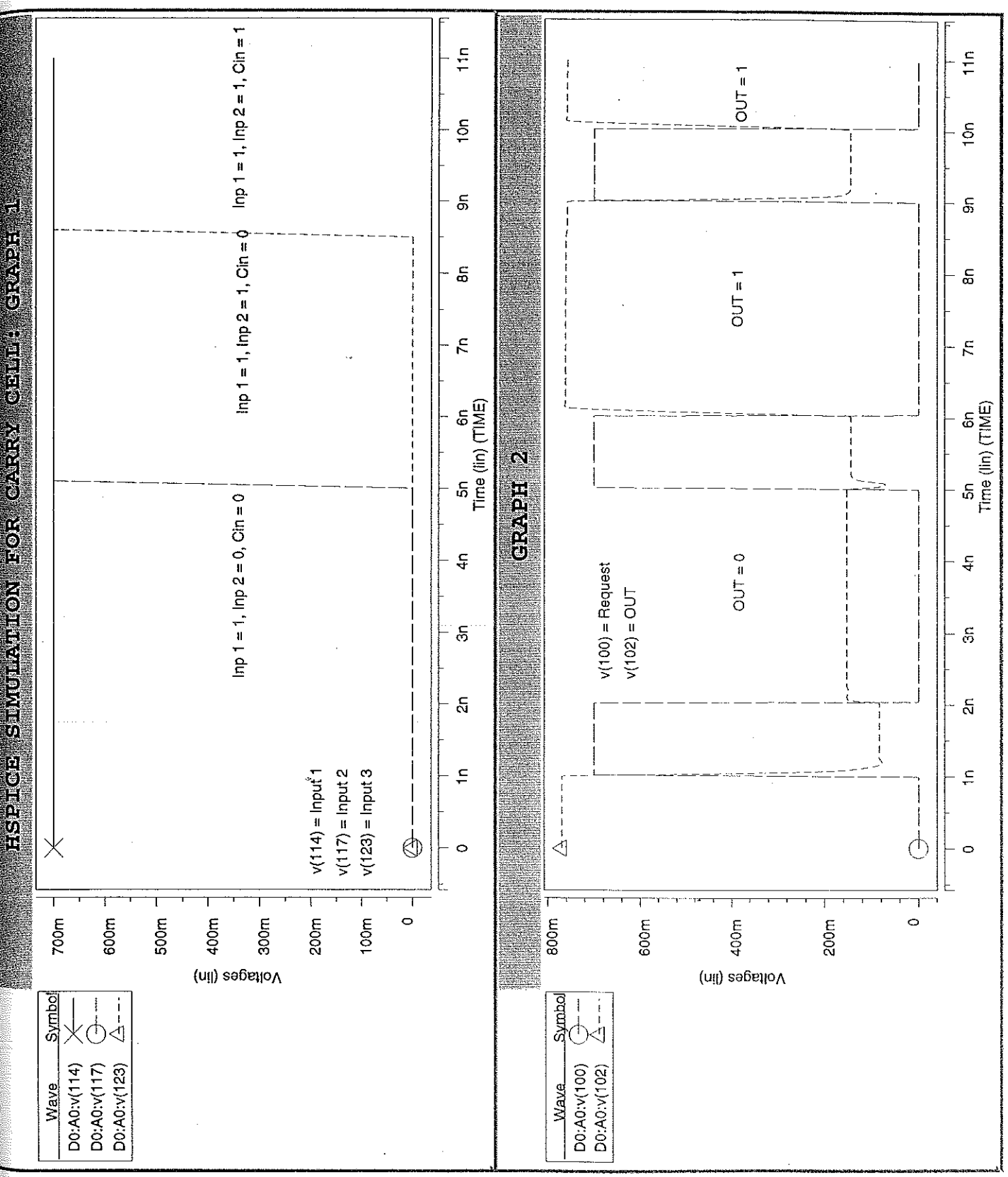


Layout 16: Self-timed Carry Cell

HSpice Simulation

The HSpice simulation for Carry cell has been shown on the next page.

- Graph 1 shows that the three inputs are switched on at different times to simulate three different conditions: (0+0+1), (0+1+1), (1+1+1).
- The Output changed upon receiving the Request signal. The Output was off, then on, then on. This agrees with the requirements set out in the behaviour table.
- The response time for the circuit was 0.15ns


HSPICE SIMULATION FOR CARRY CELL : GRAPH 1
Wave
Symbol
D0:A0:v(114)
D0:A0:v(117)
D0:A0:v(123)
Voltages (lin)
700m
600m
500m
400m
300m
200m
100m
0
v(114) = Input 1
v(117) = Input 2
v(123) = Input 3
Inp 1 = 1, Inp 2 = 0, Cin = 0
Inp 1 = 1, Inp 2 = 1, Cin = 0
Inp 1 = 1, Inp 2 = 1, Cin = 1
0
1n
2n
3n
4n
5n
6n
7n
8n
9n
10n
11n
Time (lin) (TIME)
GRAPH 2
Wave
Symbol
D0:A0:v(100)
D0:A0:v(102)
Voltages (lin)
800m
600m
400m
200m
0
v(100) = Request
v(102) = OUT
OUT = 0
OUT = 1
OUT = 1
0
1n
2n
3n
4n
5n
6n
7n
8n
9n
10n
11n
Time (lin) (TIME)

5.18 Complete Filter Cell Implementation

The logical architecture of the overall filter cell has been shown earlier. Based on this architecture, various cells were designed and implemented. Thus, to construct the filter cell, it was necessary to provide interconnections between various blocks designed.

