Near and Sub-Threshold Null Convention Logic Design for Low-power Digital Signal Processing Applications

A thesis submitted in fulfilment of the requirements for the degree of Masters of Engineering

Renuka Mandar Sovani
B.E. (Hons) Electrical and Electronics Engineering, BITS, Pilani, India

School of Electrical and Computer Engineering
College of Science Engineering and Health
RMIT University

June 2016
Declaration of Authorship

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Renuka Mandar Sovani

Date: June 27, 2016
Abstract

Portable devices such as heart monitors, pacemakers and hearing aids requiring speech frequency filtering, can achieve low power operation if operated with reduced supply voltages near the threshold of their transistor components. However, in this region the effects of Process, Voltage and Temperature (PVT) variations are much more prominent. As a result, low-voltage clocked systems can often require much greater design effort and additional resources to be applied to avoid PVT-related failures, potentially leading to excessive and undesirable design margins.

Asynchronous techniques are generally more tolerant to these variations and have been suggested for use in low power, wearable applications. Out of the range of available asynchronous techniques, Null Convention Logic has been shown to be simple to design and robust in the face of PVT variability. Its main disadvantages are that its basic components are bulky and current implementation tools are not optimized for this type of asynchronous design. Further, NCL standard cell libraries are rarely, if ever, available as part of a vendor-supplied process kit for integrated circuit manufacture.

Furthermore, Short word-length (SWL) filter systems that operate, for example, on single bit Sigma-Delta encoded data have also been suggested for low-power and portable systems. Synchronous short word length filters using oversampling of both data and coefficients and operating in this domain have already been shown to be typically more hardware efficient and give better performance than their equivalent multi-bit counterparts. However, while Sigma-Delta techniques do serve to reduce the overall complexity of the hardware, their use of oversampling to move the quantization noise out of the region of interest results in many more filter stages than in the conventional case. Thus, it is not immediately clear whether the technique will always result in more efficient filter circuits.
This research explores the concept of using a Short Word Length filter architecture in combination with Null Convention Logic to gain the benefits of reduced complexity in the multiply/accumulate (MAC) architecture as well as the robustness of NCL against circuit variability when operated at or near the threshold voltage of the transistor components.

A single bit binary Finite Impulse Response Sigma-Delta digital filter has been implemented using Null Convention Logic that can be operated down to 400mV while achieving a sampling frequency target of 8kHz, suitable for speech and other low bandwidth applications. A dedicated standard cell library of static NCL gates was designed, including layout using Cadence Virtuoso, and simulated in a readily available 130nm CMOS process. The performance, area and power of the filter built using these standard cells were measured and compared against an equivalent synchronous filter design using a similar process.

It was found that, once initialized, the filter operates comfortably within a sampling rate of 8kHz at 400mV at both the typical and slow process corners, indicating that the filter operation will track PVT variations down to this point. The average power at the typical corner with supply voltage of 400mV for 32-tap filter is 11.8µW for the proposed design as compared to 32.7µW for an equivalent synchronous design. The average power for 16 tap filter operating with 400mV and typical corner is 1.6µW for the proposed design as compared to 8µW for the equivalent synchronous design. The optimised 32-tap filter exhibits a power-delay product as low as 5.3pJ at 400mV supply, which is significantly lower than approximately equivalent designs from the literature.

Finally, while the physical area occupied by the asynchronous filter components is much larger than the equivalent synchronous system, the simulations performed in this work have indicated that at least some of this area can be reclaimed in the place and route flow as its regular structure and short word representation reduce the complexity of the interconnection network.
Acknowledgements

I would like to express my sincerest gratitude to my supervisors Associate Professor Paul Beckett and Dr. Glenn Matthews. They have been encouraging and patient throughout my research for the past two years. Dr Paul Beckett has introduced me to the field of asynchronous design techniques and its use in signal processing and has taught me to look at various approaches for solving a given problem. He also gave me an opportunity to carry out my research by using the rich EDA tool utilities available at the University laboratory. He has been a constant source of inspiration to me during the design and data analysis. He has guided me with his rich knowledge and expertise whenever I seem to have lost direction.

I would like convey my thanks to all the helpful RMIT University staff. Special thanks to all my research group members Prashant, Kashfia, Jing, Matthew, Yash, Andrew, Tahn, Huy, Lin, Sayazila for their help in tool setup and technical discussions which helped in keeping my knowledge base up to date.

I also express my gratitude to Wave Semiconductor Inc. for giving us access to a beta version of the NELL hardware description and synthesis tool.

My friends in Melbourne have played an important role in my life in Melbourne and I want to thank them specially for that.

I express my sincerest gratitude towards my parents, my father Sudhir Dixit, mother Snehal Dixit for encouraging me to embark on this journey and supporting throughout. I would like to thank my sisters Revati and Radhika and brother-in-law Omkar for constantly pushing me to finish work in timely manner. I would also like to thank grandfather Gajanan Sovani, father-in-law Suresh Sovani and mother-in-law Surekha Sovani and all extended family members.

I do not have enough words to express my appreciation to my husband Mandar Sovani whose love, affection and sincerity has kept me calm. He has always been there whenever I needed any support. Our adorable daughters Maitreyee and Mrunmayee have helped me in the final push towards completion with their beautiful smile.
# Contents

Declaration of Authorship ............................................ i

Abstract ........................................................................ ii

Acknowledgements ............................................................. iv

List of Figures .................................................................. vii

List of Tables ...................................................................... ix

Abbreviations .................................................................... x

1 Introduction .................................................................... 1
  1.1 Research Motivation .................................................... 5
  1.2 Research Questions .................................................... 6
  1.3 Methodology and Main findings ..................................... 6
  1.4 Introduction to the key concepts used in this work .......... 8
    1.4.1 Sub-threshold operation of digital circuits ................. 8
    1.4.2 Sub-threshold variation and clock constraints ............ 10
    1.4.3 Null Convention Logic ............................................ 11
    1.4.4 Short word length systems .................................... 14
  1.5 Publication arising from this work ................................. 16
  1.6 Thesis Organisation .................................................... 17

2 Literature Review ........................................................... 18
  2.1 Variability in CMOS .................................................... 19
    2.1.1 Process Variability in Nanoscale CMOS ................. 19
      2.1.1.1 Dimensional Variations ................................ 20
      2.1.1.2 Dopant Fluctuations ................................... 20
      2.1.1.3 Variation in gate oxide thickness ................. 20
    2.1.2 Supply Voltage Effects in CMOS ............................. 21
    2.1.3 Temperature Variability ....................................... 21
  2.2 NCL and Timing Considerations ................................... 22
  2.3 Subthreshold NCL ...................................................... 24
  2.4 SWL Filters .............................................................. 27
2.5 Summary ................................................................. 29

3 NCL Threshold Gate Library Design 30
3.1 Threshold gate design .............................................. 31
  3.1.1 Schematic Design .......................................... 31
  3.1.2 Performance Measurement ............................ 32
  3.1.3 Layout Creation ................................. 35
3.2 Timing characterization ........................................... 36
3.3 Summary ............................................................... 38

4 SWL NCL Filter Design Implementation and Verification 40
4.1 Basic Single Bit NCL Sigma–Delta Domain Filter Design ... 40
  4.1.1 Filter Design ............................................. 41
  4.1.2 HDL Filter Description ............................. 41
  4.1.3 NCL threshold gate mapping ................. 45
  4.1.4 Filter Performance .................................. 45
4.2 Performance optimized single bit NCL Sigma–Delta filter design ... 47
  4.2.1 Filter Design and Implementation .............. 47
  4.2.2 Performance Optimization ...................... 49
  4.2.3 Optimized Filter Performance .............. 53
4.3 Verification and Debug ........................................ 54
  4.3.1 Verification .......................................... 54
  4.3.2 Debug ................................................. 54
4.4 Discussion and Comparison .................................. 56
4.5 Summary ............................................................... 59

5 Summary, Conclusions and Future Work 60
5.1 Summary ............................................................... 60
5.2 Conclusions .......................................................... 61
5.3 Future Work ........................................................... 63

References 65

Appendix A:
NCL TH gates: Schematic, Layout and Performance 72

Appendix B:
HDL code for the basic SWL NCL Filter 76
# List of Figures

1.1 $I_{DS}$ vs $V_{GS}$ characteristic curve for nmos ................................. 9
1.2 Monotonically alternating wavefronts of completely data and completely not data. ......................................................... 13
1.3 State holding behavior of NCL 3 of 5 operator ................................. 14
1.4 NCL Threshold Gate - static implementation ................................. 14
1.5 NCL pipeline cycle with single value variable data path .................... 14
1.6 Simplified Structure of Short-word Length FIR filter ....................... 16
2.1 Effect of temperature variation on propagation delay over supply voltage. 22
2.2 Generalized Overview of a Digital Signal Processing Path .................. 29
3.1 TH22 design process .......................................................... 32
3.2 TH22 gate delay and static power per transition ............................. 35
3.3 TH22 gate delay and dynamic power per transition .......................... 35
3.4 TH22 Schematic and layout ............................................... 36
3.5 Library Characterization Testbench ........................................... 37
3.6 Partial View of Characterized Library ...................................... 38
4.1 Design Characteristics of 30Hz FIR filter .................................... 41
4.2 SWL Filter Implementation - Partial View .................................... 42
4.3 Data delay line alternate DATA and NULL register implementation ...... 43
4.4 Coefficient DATA and NULL generation ..................................... 44
4.5 Basic filter operation with externally controlled 50uS data-to-data request rate ......................................................... 45
4.6 Basic filter operation showing maximum frequency of operation with data request rate controlled by acknowledgement from the design .... 47
4.7 Design Characteristics of 32-tap FIR Filter: normalized magnitude response .......................................................... 48
4.8 Impulse Response of 8KHz filter – Original 32 coefficients and interpolated samples ......................................................... 48
4.9 SWL Filter Implementation - partial view ..................................... 49
4.10 sample request signal rate controlled by the circuit ......................... 50
4.11 SWL Filter Implementation and Speed Optimization using “bubbles” - partial view ......................................................... 51
4.12 Performance Optimized SWL NCL filter with 512 coefficients .......... 52
4.13 Basic and Optimized filter performance area comparison ................. 52
4.14 32tap filter delay at 400mV supply ........................................... 54
4.15 Filter Output Response: Hardware Implementation versus MATLAB output .......................................................... 55
1  T13 Schematic and Layout ........................................... 72
2  T13 Propagation Delay and Switching Threshold across supply voltage,  
    input slew and output load ........................................ 73
3  TH33 Schematic and Layout ........................................... 73
4  TH33 Propagation Delay and Switching Threshold across supply voltage,  
    input slew and output load ........................................ 74
5  TH23 Schematic and Layout ........................................... 74
6  TH23 Propagation Delay and Switching Threshold across supply voltage,  
    input slew and output load ........................................ 75
List of Tables