Note that all the different cells implemented were laid out so that their height was equal to one of the two standard heights used in the system. This allows us to abutt the cells together, sharing the power, ground and reset rails. As shown in the figure, the register cells, 4x1 multiplexer and data buffer were all designed be of the same height. Similarly, the adder, carry, exclusive or, subtract selector and wait cell were designed to be of the same height. Thus it was possible to abutt them side-by-side in the final layout.

The schematic layout shows the approximate sizes and interconnections for various blocks in the system. These sizes are to scale and thus the schematic can be translated directly to actual arrangement of cells in the filter cell. These blocks were interconnected together in MAGIC layout as per the schematic. However, from the point of view of analysis, HSpice simulation could not be performed. This is because the convergence equations for the HSpice simulation engine were too long, and the simulation time for each new test was prohibitively large (a few hours for each run). Thus only 3 or 4 of such blocks were connected at a time and tested.

From the schematic it is observed that the approximate dimensions of the filter cell would be under 800λ x 1000λ. Since λ for H-GaAs III technology is 0.4 microns, this would make the filter cell dimensions 0.32mm x 0.4mm, or 0.13 mm^2. Besides the filter cell, other circuitry needed in an Intelligent Pixel (eg the A/D Convertor) shall also occupy space. Thus the cell area of 0.13mm^2 would impose a limit on the minimum size of the pixel. Using technologies like H-GaAs IV and sub-micron CMOS can improve this area and enable us to implement the filter cell more efficiently. The power dissipation through the system is expected to be about 20mW.

As seen, the control circuitry for a cell becomes increasingly complex as the functionality and size increases. Further additions to this cell would be integration of finite state machines to calculate appropriate system starts, bypass circuitry and other interconnection details between two pixels. This would also require a considerable amount of control signals.

Note that apart from the cells designed for the system, there would be an inverter required at the output of the Wait cell. This is because the output pulse of the Wait cell is of opposite logic level as the required input pulse for the next cell. Besides, inverters shall also be required before sum cells performs its function. This is because the sum cell also requires complement of inputs for its function.