1.1 Asynchronous logic delay models ........................................ 3
1.2 TH22 Delay and Power vs. Supply Voltage .......................... 9

2.1 Area/Delay figures from [46] ............................................. 25
2.2 Area/Delay figures from [49] ............................................. 27
2.3 FIR filter comparisons from [50] ....................................... 27
2.4 Area-performance comparison of single-bit FIR vs. multi-bit filter: non-pipelined Mode. ........................................... 28

3.1 TH22 Performance ......................................................... 34

4.1 Filter function ............................................................. 46
4.2 Corner Analysis of 32 tap filter ......................................... 53
4.3 Comparative Area-Performance-Power Results for Asynchronous Filters .................................................. 57
4.4 Average power and energy per transition comparison of TH22 for CMRF7SF and CMRF8SF IBM technology ........................ 59
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWL</td>
<td>Short Word Length</td>
</tr>
<tr>
<td>NCL</td>
<td>NULL Convention Logic</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>OCV</td>
<td>On Chip Variation</td>
</tr>
<tr>
<td>QDI</td>
<td>Quasi Delay Insensitive</td>
</tr>
<tr>
<td>PVT</td>
<td>Process Voltage Temperature</td>
</tr>
<tr>
<td>PMOS</td>
<td>P-channel Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>NMOS</td>
<td>N-channel Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>FET</td>
<td>Field Effect Transistor</td>
</tr>
<tr>
<td>MAC</td>
<td>Multiply and Accumulate</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Traditionally, the performance of a digital logic system has been more-or-less defined by its clock. However, as silicon geometries shrink into the nanometer region, the effects of Process, Voltage and Temperature (PVT) variability are beginning to cause conventional, globally synchronous design techniques to falter. While asynchronous (clock-less) techniques have existed since the early days of integrated circuit design [1], it is still true today that the overwhelming majority of integrated circuits are based on synchronous architectures (that is, governed by a global clock). However, the flexible, local timing behavior of asynchronous techniques has led to a resurgence of interest in these systems. Their perceived benefits include a potential for lower power consumption and the elimination of global clock skew issues. There are less obvious advantages as well. For example, supply current is uncorrelated with the global clock, which can lead to simpler power supply design. Further, there is less need for area-consuming on-chip bypass capacitors, lower electromagnetic interference and a greatly increased resistance to reverse engineering attacks such as side-channel analysis. However, various problems such as the lack of a standardized tool flow have so-far prevented asynchronous techniques from becoming main-stream.

The need for low energy consumption will become increasingly important as we enter the era of wearable and portable devices that must run for extended periods without charging or which have to scavenge energy from the environment. Many low power techniques have already been developed, particularly for synchronous systems. These include clock and power gating, multi-threshold gate design, sleep mode activation, supply voltage
reduction, etc. Adjusting the supply voltage has the advantage of a square-law reduction in power often without requiring much (or even any) circuit modification.

When the supply voltage of a digital logic circuit is reduced to near, or just below the threshold of the component transistors, the circuit is said to operate in its sub-threshold region. It is already well known that a major advantage of digital sub-threshold operation is its potential for significant power/energy reduction [2]. However, moving the supply towards the transistor sub-threshold region causes both the absolute cell delays and their variability to increase significantly, forcing larger safety margins to be imposed on clocked designs and introducing other problems in the design of the clock tree [3].

The issues related to clock and PVT variability can be resolved to some extent if an asynchronous approach is taken. Asynchronous circuits do not use a clock signal to govern their operation but instead employ various types of handshaking signals to control the data flow and to indicate completion of an operation or a stage in the computation. Asynchronous circuit design styles can be defined in terms of the assumptions made about their delay properties and/or their data-path encoding. For example, in bundled-delay, completion is estimated using locally constrained delay modelling, whereas delay-insensitive designs can accommodate arbitrary delays through their circuit elements. Less restrictive assumptions about these wire and gate delays lead to more robust models in the face of PVT variations [4] but can increase power, performance and area overheads. An overview of the most important delay models and their primary properties [4] are shown in Table 1.1.

Data path encoding can be divided into two groups: bundled data-path design or bundled data encoding and quasi-delay-insensitive (QDI) data-path design or multi-rail encoding. Bundled data encoding uses similar logic to synchronous circuits i.e. one wire per data bit. The control signals are usually implemented explicitly using delay lines or domino logic. QDI employing a multi-rail encoding technique is the most robust data path design technique. In this case the inputs and outputs are in the form of m-of-n channels and the request or completion signal is implicit within the data and can be generated by simply ORing the outputs and detecting when the transitions are complete and valid.

As one of a small number of available QDI techniques (Table 1.1), NCL [5, 6] is a symbolically complete logic system that requires no external control signal to manage its data transfers as the logic signals themselves contain the timing information. As
### Table 1.1: Asynchronous logic delay models

<table>
<thead>
<tr>
<th>Style</th>
<th>Properties</th>
</tr>
</thead>
</table>
| Delay Insensitive (DI) | - most robust model to PVT variations  
                          - no assumption on wire or gate delays  
                          - each leaf cell must be multiple-output |
| Quasi DI            | - gate and wire delays can be anything except for a set of isochronic wire forks  
                          - primary input transitions to the design should be unordered |
| Speed Independent   | - gate delays can be anything but wire delays are negligible  
                          - all forks are isochronic  
                          - primary input transitions can be ordered  
                          - detailed post-layout analysis is needed for practical implementation |
| Scalable DI         | - no bound on gate or wire delay like DI but the ratio between two gate delays or two wire delays is bounded  
                          - comprises smaller sets of blocks communicating using DI model |
| Bundled Delay       | - both upper and lower bounds must be established for the gates  
                          - faster, smaller and lower power than its counterparts  
                          - needs complex design synthesis and post physical design timing analysis |
NCL timing automatically tracks the variation in component behavior and requires very little timing analysis it can be ideal for operating in the variability-prone sub-threshold region. A major disadvantage is that basic cells that form a Null Convention Logic circuit tend to occupy a much larger chip area than their synchronous counterparts, although this is offset to some degree by the higher functional density of the threshold logic.

This research work has focused on the application of NCL logic to digital signal processing, particularly those aimed at low-power applications such as wearable and portable physiological monitors, hearing aids and the like. In addition, there is increasing interest in asynchronous signal processing hardware to support non-uniform sampling for low activity signals [7].

The methodology is based on a Short Word Length (SWL) length digital filter using NCL logic operating at low supply voltages. As such, this work extends that of [8] into the asynchronous domain. A 1-bit Sigma-Delta (ΣΔ) Finite Impulse Response filter was designed using NCL and its power, performance and area was measured to determine its suitability for the target application domain. The basic idea was to exploit the advantages of asynchronous design and low voltage operation while regaining some of the lost performance by minimizing the overall complexity of the hardware, especially that of the large multiplier structures which form a major part of any DSP architecture.

While it has already been shown that fully asynchronous systems using NCL are feasible, e.g., [9, 10], it remains an open question whether using asynchronous NCL-based systems for portable applications will support both ultra-low power and simplicity of design. Since digital filtering is one of the major components of all signal processing devices, this provides a suitable platform to explore these tradeoffs. It is likely that short word length techniques can play important role here in simplifying the overall architecture, potentially making it more suited to an asynchronous implementation in terms of NCL.

To date, multi-bit filters have been a mainstay of DSP research and there has been a large amount of work reported on their simulation and fabrication. Much of this has been aimed at the efficient hardware implementation of FIR filters in Field Programmable Gate Array (FPGA) devices by using various complex multiplication reduction techniques [11–13]. More recently, Sigma-Delta modulated SWL, (i.e., 1- or 2-bit) FIR
systems [14, 15] have emerged as a viable alternative, especially in commercial audio systems.

The simple arithmetic of $\Sigma\Delta M$ can result in an efficient hardware implementation which is well suited for fabrication in both FPGA and standard cell ASIC environments. In particular the constrained, local connectivity of SWL filters makes them a suitable candidate for asynchronous techniques where long and random-length interconnect lines may introduce difficult timing issues. On the other hand, delay insensitive techniques such as NCL, are largely immune to these timing uncertainties, but can exploit the local connectivity of these filter organizations to reduce interconnection load and to constrain critical path length, thereby achieving high performance even at supply voltages near and below threshold.

1.1 Research Motivation

Systems built from semiconductor devices tend to fall between two extremes of the design spectrum depending on the trade-off between performance, area and power. At one end is design for high performance, where power is allowed to increase within some acceptable limit. At the other is ultra-low power design, given the system still achieves acceptable performance. The latter extreme is the subject of this research.

Wearable and battery operated devices such as hearing aids, heart pacemakers, electrocardiogram (ECG) monitors, wrist watches etc. clearly fall within the ultra-low power end of the spectrum. The battery used to support such an extremely low power budget has to be small in size while still meeting the performance goal. For example, in the case of hearing aids various surveys [16, 17] have indicated that battery life tends to be in the range of approximately 100 to 150 hours depending on its size. That means users typically have to change batteries approximately weekly, something that may be difficult for elderly and/or disabled users. Thus, obtaining the required performance within the stringent power/area budget is the most important challenge in such devices. As identified by [18], maintaining a given level of computation or throughput is a common concept in signal processing and other dedicated applications in which there would be little or no advantage in performing the computation faster than some given rate, since the processor will simply have to wait until further processing is required.
In summary, the primary response in this work to the objective of achieving low energy operation and extended battery life has been to explore the idea of working over a wide voltage range down to or near the transistor threshold point. The problems associated with working in this region (mainly wide PVT variability) have been addressed using a NCL asynchronous approach.

1.2 Research Questions

As will be shown in Chapter 2, of the prior work reported in the literature relating to NCL sub-threshold filter designs, virtually all concentrate on the efficient design of the central multiply/accumulate blocks, a small number have proposed efficient architectures while others have concentrated on the organization of the coefficient structures. However, all of these filters use conventional multi-bit representation of the data and coefficients. At the time that this work was conducted, there was nothing found in the available literature that explored the characteristics of SWL filter architectures in combination with NCL methodologies. This research fills this gap and is based on the idea of using SWL filter architecture in combination with NCL to exploit the benefits of reduced complexity in the multiply/accumulate structures as well as the regular structure of NCL systems to achieve the required performance (power, area, operating speed) targets. Based on this, the key research question that have guided the work are:

- Can the highly pipelined organization of SWL systems be exploited to produce usable NCL ultra-low-power DSP applications?
- Can filtering be performed using SWL NCL operating at very low supply voltage near the threshold (target 400mV) while meeting the sampling requirements for applications such as physiological data and speech processing?

1.3 Methodology and Main findings

The results reported in this thesis are derived from simulations of a number of single bit binary $\Sigma\Delta$ FIR filters designed using a 130nm IBM® planar CMOS process (cmrf8sf) using simulation models provided by the foundry. A custom-built NCL gate library
was constructed and its power, performance and area was measured. All the schematic and layout design, simulations and measurements were performed within the Cadence® Virtuoso environment. The Unified NCL Environment (UNCLE) [19, 20] was used in this work to perform cell library characterization. This tool is based on python scripts and is intended to automate the process of creating dual-rail asynchronous designs in NCL, although a different tool was used for that purpose in this work (as described below). During the cell characterization process, the tool repeatedly calls the Ultrasim simulator to perform input pin capacitance measurement and cell delay measurement for the target standard cell.

The filter design process commenced with a Register Transfer Level (RTL) description using a tool called NELL, a proprietary Verilog-like HDL, supplied under non-disclosure agreement by a U.S. start-up company working in the area of NCL-based architecture. This tool handles many of the details of the NCL process such as dual rail and completion logic etc. and outputs a description in System Verilog. Transferring this System Verilog description into a standard Cadence digital design tool flow allows the filter structures to be synthesized, placed and routed using the physical standard cell library designed for that purpose.

Overall, the work has entailed the following steps, which are detailed in subsequent chapters of this theses:

- NCL threshold gate library design at 130nm;
- Single bit ΣΔ filter analysis and design;
- Optimization;
- Physical implementation;
- Verification and debug;
- Power/Performance/Area measurements and feasibility analysis.

The main findings of this research work are:

- The filtering performance required by the applications requiring low power such as EEG and speech filtering applications can be obtained using ultra low supply SWL NCL operated down to ∼400mV;
• Physical area and external control of sampling rate are the main limitations of using NCL techniques in portable low-power applications;

• The design is made compact and less complex due to the fact that a single-bit SWL architecture is used in this work, which at least in part compensates for the area overhead of NCL systems and performance overhead of lower voltages.

In addition, other specific outcomes of this research work are:

• a 130nm NCL threshold gate library design optimized for operation at 300–400mV;

• an area optimized technique for generating DATA/NULL cycles in the fixed coefficient pipeline;

• a practical implementation of a performance-optimized tapped delay line using additional empty stages (“bubbles”).

1.4 Introduction to the key concepts used in this work

This work draws together a number of topics in digital systems design, including Null Convention Logic, Short Word Length DSP and sub-threshold operation. The following sections briefly set the context for the work by introducing the key concepts on which it is based.

1.4.1 Sub-threshold operation of digital circuits

When the supply voltage of a digital logic circuit is reduced to near, or just below the threshold of the component transistors (i.e. \( V_{DD} \leq V_{TH} \)), the circuit is said to operate in its sub-threshold region. This is illustrated in Fig. 1.1 which shows the current versus gate voltage for an N-channel MOSFET. The major advantage of digital sub-threshold operation is significant power reduction as is shown in Table 1.2. The table shows a NCL threshold gate TH22 that has been simulated with a supply varying from 1V down to 200mV. The power of this single gate varies from 2.4uW at 1V down to about 4.7nW at 300mV, a reduction achieved without any modification to the circuit or setup.
Note that as the average power is dominated by the switching power it depends to some degree on the overall simulation time. As a result, it is common in synchronous circuits to normalize the power figure to power per event, with the clock typically used as the reference (as an approximation to the switching “density”: events per second). The metric, which allows easier comparison between implementation styles, is therefore Watts/Hz (more typically, µW/MHz). In the asynchronous case, there is no clock to provide a normalization reference so, for every voltage point, the simulation time was simply arranged to be sufficient for the output(s) to settle [21] before measurements were taken and values has been reported for absolute power, power per switching event or power-delay-product (PDP), where appropriate.

Figure 1.1: $I_{DS}$ vs $V_{GS}$ characteristic curve for nmos

Table 1.2: TH22 Delay and Power vs. Supply Voltage

<table>
<thead>
<tr>
<th>Operating Voltage (volts)</th>
<th>Propagation Delay (nS)</th>
<th>Average Power per transition (nW)</th>
<th>$P_{STATIC}$ (pW)</th>
<th>Simulation time (nS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.175</td>
<td>2400</td>
<td>225</td>
<td>2</td>
</tr>
<tr>
<td>0.7</td>
<td>0.359</td>
<td>1240</td>
<td>118</td>
<td>2</td>
</tr>
<tr>
<td>0.5</td>
<td>1.28</td>
<td>254</td>
<td>68</td>
<td>5</td>
</tr>
<tr>
<td>0.4</td>
<td>4.6</td>
<td>40</td>
<td>48</td>
<td>20</td>
</tr>
<tr>
<td>0.3</td>
<td>29</td>
<td>4.7</td>
<td>31</td>
<td>100</td>
</tr>
<tr>
<td>0.2</td>
<td>231.6</td>
<td>0.313</td>
<td>18</td>
<td>600</td>
</tr>
</tbody>
</table>

Various authors (e.g., [22–24]) have shown similar power delay trends as the active energy is proportional to $V_{DD}^2$. Studies have shown that the minimum energy per operation occurs in the sub-threshold region. As identified in [22–24], other advantages of sub-threshold operation include:

- Independence of the drain current $I_{DS}$ from $V_{DS}$ for $V_{DS} \approx 76mV$ which under certain circumstances can allow more transistors to be stacked in series;
• Nearly ideal Voltage Transfer Characteristic (VTC) which gives better static noise margin;

• An exponential relation between $I_{DS}$ and $V_{GS}$, which results in high transconductance gain.

Of course, the major disadvantage is the fact that propagation delays increase significantly as the supply voltage levels are lowered. In the example presented in Table 1.2, delay increases from an average of $\approx 175\text{pS}$ at 1V to $\approx 29\text{nS}$ at 300mV potentially impacting the overall performance of the system. Any shortfall in performance would then need to be made up at the architectural level, via techniques such as parallel processing and multi-level pipelining [25].

A working hypothesis motivating this work has been that techniques such as the SWL NCL filter can achieve very short critical data paths due to the lack of complex multiplier structures and tightly constrained local routing arising from their inherently regular structures. Whether these characteristics are sufficient to overcome the requirement for very large shift register arrays resulting from the need to over-sample in the SWL domain has been a major focus of the work.

### 1.4.2 Sub-threshold variation and clock constraints

A critical problem with sub-threshold operation is its sensitivity to process, temperature and voltage variations which results in wide current drive variation. For example, as can be seen in Fig. 1.1, the current-voltage relation is exponential in the sub-threshold region as opposed to square-law in strong inversion, making it much more sensitive to variations in operating conditions. The International Technology Roadmap for Semiconductors (ITRS) predicts that process parameter and threshold voltage variations will increase from 11% and 42% for current 45nm technology to 32% and 112% respectively for 9nm technology in 2024 [26]. Variation in current translates into varying data path delay through the circuit.

To account for these variations, large safety margins must be added to the clock period in case of synchronous circuits, which can affect overall performance. Although many advanced Static Timing Analysis (STA) techniques exist to validate the design (such as on-chip variation de-rating, path based analysis etc.), it is extremely unlikely that
the variations will be able to be predicted accurately. Asynchronous approaches, which incorporate inherently self-correcting timing will be more robust against variability.

Further, the clock tree in a synchronous system is switched with 100% activity thereby consuming a large part of the power budget. Previous research \[27, 28\] indicates that this can be up to \(\approx 40\%\) of the total power. Building clock trees to meet stringent timing requirements also takes most of the routing and physical resources. Clock tree buffers and related logic (e.g. clock gating) all add to the area of a design. The clock trees are also affected by PVT variations in sub-threshold regions. The simultaneous switching of the clocked logic demands high peak currents which creates hot spots and the instantaneous voltage supply demand needs to be catered for by adding decoupling capacitors and additional power straps which in turn adds to the area and routing resource requirements.

1.4.3 Null Convention Logic

The issues outlined above relating to delay variation, clocks, engineering effort, etc. can at least in part be addressed if an asynchronous approach is applied to the design. Any performance lost when lowering supply voltage levels can also be reclaimed to some extent at an architectural level. Asynchronous methodologies use self-timed or localized signalling (e.g. handshake feedback) to control the sequencing of computations in the system and dispenses with the global clock \[29\].

While they offer significant advantages, most current asynchronous approaches are highly complex and difficult to work with. In contrast, the NCL asynchronous style is a quasi-delay-insensitive symbolically complete logic that is relatively easy to design and analyse \[29\]. NCL adds a “NULL” or “no-data” state to the two logic values (True and False) of conventional Boolean logic. Quasi delay insensitive (QDI) approaches such as NCL \[30\] require very few critical timing constraints, making them ideal choices for operating in unknown or widely varying delay regimes, such as in sub-threshold. The NCL paradigm developed by Karl Fant is explained in detail in \[5\]. Some of the aspects from \[5\] which are important in the context of this work are re-stated below.