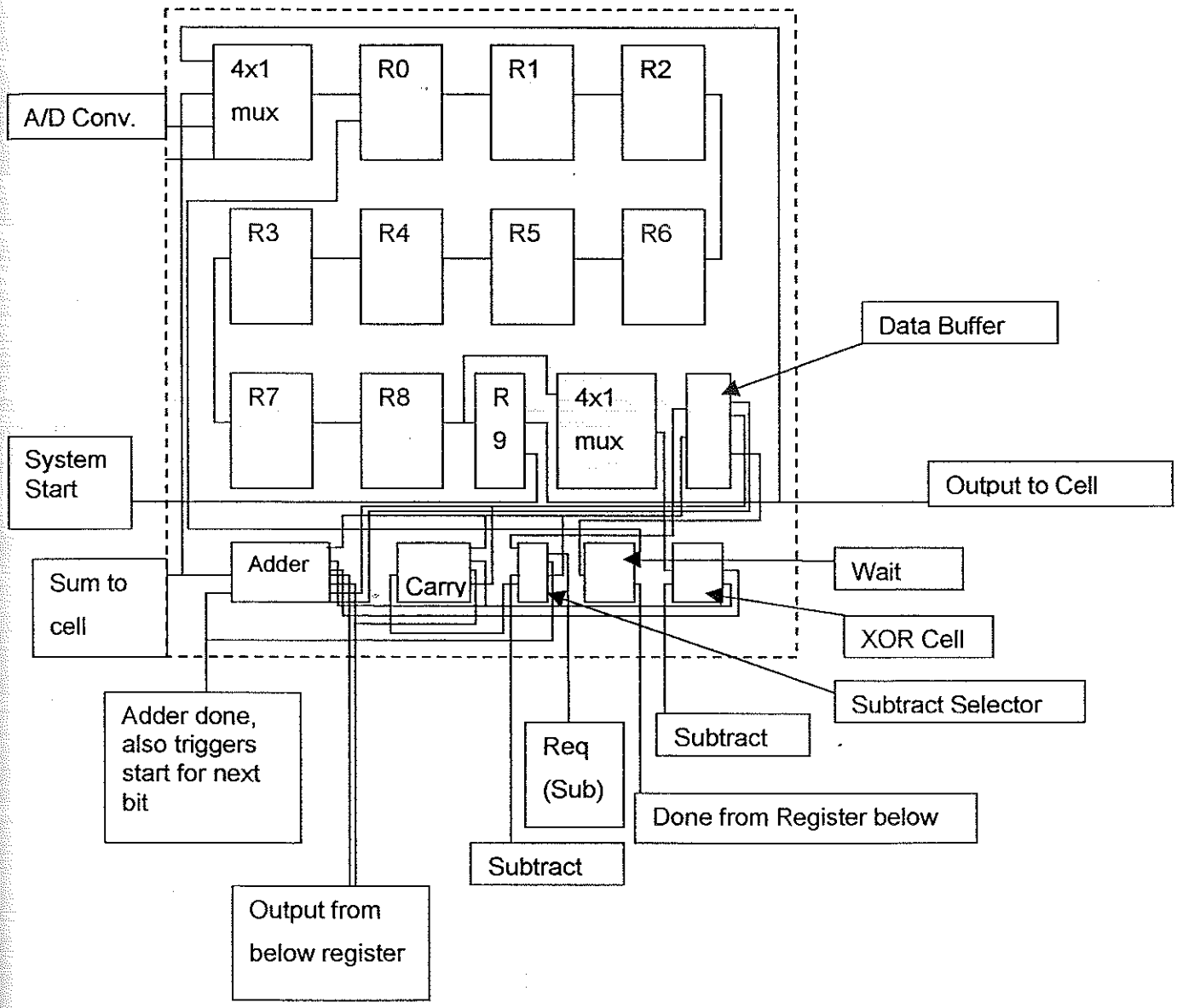


Figure 5.24: Schematic Layout of the Filter Cell. Power, Ground and Reset not shown.

5.19 Evaluation of Simulations

The HSpice simulations were shown after the design description of each cell. All the cells have performed as expected. The design and testing, however, wasn't a straightforward process in all the cases. Some cells required several design corrections. The most notable was the Register cell. Initially, attempts were made to test it individually as a single cell. But later, it was seen that a register cell can be tested better as a part of the whole register.

Circuits like Muller C, Handshake block etc were initially designed without a Reset signal. However, these were later modified to include the reset signal.

The power dissipation was calculated for all the circuits, and has been shown in table 5.13. Power dissipation is calculated by finding the mean current drawn from the Vdd rail during circuit operation and multiplying it by the Vdd voltage, which was 2 V in our case. In MESFET circuits, the current drawn in the circuits is normally quit constant, making it easier to find the mean current. The graphs showing current through Vdd for various circuits are shown at the end of this section.

Cell	Mean Current Through Vdd (mA)	Power Dissipation (mW)
Carry Cell	0.38	0.76
Subtract Selector	0.86	1.72
Data Buffer	0.71	1.42
4 Input Mux-I	3.45	6.9
4 Input Mux-II	1.95	3.9
Register	9.35	18.7
Sum Cell	0.37	0.74
Wait Cell	1.46	2.92
And Cell	0.58	1.16
2-Input Mux	1.15	2.3
Exclusive OR	0.85	1.7
Handshake Block	0.25	0.5
LCFL Basic Cell	0.39	0.78

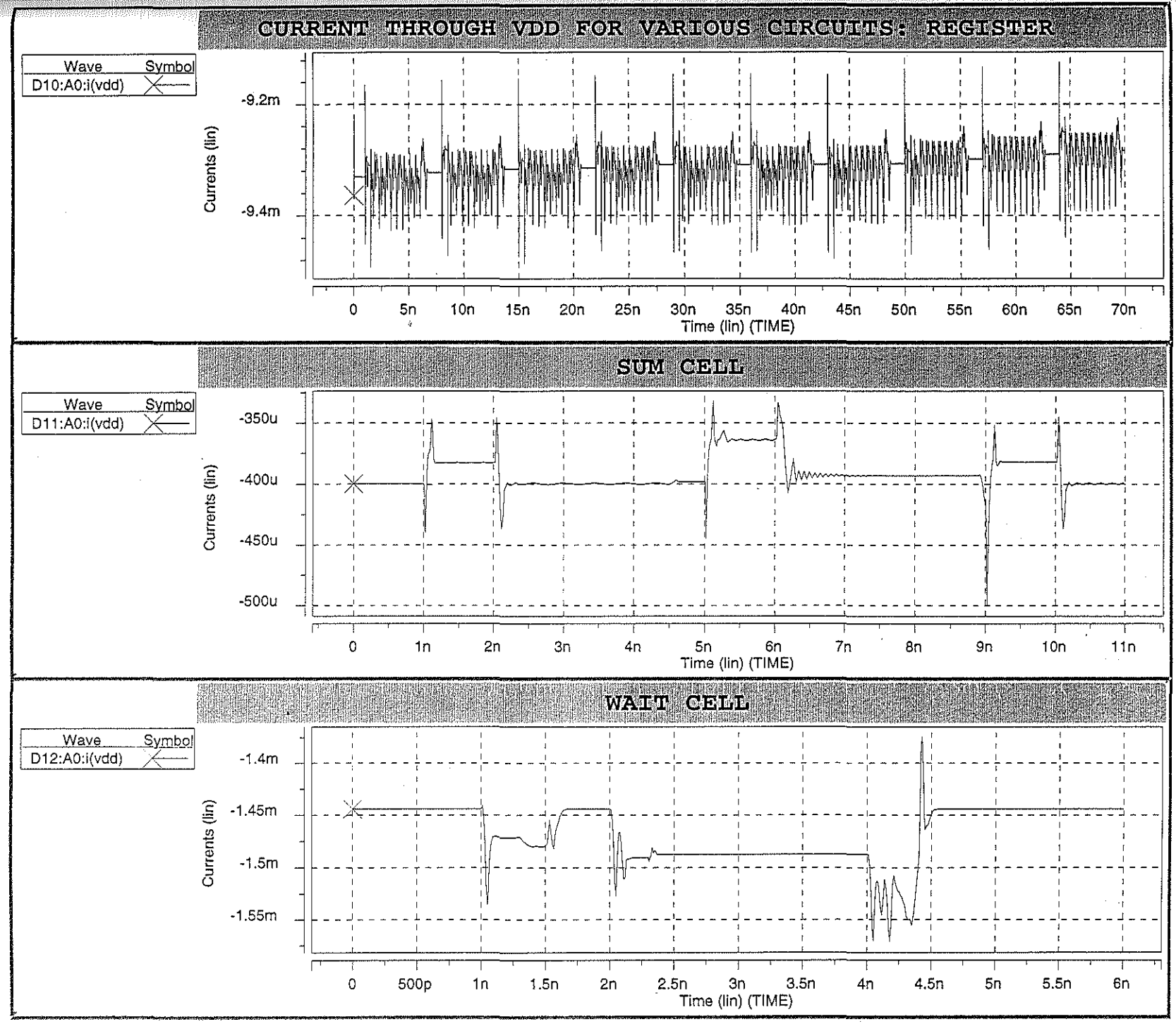
Table 5.13. Power Dissipation for cells implemented