NCL uses a “no-DATA” or NULL state in addition to its normal DATA state (equivalent to ‘True’ and ‘False’ states of binary logic), to indicate whether an input to output
transaction on a line is complete. Thus, NCL is a three valued logic that needs at least 2 rails to represent the two DATA states of Boolean logic (True and False). One voltage level (e.g., 0 volts) is used to represent NULL and other voltage level (say, 5 volts) is used to represent DATA value of either True or False depending on which of two rails is high. It is illegal for both the wires in same group to be high simultaneously. NCL can utilize multi-rail signals, such as dual-rail logic, to achieve quasi delay-insensitivity. In this way, a 1-bit binary signal (say, “A”) requires at least two signal rails, although a ‘one-hot’, 3-rail encoding is also possible. A typical representation might be: DATA A0A1=10 (False) and 01 (True) and NULL A0A1=00

NCL is symbolically complete, requiring no external authority such as a clock or delay line to signal the completion. NCL systems operate entirely in terms of synchronized wave fronts of monotonic level transitions. As shown in Fig. 1.2 [31], a transition from NULL state or completely not data to DATA state is called a data wavefront and indicates the beginning of valid data at the output. The output stays at DATA state until all the inputs go to NULL again and this transition from completely data to completely not data or NULL called NULL (not data) wavefront indicates that the DATA now is invalid and next DATA will be obtained.

NCL is commonly implemented using a set of “M-of-N” threshold logic gates, where N is the number of inputs, M is the threshold level and M ≤ N. Two important aspects of NCL gates are their threshold and hysteresis behaviour. Threshold behaviour means the output changes to the DATA value only when at least M of N inputs change to Data. Hysteresis behaviour means the output remains in its current state (DATA or NULL) until the input conditions for the opposite state are satisfied i.e., either all inputs go to NULL to drive the output from DATA to NULL or M inputs go to DATA to send the output from NULL to DATA. This behavior is further explained in Fig. 1.3 [31], which is a representation of a 3 of 5 operator. The number inside the operator indicates the threshold of the operator M, and the number of input connections are the value of N. The thick lines indicate that the signal is in its DATA state while the thin lines are NULL. If the output is in NULL state, it will remain in that state until the input data set reaches the threshold of 3 in this case at which point output becomes DATA. This is the ‘threshold’ behavior. Now it will remain in DATA state unless all the inputs go to NULL–demonstrating its ‘hysteresis’ behavior.
Fig. 1.4b shows the static implementation of TH23 gate or 2 of 3 operator. The output transitions to DATA (in this case, logic 1) only when 2-of-3 inputs go to DATA ('1') and transitions to NULL ('0') only when all the inputs similarly go to '0'. In all other conditions, the output remains unchanged. The NULL cycle is the natural sleep mode during which there are no spurious transitions until all the inputs settle to a valid value, thereby saving power. These M-of-N gates can be implemented in three styles of CMOS:

1. **Static hysteresis behaviour** is implemented exclusively using HOLD-DATA and HOLD-NULL circuit. Fig. 1.4a shows the general structure of the static implementation of NCL gates and Fig. 1.4b gives a specific example of a TH23 gate implementation.

2. **Semi-static hysteresis behaviour** is implemented using a loop comprising an output inverter and weak feedback inverter.

3. **Dynamic hysteresis behaviour** is maintained by internal node capacitance which can typically hold the current state for few milliseconds. These types of designs can be used in real time applications involving signal processing where input data stream is continuous and faster than the node voltage decay time.

In a similar way to the sequential–combinational structure of a clocked system, NCL systems can be arranged into structures called **cycles**, which in turn can be composed into pipelines through which data flows spontaneously [31]. In the data path in Fig. 1.5(from [31]), the cycle can be viewed in terms of a forward data path and a reverse acknowledge path. Here, the data path is just a single bit variable but could be easily extended. The 2 of 2 operator represents the registration stage, while the acknowledge signal is a simple binary signal forming a request for DATA or a request for NULL. This request is fed to the input of the previous register stage along with the other inputs to begin new DATA or NULL cycle. Every signal path in a system must be part of a cycle.

![Figure 1.2: Monotonically alternating wavefronts of completely data and completely not data.](image)
1.4.4 Short word length systems

Sigma-Delta techniques are inherently simple and have been widely adopted for processing audio and other low frequency signals although as CMOS technology improves they are also increasingly found in high-frequency RF systems [32]. In [33], ΣΔ conversion is identified as the technique most suited to digital audio products. The converter typically use a high sampling rate (called oversampling) operating at many times the Nyquist rate. This sampling results in a stream of single-bit (binary) or dual-bit (ternary) data interpreted as a difference signal of either +/-1 (in the binary case) or -1/0/+1 for a ternary...
encoding. To be clear, this is not the same as the NCL encoding. In this work, only a binary (single-bit) $\Sigma \Delta$ symbol encoding was considered, using a dual-line NCL scheme to encode the bit set 0, NULL, 1. The noise introduced by this coarse quantisation is moved by the oversampling to an out-of-band region which can then be easily removed by further filtering and decimation.

In a general sense, the SWL filter is identical in form to its multi-bit counterpart in that both represent a convolution process. The sampled data enters a large delay line (shift register), is multiplied at each point in the line by the corresponding filter coefficient and the resulting partial products then summed. A key advantage of short word-length systems is that they do not require complex integer multiplication hardware [15], which in the binary case is replaced by simple bit-wise AND gates. The area-power-performance characteristics of the SWL techniques have been extensively explored using FPGA implementations in [34] and [35] and have been found to offer significant power and performance advantage over their corresponding conventional (multi-bit) approaches. The ability to stay entirely in the sigma-delta sampling domain avoids the need to repeatedly transfer between sigma-delta and absolute binary. As already mentioned, it was considered that these pipelined organizations would be well suited to the inherently state-holding characteristics of NCL.

As a result, it can be seen that the overall architecture of a single-bit FIR filter shown in Fig.1.6 is virtually identical to its conventional counterpart, except that its taps are constrained to binary. The FIR filter output $y(k)$ is given by the convolution of the individual taps $h_i$ and the input signal $x(k)$ as follows:

$$y(k) = \sum_{i=0}^{M} h_i x_{k-i}$$

where $M$ is the order of the filter (≡ number of taps) and $h$ represents the binary FIR filter coefficients. The coefficients can be generated using a second order sigma delta modulator ($\Sigma \Delta M$) technique such as shown in [15]. Thus, a SWL filter comprises the conventional two stages i.e., multiplication of the coefficient taps with the binary input followed by the addition of the partial products. However, while the SWL multiplication stage reduces to a simple 2-bit AND function, greatly reducing its impact on area, the fact that the filter is operated at a high Oversampling Ratio (OSR) results in a large
number of taps, which increases the area again. For example, a FIR filter with 32 original (Nyquist rate) taps will require 1024 taps at an OSR of 32.

Figure 1.6 also indicates that from a generalized architectural point of view, the filter structure comprises a long shift register tapped at each stage. In the synchronous case, its overall timing will be determined by things like the fanout (path) delays of the clock signal and the propagation delay through the critical path, which is most likely here to be the tap addition network. The initial latency (before the first valid output signal occurs) is dependent on the number of register stages and the clock period. In the NCL case the completion handshaking at each node (register) includes a contribution from both the data pipeline and the addition tree and its timing will be determined by the worse case of these two sources. As an acknowledgement signal could, worse case, involve the entire length of the filter, long pipelines such as in this oversampled SWL architecture can result in slow performance and large latency [36]. In this work we have implemented NCL-related techniques such as optimising “bubble” and “wavefront populations”, as suggested in [5], to maintain a high throughput with low latency. This balance between filter length and performance will form part of the discussion in Chapter 4.

1.5 Publication arising from this work

A part of the material in this thesis has been published in the following:


1.6 Thesis Organisation

The remainder of this thesis proceeds as following:

In Chapter 2, the sources of variability in CMOS and its effect on timing analysis of the circuit is discussed with respect to both synchronous and NCL approaches. This is followed by a discussion of prior literature relating to sub-threshold NCL filter design and short word length filter organization.

Chapter 3 describes the NCL Threshold gates design, layout implementation and library characterization process followed in this work.

Chapter 4 covers the design and implementation of the single-bit binary FIR NCL asynchronous filter proposed in this work. It presents the design of the different circuit components, its simulations to measure its performance in terms of speed of operation, power dissipation and area, verification and debug process and discusses the results.

Finally Chapter 5 summarizes the work drawing conclusions and the future work arising from this.
Chapter 2

Literature Review

Much of the previous work relating to asynchronous techniques has focused on high performance applications and microprocessor design due to the well recognised benefits of asynchronous design styles. As an early example, a complete microprocessor based Digital Signal Processor was designed using asynchronous techniques in [37]. That work was based on a self-timed circuit with a four phase handshaking protocol sensing the timing from delays in the logic and thereby localising the timing behavior instead of relying on a global clock.

Since NCL implementations are inherently bulkier than their synchronous counterparts (typically in the range of 1.6-2 times [30]), a relatively smaller body of work has been reported relating to its application in portable or wearable systems that require moderate sampling rates. In [38] an architecture of an asynchronous FIR filter was proposed to handle irregularly sampled signals, in that case focusing on “bursty” speech signal sampling. Although the study focused more on the algorithm than the implementation technique, it did show that the energy efficiency of the asynchronous technique combined with the irregular sampling could be nearly an order of magnitude better than the equivalent synchronous case. In another asynchronous approach, the authors in [39] used a Memristor based storage circuit to design and simulate a 15-tap continuous time FIR filter circuit. The filter was intended to remove high frequency noise from electrocardiograph signals, and the simulations showed a low power consumption of $6.63\mu\text{w}$ at a 3.3V supply.
From these various examples, it is clear that asynchronous techniques can offer lower power/energy consumption that their synchronous counterparts. The key idea in that case is to greatly reduce the number of switching events in the processing path by exploiting techniques such as non-uniform sampling etc. In this thesis, the focus is on the robustness of asynchronous NCL circuits to PVT variability. This chapter first examines the issue of variability in CMOS then examines some of the previous work that has focused on NCL operating in the region near the threshold of the transistors focusing particularly on the resulting performance and power benefits.