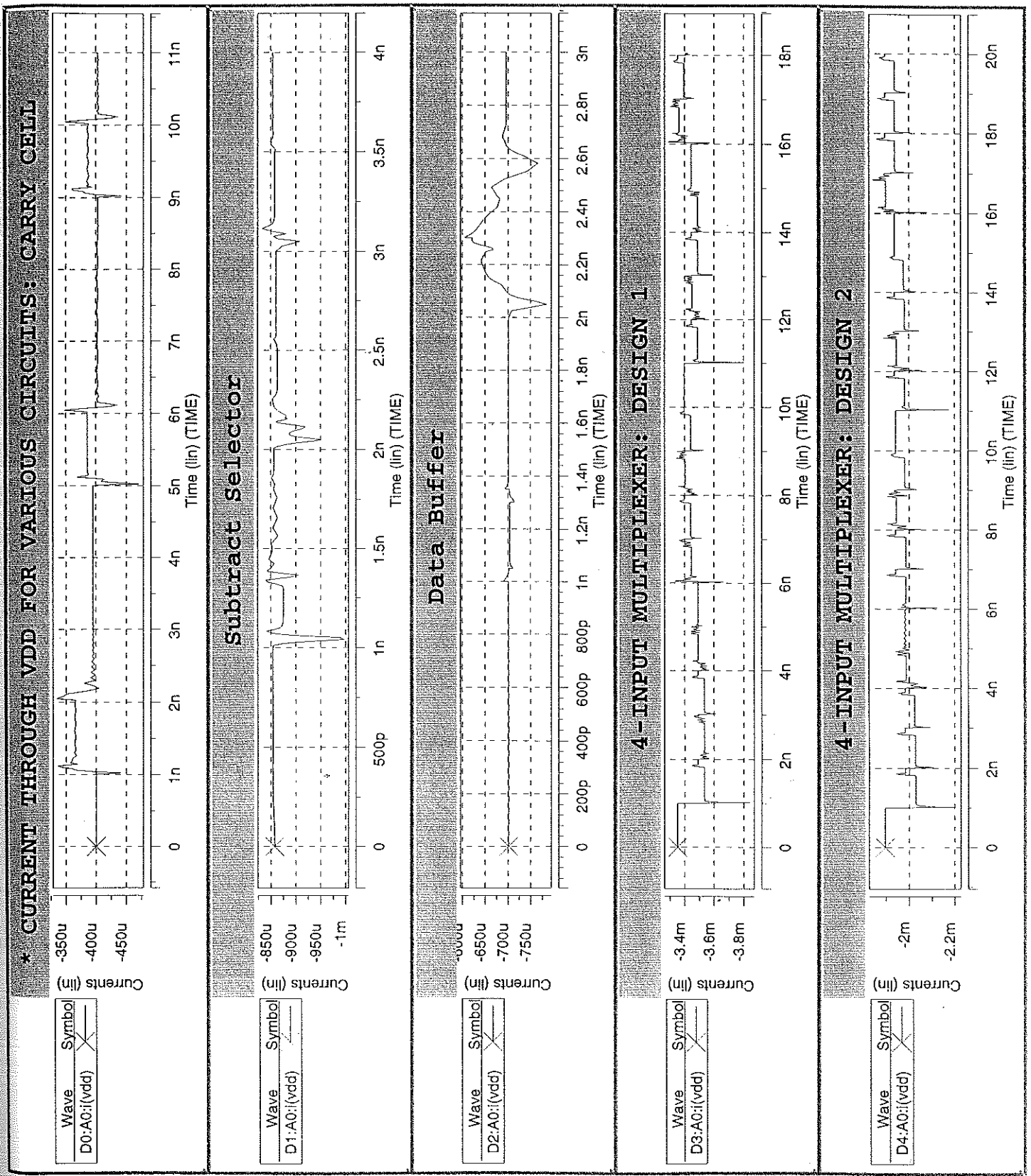
As seen from the table, even simple combinational circuits in MESFET GaAs consume considerably more power than more complex PDLL based self-timed circuits.