2.1 Variability in CMOS

As CMOS technology continues to scale into the deep sub-micron region and then to nanometer dimensions, variability in the transistor behavior has become a major problem. Such variation tends to arise from combinations of uncontrolled changes in the manufacturing process, the supply voltage and the operating temperature. Collectively, this is referred to as PVT variability.

2.1.1 Process Variability in Nanoscale CMOS

Variation in CMOS due to manufacturing defects typically arises from variations in the physical dimensions (width, length oxide thickness, line edge roughness, etc.) of the transistors, plus random doping fluctuations within the source, drain and channel which in turn cause variations in the threshold voltage of the transistor components. As CMOS gate dimensions shrink it is becoming increasingly difficult to control these physical process parameters, resulting in a much wider standard deviation around a desired mean value.

These variations can be global, implying that the parameters change equally for all the transistors on the same wafer or they can be local meaning each transistor is affected differently during manufacturing. These changes in the process lead in turn to the variations of the electrical parameters of the design affecting gate delays, leakage current, capacitance, power etc.
2.1.1.1 Dimensional Variations

In nanoscale technology, the dimensions (widths and lengths of the transistors) are much smaller than the wavelength of the light used to image them. As a result, manufacturing processes like photo-lithography and etching cause variations in the nominal widths and lengths of the transistors and interconnection lines [40]. The drive current ($I_{DS}$) of MOSFET is directly proportional to the width/length ratio and is therefore directly affected by the variations in these. Fluctuations in the channel length also change its threshold voltage and shifts the drive current. Additionally the line-edge roughness i.e. the random variation in the gate length along the width of the channel, which is caused by statistical variations of the photon count or imperfections during photo-resist removal, changes the transistor’s ON to OFF current ratio. In turn, the propagation delay of the CMOS gate changes in proportion to that change in drive current [41].

2.1.1.2 Dopant Fluctuations

The number of dopant or impurity atoms added in the channel of the FET vary randomly due to implantation techniques used during manufacturing. In larger technology nodes, the absolute deviation of the dopant concentration is negligible but in smaller nodes even a small shift in the number can have a large effect, significantly shifting the carrier concentration and thus the drive strength of the transistor [41].

2.1.1.3 Variation in gate oxide thickness

Gate oxide thickness variation arise due to the deposition and growth process and chemical-mechanical planarization step. As the technology node shrinks to the nanometer range, the gate oxide becomes thinner and thinner, ultimately reducing to just few atomic layers. While a shift to more exotic gate materials such as Hafnium oxides has forestalled this problem somewhat [42], it it still the case that slight variations in thickness results in threshold variation, which in turn affects the drive current and therefore the transistor delay characteristics.
2.1.2 Supply Voltage Effects in CMOS

Variations in the supply voltage of the CMOS circuit arise due to voltage drops (i.e., $IR$ losses due to current flow through parasitic resistances in the power grid wires), noise due to the changing current ($di/dt$ caused by the inductance and capacitance of the power grid), inaccuracies in the voltage regulator or voltage reference circuit etc. [41]. The combined effect can either result in voltage drop or voltage overshoot changing the gate delay, $t_{\text{gate}}$, given by:

$$t_{\text{gate}} \propto \frac{V_{DD}}{b(V_{DD} - V_t)^a}$$  \hspace{1cm} (2.1)

where $a$ and $b$ are gate specific fitting parameters and $V_t$ is an effective threshold voltage. The path delay is the summation of these individual gate delays.

2.1.3 Temperature Variability

Power dissipated in the form of heat causes both global and local temperature variations in the design. Temperature variation affects the electron mobility and threshold voltage. At higher supply voltages and technology node points above around 65nm, the mobility effect tends to dominate, causing the circuit to slow (i.e., rising $t_P$) with increasing temperature. But at more advances fabrication nodes operating at lower supply voltages the effect on threshold voltage dominates over mobility and the speed increases with increasing temperature (called the “temperature inversion effect”) [41]. While this effect is important at more aggressive technology nodes, such as 28nm [43], the work reported in this thesis was performed using a 130nm process that exhibits a conventional temperature–delay characteristic at its typical supply range. Nonetheless, this effect emerges at lower supply voltages below 500mV even at 130nm. Simulations performed on a simple TH22 gate at its typical corner varying the temperature between -40°C to +65°C across the voltage range from 300mV to 1200mV (Fig. 2.1) clearly show this temperature dependency. It can be seen that the cross-over in this technology occurs at around 600–700mV, at which point the temperature–delay slope goes from positive (increasing delay) to negative (decreasing delay with temperature) and therefore delay is almost insensitive to temperature.
2.2 NCL and Timing Considerations

All the PVT variations outlined in the previous section, which affect the circuit level parameters (delay, power etc) need to be taken into account during the circuit design process. In the case of synchronous circuits, the clock could be said to be “imposed” on the circuit as it is typically derived externally and independently of the data flow of the design. Thus, by definition, any fixed clock period will be independent of the PVT-induced timing changes and hence the designer needs to build appropriate safety margins into the timing to allow for the full range of potential variation induced changes in the circuit parameters, even if they do not actually occur in practice. Often this is achieved by maintaining over-designed timing margins based on careful timing analyses.

Regardless, the success of a given design is dependent on the accuracy of the circuit models and extreme combinations of process corner and operating conditions may still cause the circuit to fail. CMOS circuits are routinely “binned”, or sorted into (clock) speed grades depending on parametric tests of the silicon behavior (particularly mobility), with lower maximum clock speeds implying lower selling price. The circuit is still sensitive to operating conditions, which may require additional (and expensive) cooling to ensure that the clock always operates with sufficient timing margins.

As mentioned earlier, non-DI techniques such as bundled delay methodologies can be difficult to design and may still require extensive timing analysis. In this respect, techniques such as bundled delay (for example), that require careful matching of data and handshaking path delays, are not much better than the synchronous case. In contrast, quasi-delay insensitive (QDI) asynchronous design styles such as NCL use the completion signal generated by the data flow to trigger the next signal event. Thus, variation
at any point in the data flow will simply result in the next event being advanced or delayed accordingly. Thus, very little timing analysis and minimal margins are required by these asynchronous design styles. The timing analysis that is required tends to be limited to design rules such as transition time, capacitance and load checks to ensure that the average case timing meets the requirements of the design (e.g., sampling period, throughput, latency etc). Since the timing information is implicitly generated from the data flow itself, NCL tends to be inherently robust to the variations described above.

The behaviour of NCL have been discussed in detail in many places (e.g., [5, 31]). Some of the more important of these characteristics, which have influenced the research described in the following chapters, can be summarized as follows:

- **Ease of Design:** NCL circuits are easy to design in the sense they can be fully expressed in hardware languages and compiled onto silicon just as for synchronous logic. Designers need not worry about timing issues and clock tree design since the data flow itself signals completion and readiness to accept new data.

- **Lower design cost and risk:** Since the global control signals (e.g., the clock) are removed, issues such as cross-chip clock skew and setup time control also disappear.

- **Potentially lower power consumption:** NCL systems operate in terms of synchronized wavefronts of monotonic level transitions and distributes the demand for power. Clock tree power and the power associated with spurious switching of the transistor is eliminated. However, the number of switching events per data transaction may rise, so this aspect of the design requires careful analysis.

- **Convenient technology migration:** Since NCL logic is delay insensitive, it will tend to be more easily portable from one technology to other.

- **Automatic adaption to physical properties:** Implicit timing signalling makes it more robust against PVT variations and different implementation environments. NCL circuits have been shown to work over a large range of operating conditions such as supply voltage and temperature.

- **Speed of operation:** NCL circuits do not need extensive timing margins as is the case with clocked circuits and operate with average-case propagation performance rather than worst case. As a result, they can operate at the full rate allowed by
the technology and the logic design. Fine grained pipelining is possible which can result in higher throughput rates than conventional clocked techniques, at the cost of additional power [44].

It is these final two issues that have primarily motivated this research, and in particular how to operate with adequate performance at low energy in NCL by running at or near the transistor threshold voltage.

2.3 Subthreshold NCL

The power–performance trade-offs of sub-threshold NCL logic were explored in [30] by comparing the sub-threshold behavior of a conventional multi-bit five-tap FIR filter in NCL to an equivalent clocked Boolean logic filter using a 65nm process. This NCL FIR filter was shown to exhibit an average power–delay product of around 4.56pJ at $25^\circ\text{C}$, $V_{DD} = 0.3V$. As the work was intended for military applications, robustness against process and environment variations was an important issue. By using constant coefficients to reduce the multiplier complexity, it was found that meeting the worst case performance corner resulted in a maximum clock of 9.15KHz. The equivalent NCL clock-less circuit could run at 11.4KHz at the same worst case corner, but up to 198KHz under nominal conditions. NCL was also shown to suffer significantly less variation in performance than the clocked Boolean logic (CBL) circuits, something which can translate to increased yield and reduced die costs. It was also observed that NCL offers some opportunities for novel data driven control of the supply voltage to allow the circuits to be “tuned” for optimum power vs. performance although this has to be weighed against the difficulty of optimizing transistor geometries (width & length) for both normal and sub-threshold operation.

The design and implementation of a specialized quasi-delay-insensitive style cell library is shown in [26] using a so-called PCSL technique that is able to exploit full dynamic voltage scaling. The library components have been verified using a conventional 8-tap, 8-bit FIR filter operating down to 130mV and their behavior compared to other sub-threshold NCL techniques. The PCSL technique appears to offer smaller area, faster operation (less delay) and lower power dissipation than comparable approaches that can work with full Dynamic Voltage Scaling, including at sub-threshold.
A general purpose dual rail NCL compatible 6th order low pass conventional multi-bit FIR filter was designed in [45], which can operate down to 170mV in the particular 130nm process used. As in [26], this work also illustrates the robustness of the NCL technique against process variation, in this case via data derived from Monte-Carlo simulations.