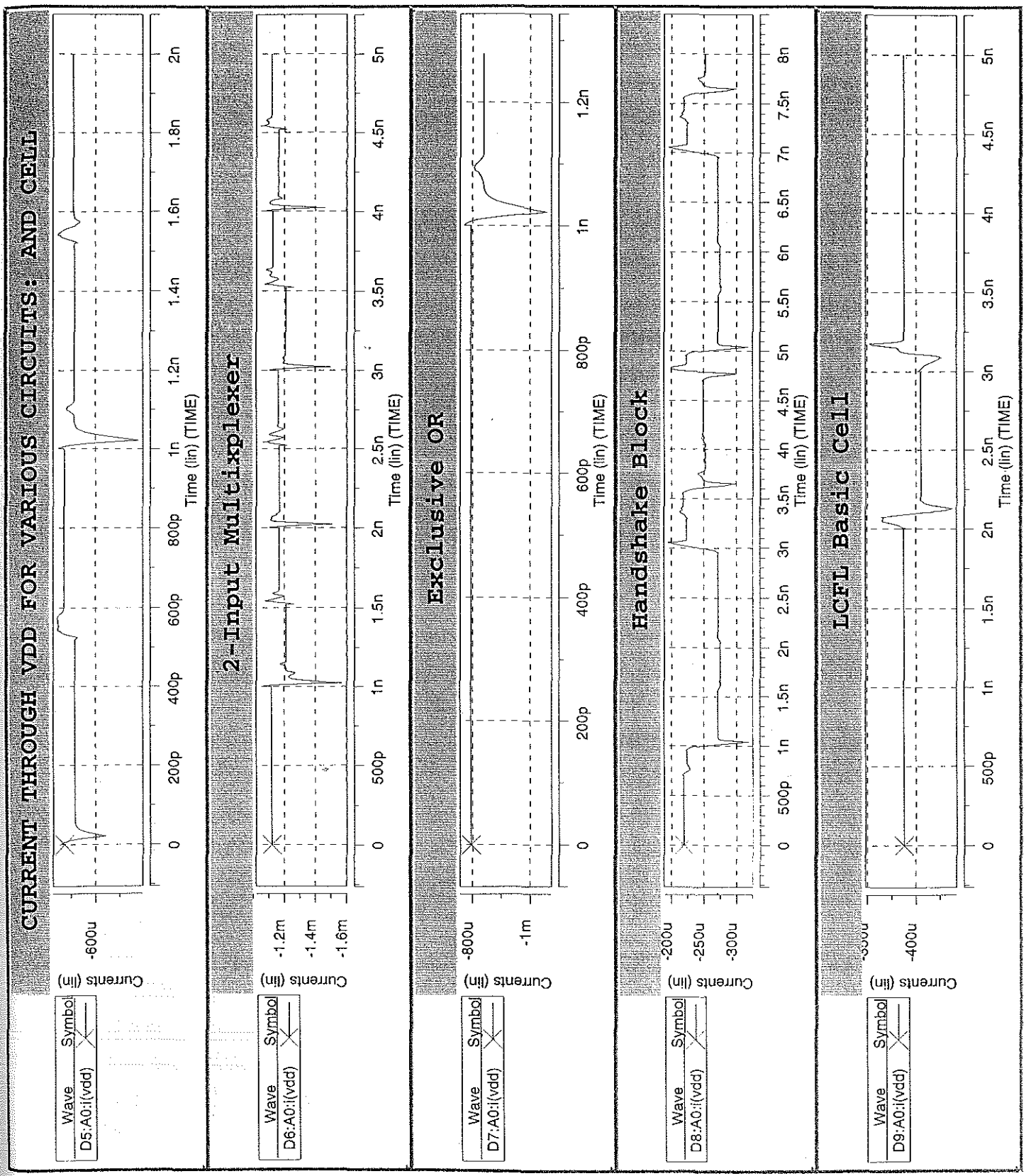
Table 5.14 shows the dimensions of various cells implemented. The number of transistors used in design is also shown, thus deriving the layout density. All these figures are integrated together to find the parameters for the final filter cell.