The sixth-order asynchronous FIR filter implemented in [2] was used to explore beta-ratio modulation for achieving process variability tolerance. Further, it was shown that the bit format of the filter coefficients can have a significant effect and can help reduce power and area by simplifying the structure of the multi-bit multipliers.

In [46], a conventional multi-bit FIR filter was designed to be operated over a supply range of 1V to 2.5V. A conventional Boolean 8-tap FIR filter was compared with an NCL 8-tap FIR filter in 90nm technology. In this case, the NCL design dissipates 20.8mW with 30.6nS of delay and an area of 1.52nm\(^2\) (≈ 72460 transistors). The key power/area figures from that work are summarized in Table 2.1 below.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Conventional CMOS Logic</th>
<th>NCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistor count</td>
<td>27954</td>
<td>72460</td>
</tr>
<tr>
<td>Power Consumption (mW)</td>
<td>26.8</td>
<td>20.8</td>
</tr>
<tr>
<td>Delay (nS)</td>
<td>32.6</td>
<td>30.6</td>
</tr>
<tr>
<td>Area (mm(^2))(^1)</td>
<td>0.587</td>
<td>1.52</td>
</tr>
<tr>
<td>Power Delay Product (Joule x 10(^{-9}))</td>
<td>0.88</td>
<td>0.64</td>
</tr>
</tbody>
</table>

# Table 2.1: Area/Delay figures from [46]

The design of asynchronous circuits for low power was addressed in [47] using an example of a FIR filter bank for a digital hearing aid. The asynchronous design re-implements an existing synchronous circuit which is used in a commercial product. For comparison, both designs have been fabricated in the same 700nm CMOS technology and operated down to 1.5V. The synchronous design contains around 48000 transistors and its power consumption is approximately 470\(\mu\)W. The asynchronous design contains approximately 70,000 transistors and its power consumption is 85\(\mu\)W of which 38% is consumed by the multiply/accumulate array (MAC). The work employs a four phase bundled delay protocol, slicing of the data and memory blocks and conditionally activating a number of slices, using a folded implementation of the filter coefficients and approximating them

\(^1\)The paper erroneously says nm\(^2\).
to contain at most three ones, simplifying the multiplier with shift and adder module etc. to save power and area in the MAC array.

A high-speed hardware efficient 41-tap, 15-bit word length FIR Compensation Filter has been designed in [48] as a component in a Delta-Sigma Modulator (DSM) Analog-to-Digital Converter (ADC). The filter targets high-throughput by pipelining its adders and multipliers. Efficient circuit-level techniques, namely customized adders, multipliers and D-flip-flops are used to further improve performance. The FIR filter is implemented in CMOS 130nm technology which contains approximately 180K transistors and occupies 2.69 mm$^2$ area. The filter is capable of operating at a maximum clock rate of 1.25 GHz. The multiplier at the heart of the compensation filter takes up about half of the area. To avoid wasting area and power for fixed filter coefficients, the multiplication is realized more efficiently with shift-and-add functions, by executing only those operations corresponding to the positions where the coefficient value is logic 1. This precludes its use in a situation where variable coefficients are required.

In [49], a FIR filter is implemented as a delay-insensitivity asynchronous circuit using the pipeline architecture of Multi-Threshold NULL Conventional Logic (MTNCL). The computing units with different pipeline stages and pattern delay shift registers are integrated as four FIR filter designs using an IBM 130nm process. A generic FIR filter is constructed based on the unsigned 8×8 multiplier and the generic adder and pattern delay shift register used to insert DATA and NULL cycles. The FIR filter has a fixed 8-bit input and a reconfigurable number of taps. As shown in Table 2.2, the paper reveals a conventional trade off between energy and delay using pipelining in this MTNCL system.

The ultra-low power Delayed Least Mean Square (DLMS) adaptive filter in [50] operates in the sub-threshold region for hearing aid applications. Sub-threshold operation has been accomplished by using a parallel architecture with pseudo NMOS logic style. The parallel architecture enables the system to run at a lower clock rate with a reduced supply voltage, while maintaining the same throughput. As illustrated in Table 2.3, pseudo NMOS logic operating in the sub-threshold region (called “sub-Pseudo NMOS” in the table) was shown to exhibit a better power-delay product than subthreshold CMOS logic (“Sub-CMOS”). Simulation results show that the system can process voice signals at a throughput of 22KHz with a supply voltage of 400mV and achieve 91%
Table 2.2: Area/Delay figures from [49]

<table>
<thead>
<tr>
<th>Parameters</th>
<th>FIR\textsubscript{NOPIPE}</th>
<th>FIR\textsubscript{FULLPIPE}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average delay (nS)</td>
<td>13.35</td>
<td>4.51</td>
</tr>
<tr>
<td>Energy (for 100 data) (pJ)</td>
<td>4675</td>
<td>11155</td>
</tr>
<tr>
<td>Gate Count</td>
<td>6831</td>
<td>15278</td>
</tr>
</tbody>
</table>

Table 2.3: FIR filter comparisons from [50]

<table>
<thead>
<tr>
<th>Implementation styles</th>
<th>Clock frequency</th>
<th>$V_{DD}$</th>
<th>Energy per Operation</th>
<th>no. of transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folded Standard CMOS</td>
<td>748KHz</td>
<td>650mV</td>
<td>19.1nJ</td>
<td>31121</td>
</tr>
<tr>
<td>Non-folded Sub-CMOS</td>
<td>22KHz</td>
<td>450mV</td>
<td>2.47nJ</td>
<td>110916</td>
</tr>
<tr>
<td>Non-folded Sub-Pseudo       NMOS</td>
<td>22KHz</td>
<td>400mV</td>
<td>1.77nJ</td>
<td>85764</td>
</tr>
</tbody>
</table>

improvement in energy compared to the non-parallel architecture using standard CMOS logic.

A relatively complex signed truncated multiplier is described in [51], which includes a transmission-gate shifter structure, transparent latch plus a power-efficient speculative delay line to reduce power and area in a FIR filter. The FIR filter circuit in [52] uses a Reduced Slack Pre-Charged Half Buffer (RSPCHB) QDI template, RSPCHB adder and multiplier to design the asynchronous filter. This eliminates the need for an enable signal from the control block compared to a pre-charged half buffer. Targeting the need for asynchronous design methodology, [53] proposes an asynchronous design methodology based on a new latch controller for data transfer from one pipeline stage to other. Their 32-tap FIR filter design is based on delay lines implemented using a 350nm CMOS technology.

2.4 SWL Filters

It is clear from the study of the literature that the Multiply–Accumulate (MAC) stage is one of the more important modules in digital filters and so its efficient design and implementation is an ongoing and active area of study. While much of the work to date has been carried out in the multi-bit domain, it also appears that short word length systems are emerging as the important organizations for filter implementation.
Table 2.4: Area-performance comparison of single-bit FIR vs. multi-bit filter: non-pipelined Mode.

<table>
<thead>
<tr>
<th>Device</th>
<th>Single-bit</th>
<th>Multi-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tern. Coeff</td>
<td>LUTs</td>
</tr>
<tr>
<td>Cyclone-III</td>
<td>512</td>
<td>4098 (3%)</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>15603(13%)</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>30894(26%)</td>
</tr>
<tr>
<td></td>
<td>8192</td>
<td>62747 (53%)</td>
</tr>
<tr>
<td>Stratix-III</td>
<td>512</td>
<td>3925 (1%)</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>14368(5%)</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>28499(11%)</td>
</tr>
<tr>
<td></td>
<td>8192</td>
<td>55927(21%)</td>
</tr>
</tbody>
</table>

As introduced in Section 1.4.4, SWL devices make use of ΣΔ domain representation of the bits (data or coefficient, or both) to make the word length smaller but use over-sampling to reduce resolution error. On the face of it, this appears to imply that SWL filters will always be very much larger than their multi-bit counterparts. However, in a direct comparison of area and performance between short-word filters and conventional techniques on FPGAs [54], it was demonstrated that a SWL filter could achieve up to 40% higher performance and a slightly smaller area than its conventional equivalent. In that study, both SWL and the equivalent multi-bit filters were implemented with various orders and pipeline stages on FPGA devices to compare power, performance and area numbers. It was shown that even though the number of taps increase significantly due to oversampling requirement, the increase in area is only 1-2% owing to its efficient mapping to the LUT organization of typical FPGA systems.

In [35] similar comparisons on FPGA between SWL and conventional filters showed that, for lower filter orders, the performance of SWL organizations can be consistently better with almost equivalent area compared to its multi-bit counterpart. Table 2.4, drawn from that work, shows that it is not until the order reaches 8192 (equivalent to 64 taps in the conventional case) that the area of the single bit filter significantly exceeds that of its multi-bit counterpart. The authors indicate even better area savings in case of ASIC implementation where removing the need for high performance multi-bit parallel multiplication can significantly reduce the overall complexity and area of the circuit.
2.5 Summary

Digital signal processing applications appear to be inherently synchronous due to their regular sampling rate requirement. However, it is clear from prior work in this domain that asynchronous designs can exploit low-level data dependencies in these applications that are not available to synchronous designs. Most of the work discussed above has focused on optimizing the multiply-accumulate modules as these play a major part in the power/area consumption of the filters. The multi-bit coefficient representation further adds to the multiplication complexity. Little or no prior work has been identified that specifically analyses the characteristics of NCL static threshold gates operating in sub- or near-threshold region in combination with Short Word Length organizations and this will be the main focus of the work described in the subsequent chapters of this thesis.

In summary, the complete signal path of a Σ∆ filter system (Fig. 2.2) comprises the Σ∆ acquisition, filtering and “re-quantization” stages, usually under some form of microprocessor control. The work in this thesis concentrates on just the central filter part, shown inside the red dashed area on Fig. 2.2. As highlighted in [47] the synchronous design and asynchronous design are not necessarily opposites but represent design alternatives. Both have advantages and disadvantages. The choice of one over the other depends largely on the application and context. The following chapters will describe the NCL threshold gate library development followed by the filter design and implementation finally ending with the discussion of the results obtained in this work.
Chapter 3

NCL Threshold Gate Library Design

This chapter describes the NCL Threshold gate design, layout implementation and library characterization process followed in this work. The set of basic threshold gates (equivalent to the standard cells used in Boolean logic) thus developed are then used to map the proposed filter onto a physical structure to measure its performance, power and area characteristics. After being characterized at a typical process corner, the library can be used to perform timing aware Place and Route of the proposed design.

An NCL gate library is an instantiated set of the 27 basic threshold gates that have been defined for 2-5 inputs and which exhibit hysteresis behavior [6]. The gate library has been designed using a 130nm planar CMOS process (IBM cmrf8sf) considering balanced propagation delay. Transistor geometries (width and length) were adjusted such that \( t_{PHL} \) (cell delay when output switches from high to low level) and \( t_{PLH} \) (low to high transition delay) vary no more than 20% with the supply and the switching threshold deviation was constrained to be 50% ±10% of the supply voltage of 300mV. The circuit behavior was then measured across a range of supply voltages between 1.2V and 300mV (i.e., near/sub-threshold). Balancing the threshold helps in maintaining the noise margins whereas balancing the propagation delays allows the design process to use average propagation delay values during the timing analysis stage.
Chapter 3. *NCL Threshold Gate Library Design*

The design process for this NCL threshold gate library involves schematic and layout design and timing characterization across a range of input slew and output load values, and finally the layout creation, as is described in the following sections.

### 3.1 Threshold gate design

As mentioned previously, NCL threshold gates can be implemented using one of the static, semi-static or dynamic styles. A fully static style was used in this work as, although it requires more transistors, it reduces the effort needed to size the transistors. Furthermore, as the gates are fully static they will work at any data throughput rate (effectively down to DC).

#### 3.1.1 Schematic Design

Transistor level schematics of the basic 27 state holding threshold gates were created using the Cadence Virtuoso Schematic Editor tool. As shown previously in Fig. 1.4, these static gates comprise four identifiable sections: pull-up or go-to-data; pull-down or go-to-null; hold-data and hold-null. The pull-up and hold-data sections are implemented using PMOS transistors whereas the pull-down and hold-null sections are implemented using NMOS transistors. The gate also has an output inverter driving the final output. Since hold-null and hold-data sections serve to hold the previous state, these are kept at minimum allowed size in 130nm IBM technology, which is 160nm for the width and 120nm for the length, thereby saving transistor area. In order to determine the geometries for go-to-data and go-to-null transistor sections, the gates are divided into three coarse groups:

- **TH1n**: These are similar to Boolean OR gates and so usual sizing techniques are employed i.e., making the PMOS network stronger (wider) as ‘n’ increases from 1 to 4.

- **THnn**: This group consists of the threshold gates where all the inputs need to be DATA or NULL to change the state. These gates have feedback and so the strategy used to size them is slightly different from above. The length of the PMOS is set first, setting the level of the propagation delay in an acceptable range. A figure of
∼40ns at 300mV was decided after examining roughly comparable figures available in the literature, which tend to fall in the range of 60ns–70ns for a NAND gate across 180nm and 130nm technology. The width of the PMOS network was then adjusted to balance the rise and fall propagation delay within the target limits. Finally, the output geometries and NMOS geometries were adjusted to maintain the switching threshold close to 50% of the supply voltage.

- THmn: This group consists of the threshold gates where m≠n. Parametrized cells were created for each of these gates and the transistor geometries were determined using parametric analysis. To decide the transistor geometries, various graphs were obtained using automated Ocean scripts running within the Cadence Analog Design Environment at 300mV. For example Fig. 3.1 shows $t_{PLH}$ and $t_{PHL}$, switching threshold ($V_m$) and the deviation of $V_m$ from the mid-point for the TH22 gate at a supply of 300mV. The PMOS transistor width is varied in this case to achieve acceptable delays and $V_m$ values. The resulting delays are within 2% of the average and threshold is within 3% of the target 10% limit.

![Figure 3.1: TH22 design process](image)

### 3.1.2 Performance Measurement

All the NCL threshold gates developed above were simulated across a range of voltages from 1.2V down to 300mV for various output load and input slew combinations. The simulations were performed using a custom Ocean script which calls the Spectre simulator. The performance was measured in terms of rise and fall propagation delays,
switching threshold voltage and average power consumption per switching. As an example of the type of optimizations arising from this analysis, it was found that TH44 gate was not giving the expected performance at lower voltages due to the large transistor stack (both PMOS and NMOS) that characterizes this gate. Partitioning the TH44 circuit into multiple TH22 gates solved this problem. Table 3.1 shows the results obtained post simulations for a TH22 gate.

As shown in the graph in Fig. 3.3(a), there is a steep increase in the propagation delay just after 300mV. This graph was obtained from the values printed by Ocean script in Table 3.1 for output load of 5fF and input slew of 100pS section. Also when the $I_{DS}$ vs. $V_{GS}$ curve was plotted for an N-channel MOSFET (see Fig. 1.1), the switching threshold ($V_m$, defined as the point on the voltage transfer curve where $V_{IN}=V_{OUT}$) was found to be in the range of 300mV–400mV. Thus, the minimum supply voltage was set to be 300mV for all simulation and testing purposes in this work. The propagation delay and $V_m$ were balanced at 300mV and the ‘% Deviation’ column in Table 3.1 represents the resulting position of the switching threshold, $V_m$, compared to its ideal value at $V_{DD}/2$.

It can be seen that in the majority of cases, $V_m$ is less than 6% away from its ideal, although at $V_{DD}=0.7V$, this increases to around 14%.

The performance of the proposed filter was also measured at 400mV i.e., near/just above the threshold point of the individual transistors. Note that there is slight ambiguity in the measurement performed by the tool while printing the data and plotting the data as seen from the graph in Fig. 3.1 and Table 3.1. For example, the switching threshold shown in the graph (in Fig. 3.1) is 154mV whereas the one printed in the table for same supply voltage of 300mV, output load of 5fF and input slew of 100pS, by the Ocean script is 153mV. But this slight variation is almost certainly to do with rounding errors, or the like, within the printing and plotting utilities and does not influence the findings or the resultant dimensions.

As was identified previously in Table 1.2 (Chapter 1 ), which shows the energy and delay behavior of a single NCL TH22 gate, the propagation delay increases significantly as the supply voltage is lowered (from 175pS at 1V to 29ns at 300mV). Portable/wearable applications typically need to operate at lower supply voltages for low power and longer battery life while still achieving their performance targets so that these higher delays
Table 3.1: TH22 Performance

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Supply</th>
<th>t_{PLH}</th>
<th>t_{PHL}</th>
<th>Average</th>
<th>V_m</th>
<th>% Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>slew (ps)</td>
<td>load (fF)</td>
<td>voltage (V)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>Delay (ns)</td>
<td>(mV)</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0.2</td>
<td>236</td>
<td>228</td>
<td>232</td>
<td>106</td>
<td>5.9</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0.3</td>
<td>29.2</td>
<td>28.9</td>
<td>29.1</td>
<td>153</td>
<td>1.9</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0.4</td>
<td>4.64</td>
<td>4.76</td>
<td>4.7</td>
<td>212</td>
<td>5.9</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0.5</td>
<td>1.28</td>
<td>1.28</td>
<td>1.28</td>
<td>245</td>
<td>-1.9</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0.7</td>
<td>0.37</td>
<td>0.35</td>
<td>0.36</td>
<td>301</td>
<td>13.9</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>1.0</td>
<td>0.18</td>
<td>0.17</td>
<td>0.17</td>
<td>510</td>
<td>1.9</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>1.2</td>
<td>0.14</td>
<td>0.13</td>
<td>0.13</td>
<td>564</td>
<td>5.9</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0.2</td>
<td>338</td>
<td>280</td>
<td>309</td>
<td>106</td>
<td>5.9</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0.3</td>
<td>34.6</td>
<td>35</td>
<td>38.3</td>
<td>153</td>
<td>1.9</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0.4</td>
<td>6.76</td>
<td>5.53</td>
<td>6.14</td>
<td>212</td>
<td>5.9</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0.5</td>
<td>1.82</td>
<td>1.51</td>
<td>1.66</td>
<td>245</td>
<td>1.9</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0.7</td>
<td>0.49</td>
<td>0.43</td>
<td>0.46</td>
<td>301</td>
<td>13.9</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1.0</td>
<td>0.24</td>
<td>0.22</td>
<td>0.22</td>
<td>509</td>
<td>1.9</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1.2</td>
<td>0.18</td>
<td>0.17</td>
<td>0.18</td>
<td>564</td>
<td>5.9</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>0.2</td>
<td>20200</td>
<td>9960</td>
<td>15100</td>
<td>106</td>
<td>5.9</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>0.3</td>
<td>25400</td>
<td>11000</td>
<td>18200</td>
<td>153</td>
<td>1.9</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>0.4</td>
<td>421</td>
<td>148</td>
<td>285</td>
<td>212</td>
<td>5.9</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>0.5</td>
<td>106</td>
<td>44</td>
<td>75.2</td>
<td>245</td>
<td>1.9</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>0.7</td>
<td>24.1</td>
<td>16.6</td>
<td>20.3</td>
<td>301</td>
<td>13.9</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>1.0</td>
<td>10.3</td>
<td>9.7</td>
<td>10</td>
<td>510</td>
<td>1.9</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>1.2</td>
<td>7.8</td>
<td>8.2</td>
<td>8</td>
<td>564</td>
<td>5.9</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>0.2</td>
<td>236</td>
<td>228</td>
<td>232</td>
<td>106</td>
<td>5.9</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>0.3</td>
<td>29.3</td>
<td>29</td>
<td>29.2</td>
<td>153</td>
<td>1.9</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>0.4</td>
<td>4.76</td>
<td>4.84</td>
<td>4.8</td>
<td>212</td>
<td>5.9</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>0.5</td>
<td>1.39</td>
<td>1.38</td>
<td>1.38</td>
<td>245</td>
<td>1.9</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>0.7</td>
<td>0.45</td>
<td>0.44</td>
<td>0.45</td>
<td>211</td>
<td>5.9</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>1.0</td>
<td>0.24</td>
<td>0.25</td>
<td>0.24</td>
<td>510</td>
<td>1.9</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>1.2</td>
<td>0.19</td>
<td>0.2</td>
<td>0.19</td>
<td>564</td>
<td>5.9</td>
</tr>
</tbody>
</table>

may need to be compensated at the architectural level. Techniques such as SWL structures can achieve very short critical data paths and tightly constrained local routing. Other NCL-related optimization techniques such as “bubble” and “wavefront population” circuits, as suggested in [5], are also used in this work to maintain high throughput.