Cell	Height j	Width j	Area (micron^2)	Transistors	Transistors Per mm2 (1000's)
Register Cell	227	189	6864	19	3.96
Register	227	1100	39952	190	6.77
Muller C	84	70	941	8	10.99
AND	109	54	942	4	5.54
XOR	136	90	1958	8	4.95
LCFL Basic	92	96	1413	6	6.35
Handshake	161	151	3890	20	5.83
2x1 Mux	137	107	2345	10	5.35
4 x 1 Mux	152	349	8488	30	4.74
4x1 Mux 2	202	191	6173	22	4.43
WAIT	126	105	2117	8	4.65
Subtract Selector	124	100	1984	6	3.97
SUM	137	191	4187	18	5.29
Carry	139	143	3180	12	4.97
Filter Cell	**1000**	**800**	**128000**	**328**	**3.73**

Table 5.14 Cells and Layout Dimensions


CURRENT THROUGH VDD FOR VARIOUS CIRCUITS: REGISTER
Wave
Symbol
D10:A0:i(vdd)
Currents (lin)
-9.2m
-9.4m
0
5n
10n
15n
20n
25n
30n
35n
40n
45n
50n
55n
60n
65n
70n
Time (lin) (TIME)
SUM CELL
Wave
Symbol
D11:A0:i(vdd)
Currents (lin)
-350u
-400u
-450u
-500u
0
1n
2n
3n
4n
5n
6n
7n
8n
9n
10n
11n
Time (lin) (TIME)
WAIT CELL
Wave
Symbol
D12:A0:i(vdd)
Currents (lin)
-1.4m
-1.45m
-1.5m
-1.55m
0
500p
1n
1.5n
2n
2.5n
3n
3.5n
4n
4.5n
5n
5.5n
6n
Time (lin) (TIME)


* CURRENT THROUGH VDD FOR VARIOUS CIRCUITS: CARRY CELL
Wave Symbol
D0:A0:i(vdd)
Currents (lin)
-350u
-400u
-450u
0 1n 2n 3n 4n 5n 6n 7n 8n 9n 10n 11n
Time (lin) (TIME)
Subtract Selector
Wave Symbol
D1:A0:i(vdd)
Currents (lin)
-850u
-900u
-950u
-1m
0 500p 1n 1.5n 2n 2.5n 3n 3.5n 4n
Time (lin) (TIME)
Data Buffer
Wave Symbol
D2:A0:i(vdd)
Currents (lin)
-650u
-700u
-750u
0 200p 400p 600p 800p 1n 1.2n 1.4n 1.6n 1.8n 2n 2.2n 2.4n 2.6n 2.8n 3n
Time (lin) (TIME)
4-INPUT MULTIPLEXER: DESIGN 1
Wave Symbol
D3:A0:i(vdd)
Currents (lin)
-3.4m
-3.6m
-3.8m
0 2n 4n 6n 8n 10n 12n 14n 16n 18n
Time (lin) (TIME)
4-INPUT MULTIPLEXER: DESIGN 2
Wave Symbol
D4:A0:i(vdd)
Currents (lin)
-2m
-2.2m
0 2n 4n 6n 8n 10n 12n 14n 16n 18n 20n
Time (lin) (TIME)


CURRENT THROUGH VDD FOR VARIOUS CIRCUITS: AND CELL
Wave Symbol
D5:A0:i(vdd)
Currents (lin)
-600u
0 200p 400p 600p 800p 1n 1.2n 1.4n 1.6n 1.8n 2n
Time (lin) (TIME)
2-Input Multiplexer
Wave Symbol
D6:A0:i(vdd)
Currents (lin)
-1.2m
-1.4m
-1.6m
0 500p 1n 1.5n 2n 2.5n 3n 3.5n 4n 4.5n 5n
Time (lin) (TIME)
Exclusive OR
Wave Symbol
D7:A0:i(vdd)
Currents (lin)
-800u
-1m
0 200p 400p 600p 800p 1n 1.2n
Time (lin) (TIME)
Handshake Block
Wave Symbol
D8:A0:i(vdd)
Currents (lin)
-200u
-250u
-300u
0 500p 1n 1.5n 2n 2.5n 3n 3.5n 4n 4.5n 5n 5.5n 6n 6.5n 7n 7.5n 8n
Time (lin) (TIME)
LCFL Basic Cell
Wave Symbol
D9:A0:i(vdd)
Currents (lin)
-400u
0 500p 1n 1.5n 2n 2.5n 3n 3.5n 4n 4.5n 5n
Time (lin) (TIME)

6 Conclusion

As stated in the project definition, the aim of this project was to design a triangular (three tap) FIR filter capable of performing a discrete wavelet transform. This DWT algorithm was required to be implemented in an Intelligent Pixel (IP). The main objectives relating to this aim were:

- Investigation of properties of Gallium Arsenide, and a study of various transistor devices possible.
- Understanding the architectural requirements for an FIR filter cell to be able to perform a discrete wavelet transform within the Intelligent Pixel.
- The implementation of an FIR filter cell in Gallium Arsenide.
- Comparison of hardware and software simulations.

6.1 Project Achievements and Contributions

The original contributions of this project are as follows.

- A study of GaAs and its logic families, design styles and performance characteristics was done. Data and information from various sources relating to self-timed systems, the MRN and various logic families was presented in a cohesive form.

- A theoretical coverage of wavelets and wavelet transforms and their applications to image compression was provided. This material was adopted from various sources and provides a lucid explanation of concepts of this conceptually difficult mathematical subject. The process of wavelet transformation and its merits were discussed and finally, the triangular wavelet transform was analysed.

- The triangular wavelet transform algorithm was evolved and the filter architecture was evolved based on the specification of the transform. This demonstrated the constitution of a circuit design from given specification.
- The filter design was implemented in H-GaAs III technology. Various issues of self-timing and design style were discussed and incorporated into the project.
- The implementation of filter design provides not only a functional layout, but also gives background information of the various issues to be considered for future projects in this area. The results obtained can be used as a benchmark to evaluate other current and future projects related to the field.

6.2 Comments and Recommendations for Future Research

The project scope can be extended by performing a VHDL analysis on the filter architecture. The VHDL option for testing and simulation of the system was forsaken because of unavailability of software licence and time constraints during the course of the project.

During the implementation of filter, there were a lot of design changes and re-design efforts made on the circuit level. These changes also translated onto the layout implementation, leading to quick-fix adaptations at a later stage. Development of a well-defined electrical layout for the filter cell, along with a review of the design style, floor plan and such other issues would result in more sophisticated, compact and optimised designs.

Wavelets and related theory are mathematically intensive concepts, requiring extensive reading of texts and related papers. Emphasis should be laid on establishing a strong background on wavelet transforms before commencing the initial design for related circuit systems. As a part of further research, an analysis of various wavelet transforms and comparison of their performance with the triangular wavelet transform using software-based simulation should be undertaken. This would provide a better perception of the overall scheme of things and be an interesting primer for future projects in this area.

On the whole, the project contributed sucessfully to the research activites at the Centre for Very High Speed Microelectronics, Edith Cowan University. Not only did it provide a working model for various components of the FIR filter system, it will also act as a case-study for future design efforts in this area.

Appendices

Appendix A: Algorithm for MATLAB Analysis of Triangular Wavelet Transform.

Beginning with a one-dimensional row of data. We obtain the coefficients for high and low pass on the row.

The values of the coefficients L and H of the first step are;

$L_{11} = X_{11}$, $H_{11} = -(X_{11}/2) + X_{12} - (X_{11}/2)$, $L_{12} = X_{13}$

$L_{21} = X_{21}$, $H_{21} = -(X_{21}/2) + X_{22} - (X_{21}/2)$, $L_{22} = X_{23}$

The inverse transform function can be calculated adding the next values

$X_{11} = L_{11}$, $X_{12} = (L_{11}/2) + H_{11} + (L_{12}/2)$, $X_{13} = X_{12}$

The general equation for obtaining the high pass and low pass after the first pass filtering.

Low Pass $L_k = X_{2k-1}$

High Pass $H = -\frac{1}{2}(H_{2k-1}) + H_{2k} - \frac{1}{2}(H_{2k+1})$

And for inverse transform

$X_k = L_{(K+1)/2}$, for K = odd values

$X_k = \frac{1}{2} L_{(k/2)} + H_{(K/2)} + \frac{1}{2} L_{(K/2)+1}$

These general equations can be applied to matrix operations as shown in the following algorithm for MATLAB based wavelet transform.

Start

1. Read target image into n x n matrix.
2. Pad '0's to the column and row beginning and end.

   Obtain 2 Matrices: $L_k = X_{2k-1}$ and

$H = -\frac{1}{2}(H_{2k-1}) + H_{2k} - \frac{1}{2}(H_{2k+1})$. Decimate alternate values of H and L (Remove 0s)

3. Form a new matrix with interleaved H and L coefficients.
4. To perform a columnar transformation, transpose the new obtained matrix and pass it through step 2 and 3.

And for Inverse Transform

1. Form two matrices, Odd and Even
2. Read contents of L into Odd. Interleave Odd with alternate '0's, starting position 2. Call the new Matrix Y.
3. Interleave H with '0's, starting position 1. Perform $Z = \frac{1}{2}Y_k + H + \frac{1}{2}Y_{k+1}$.
4. To obtian columnar inverse transform, transpose Z and perform steps 1 through 3 again.

The final matrix Z should represent reconstructed image. Compression can be achieved by discarded the low coefficients after a transform.

The signal to noise ratio on such a transform and compression is calculated as

Original = Signal

Reconstructed = Signal + Noise

$\therefore$ SNR = Signal/Noise

= Original/(Reconstructed – Original)

The values for original and reconstructed image are the RMS (root mean square) value of all the pixels in the image. Such algorithm was applied to the image "Lena" using MATLAB software in the VLSI research laboratory. The results of compression on the image shall be demonstrated visually during the seminar presentation for the project.

Bibliography

[1] S. W. Lachowicz, K. Eshraghian and J. Lopez, "GaAs self-Timed filter bank architecture", *IEEE Transactions on VLSI Systems.* Copy provided by Dr. S. W. Lachowicz.

[2] S. W. Lachowicz, K. Eshraghian, and H. J. Pfleiderer, "Efficient low-lower circuit design with Gallium Arsenide latched logic". *Proceedings of the ISIC-97 Conference*, pp 158-161, Singapore, September 1997.

[3] K. Eshraghian, S. W. Lachowicz, D. Lucas and A. Rassau, "Design of low power, high density Gallium Arsenide asynchronous primitives for multimedia computing", *Proceedings, ASILOMAR'97 Conference on Signals and Computers*, Pacific Grove, CA, November 1997.

[4] S. W. Lachowicz, K. Eshraghian, J.F. Lopez, H.J. Pfleiderer, "Design of self-timed Gallium Arsenide integrated systems for multimedia computing". Copy provided by Dr. S. W. Lachowicz.

[5] K. Eshraghian, S. W. Lachowicz, G. Alagoda and K. Ang, "Architectural mappings for multimedia smart-pixel arrays". Copy provided by Dr. S. W. Lachowicz.

[6] S. W. Lachowicz, K. Eshraghian and H.J. Pleiderer, "Design methodology for Gallium Arsenide self-timed systems". Copy provided by Dr. S. W. Lachowicz.

[7] J. F. Lopez, K. Eshraghian, R. Sarmiento, A. Nunez and D. Abbott, "Gallium Arsenide pseudo-dynamic latched logic for high performance processor cores", *IEEE journal for solid state circuits.* Vol. 32 (8), pp. 1297-1303, August 1997.

[8] S. W. Lachowicz and Hans-Jorg Pleiderer, "Analysing the noise margin for Gallium Arsenide circuits: DCFL", Copy provided by Dr. S. W. Lachowicz.

[9] A. S. Lewis and G. Knowles, "VLSI architecture for 2-D Daubechies wavelet transform without multipliers", *Electronic Letters*, Vol. 27(2), pp. 171-173, 1991.

[10] Vitesse Semiconductor Corporation, "Foundry Design Manual", 1993.

[11] S. Cui, "Hardware mapping of critical paths of a GaAs core processor for solid mo[illegible]g accelerator", *PhD Thesis,* The University of Adelaide.

[12] Lop[illegible]-Sosa, A., "Smart Pixel Architectures", *Unpublished Booklet*, Centre for Applie[illegible]Microelectronics, University of Las Palmas de G.D., 1998.

[13] Long, S. I. And Butner, S. E., "Gallium Arsenide Digital Integrated Circuit Design". *Mc Graw Hill Inc*, 1990.

[14] Weste, Neil H. E. and Eshraghian, K., "Principles of CMOS VLSI Design, A systems perspective", *Addison Wesley*, second edition, 1992.

[15] Eshraghian, K. and Pucknell, D., "Basic VLSI Design", *Prentice Hall Publications*, third edition, 1994.

[16] Strang. G. and Nguyen. T., "Wavelets and Filter Banks", *Wellesley Cambridge Press*, 1996.

[17] Chu, Tam-Anh., "Synthesis of Self-timed VLSI Circuits for Graph-theoretic Specifications", *PhD Thesis*, Massachusetts Institute of Technology, 1987.

[18] Mead, C. and L, Conway., "Introduction to VLSI Systems", *Addison Wesley*, 1981.

[19] Muller, D. E. and Bartky, W. S., "A theory of asynchronous circuits", *Annals of Computation Laboratory of Harvard University*, Harvard Univ. Press, 1959.

[20] J. B. Dennis and S. S. Patil., "Speed-independent asynchronous circuits", *Proceedings of the fourth Hawaii conference on System Sciences*, pp 55-58, 1971.

[21] Mitchell, J. L. et al, "MPEG Video Compression Standard", *Chapman and Hall*, 1996.

[22] Huelsman, L. P., "Active and Passive Analog Filter Design", *Mc Graw Hill*, 1993.

[23] Torres, L. and Kunt, M., "Video Coding. The Second Generation Approach", *Kluwer Academic Publishers*, 1996.

[24] Riley, M. and Richardson. I., "Digital Video Communications", *Artech House Publishers*, 1997.

[25] Ozer, J., "Video Compression for Multimedia", *Academic Press, Inc*, 1995.

[26] Relue, R., "Parallel Processing Implementation of Wavelet Transform", *Project Report*, Mines Institute, 1994.

[27] Riley, M and Richardson, I., "Digital Video Coding", *WWW Site: "http://jura2.eee.rgu.ac /dsk1/digvid/dvhome.html"*, Robert Gordon University, 1996.

[28] Gibson, D., "Finite State Machines", *WWW Site: "http://www.microconsultants.com/tips/fsm/fsmartcl.htm"*, Microconsultants (Aust.) Pty Ltd.

[29] Burrus, C.S., Gopinath, R.A. and Guo, H., "Introduction to Wavelets and Wavelet Transforms: A Primer", *Prentice Hall Publications*, 1997.

[30] Jeld, K. "Introduction to Petri Nets", *WWW Site: "http://www.daimi.aau.dk/~khm/hot_petri.html"*, 1998.

[31] M.A. Cody, "The Fast Wavelet Transform," *Dr. Dobb's Journal*, 1992, pp. 16-28.

[32] J. Lu, "Parallelizing Mallat Algorithm for 2-D Wavelet Transforms," *Information Processing Letters*, 1993, pp. 255-259.

[33] J. Lu, V.R. Algazi, and R.R Estes "Comparison of Wavelet Image Coders Using the Picture Quality Scale (PQS)", *In Wavelet Applications II Proceedings SPIE*, Vol. 2491, pp. 1119-1130, 1995.

[34] Whyte, P., "Design and implementation of high-radix arithmetic systems based on the SDNR/RNS data representation", *Engineering Project Report*, Edith Cowan University, 1997.