Note that the average power is dominated by the switching power and so also depends on for how long the circuit is simulated. In this case, for every voltage, the simulation time is kept long enough for the output to settle. The graphs of Fig. 3.2 and Fig. 3.3 shows the static and dynamic power components for a TH22 gate. The propagation delay t_{PHL} and t_{PLH} and the average cell delay vs. supply voltage are also shown. In Fig. 3.3, the power and delay figures are given on a log scale to reinforce their exponential
relationship with voltage near the threshold point. As mentioned in the first chapter, the average power per transition is dominated by the dynamic power since the circuit is simulated to the point in time when the output settles for one switching event.

![Figure 3.2: TH22 gate delay and static power per transition](image)

![Figure 3.3: TH22 gate delay and dynamic power per transition](image)

### 3.1.3 Layout Creation

The layouts were created with optimized transistor placement and standard place and route tool compatibility measures. An example of a library component schematic and layout (TH22) is shown in Fig. 3.4. These threshold gates are used for mapping and placing the filter design into a (preliminary) physical structure later in this work to determine the overall physical area occupied by the design. The factors considered while creating the layout are:
• Basic Virtuoso Design Rule Checks (DRC) and Layout versus Schematic (LVS) rule checks;

• Place and Route tool compatibility factors such as standard cell height (2.8\textmu m), standard cell width (multiple of minimum cell width), power rails $V_{DD}$ and $V_{SS}$ widths, preferred metal routing directions, N-Well area etc. For example, as much as possible the layout followed a “HVH” style i.e., vertical poly, horizontal M1.

• Optimized transistor placement and overlapping wherever allowed so as to minimize the physical area.

In order to enable the foundry to manufacture the devices, every foundry decides on one tool to be the golden or industry standard tool which should be used to perform all the design rule checks and parasitic capacitance extraction. However, due to unavailability of the necessary resources, golden DRC/LVS checks and extraction could not be performed. All the layouts are basic Virtuoso DRC/LVS “clean” i.e., they pass all of the checks specified in the standard rules files that are specified by the 130nm PDK.

3.2 Timing characterization

Timing characterization is the process of obtaining the timing and capacitance data for a given standard cell (threshold gate in this work) across the range of input slew and
output loads. The characterized library was created to be used as part of a standard EDA (place and route) tool flow so that the timing aware placement and routing of the FIR filter designs can be performed after being mapped to these standard cells.

The Unified NCL Environment (UNCLE) tool was used to perform the library characterization for the NCL threshold gates developed. The library file thus obtained was converted to standard liberty format using a Python script developed for that purpose. The characterization process involved connecting the Design Under Test (DUT) to the input inverter driver with varying load and output capacitance values. The output capacitance was varied across a predetermined range (0.2fF to 204.8fF) and the input load capacitance was varied to achieve a range of input slew. This range was determined by the UNCLE scripts and was not altered. The characterization was done using Ultrasim called from inside the UNCLE python script. The testbench described above is shown in Fig. 3.5

![Figure 3.5: Library Characterization Testbench](image)

Similarly, the input capacitance for each input pin of each threshold gate was measured using Ultrasim cmeas (i.e., measure capacitance) command called inside a python script within the UNCLE tool. The characterized timing and capacitance values and the values obtained during design process correlate well considering the tool differences (Spectre for design versus Ultrasim for characterization).

A partial view of the library file thus obtained is shown in Fig. 3.6

Snapshots of some of the other threshold gate schematics, layout and performance data are given in Appendix A.
Summary

This chapter began with schematic entry followed by layout creation and cell characterization of the 27 cells forming a NCL gate library. It has been observed that cells behave differently with respect to propagation delays and switching threshold near the threshold voltage of the transistor components and it is difficult to get the optimum performance across the wide supply voltage range (1.2V down to 0.3v in this case). So the library was optimized to work at lower voltages (0.3V) and then simulated across the full supply voltage range permitted by the 130nm technology. The lower voltage limit was decided after observing current voltage characteristic curves for the selected technology and cell delay behaviors for typical NCL threshold cells.

A further important task encompassed finding the range within which physical parameters of the transistors (width and length) can be varied to achieve the performance targets. Increasing these parameters too much leads to excessive self-loading on the feedback paths from the output as well as increased physical size. Thus, trial simulations were performed to set these limits. Some of the gates (e.g. TH44) did not give the expected results due to large transistor stacks in the PMOS and NMOS stages. These gates were partitioned into combinations of smaller gates such as the TH22 and the layouts merged into a final cell. The set of threshold gates thus obtained and as described in this chapter was then used to synthesize the proposed FIR filter and to
measure the performance, area and power of the resulting physical design. The next chapter describes the filter implementation.
Chapter 4

SWL NCL Filter Design
Implementation and Verification

This chapter describes the design and implementation of the single-bit binary FIR NCL asynchronous filter proposed in this work. The basic implementation is described first which targets low sample-rate wearable applications such as physiological signal monitoring and the like. The optimized SWL-NCL filter implementation is then described to verify its applicability in the speech frequency range. The design is then verified by simulation. A test circuit is demonstrated for each implementation, which are then analysed for power, speed of operation and area characteristics.

4.1 Basic Single Bit NCL Sigma–Delta Domain Filter Design

A single bit representation of the Sigma–Delta domain filter coefficients and data samples has been chosen to implement the proposed FIR filter using NCL. The design process is divided into two main parts: the filter design in MATLAB® and the hardware implementation in Cadence®. After designing the filter in MATLAB with existing algorithms and obtaining the coefficients, the design was transferred to NELL for synthesis. The detail process followed is mentioned in the following subsections.
4.1.1 Filter Design

As a first step, a 16-tap low pass filter was developed with passband of 30Hz, a sampling rate of 80Hz and stop-band ripple of 67dB as shown in Fig. 4.1. A 16-tap filter response (Fig. 4.1a) was designed using the MATLAB® filter design tool. As described in [55], Sigma-Delta filters require over-sampling of the coefficients so that they can be represented in the delta sigma domain using short word length samples (e.g., single bit or dual/ternary representation). The filter impulse response (Fig. 4.1b) was then interpolated at the Over Sampling Ratio (OSR), here chosen to be 32. In turn, the number of over-sampled coefficients sets the number of delay elements in the filter, in this case 512 single-bit stages.

![Design Characteristics of 30Hz FIR filter](image)

**Figure 4.1:** Design Characteristics of 30Hz FIR filter

4.1.2 HDL Filter Description

As was seen previously in Chapter 1, Fig. 1.6 [55], the basic structure of a SWL FIR filter is more-or-less identical to its conventional counterpart and comprises two main sections: multiplication of the coefficient taps with the input samples followed by the addition of the partial products. The final result is a multi-bit value that needs to be further processed (out of the scope of this work) to remove the high frequency noise component introduced by the coarse quantization. For example, in [15] an area-efficient single-pole IIR re-modulator filter was proposed to both remove the noise and transform the data back into single-bit form.

The 512-stage single-bit FIR filter was first described in NELL and then mapped to the library of NCL cells previously described in Chapter 3. As mentioned above, this HDL has been optimized to generate efficient NCL circuits and handles many of the special
constructs required by the logic style. For example, it implicitly understands dual rail logic and warns the designer of any missing completion paths.

Fig. 4.2 shows a simplified fragment of the filter structure after transformation to NCL. The complete code for the filter description is given in Appendix B. It can be seen that the shift register-style implementation requires a sequence of two latches to hold a single bit i.e., for both its DATA and NULL values. NCL registers are formed from pairs of simple TH22 gates that regulate the data flow under the control of the completion feedback from their successor stages, which in this case incorporates the data registers and the Multiply–Accumulate (MAC) unit. Here, the bit-wise “multiplier” is a simple AND gate (shown by the multiplication symbol in Fig. 4.2) while the ADDER tree (shown by addition symbol in Fig. 4.2) comprises $\log_2 K$ stages of binary adders (where $K=\text{number of data registers}$), which accumulate the partial products into a multi-bit sum.

Data flowing through the MAC unit constitutes what is referred as “vertical flow” and data through the DATA delay line is referred to as “horizontal flow”. When the signal filterOut arrives at the output of OutputReg, the two paths represented by samplereqIn and pipeReqIn send requests for new DATA/NULL. These request signals flow backwards vertically and horizontally through the completion feedback gates mentioned above updating the request to the previous stages.

Thus, in case of the basic filter implementation, the time taken to produce a new DATA/NULL demand depends on how long the current wavefront takes to travel both vertically to the OutputReg and horizontally through all 512 register stages. Only when the “dataIn[0]” register shown in Fig. 4.2 updates itself will the previous NCL register be able to update its state and so down the line of registers.
As the behavior of the two delay lines in the filter (the data and coefficient lines) is different, their implementation requires slightly different approaches, as described below.

**Data Delay Line Implementation**

Since the single-bit delta-sigma domain data samples are flowing continuously in the data delay line at the rate determined by data completion in case of NCL (or sampling rate in case of an externally controlled filter), this piece of logic is implemented by using alternate DATA–NULL resettable TH22 gates as shown in Fig. 4.3. The request signal is inverted version of the acknowledgement signal generated by ORing (TH12 in this case) both the data rails. This technique has the advantage of starting the sampling process right after the reset without the need of generating special signal to indicate whether the pipeline is full. Thus the ‘data[0]’ and ‘data[1]’ in Fig. 4.2 are all set to DATA value of ‘zero’ at the beginning of the operation.

**Coefficient Line Implementation**

Since the single-bit delta-sigma domain coefficients (obtained from the process of oversampling described above in Section 4.1.1) need to stay in the pipeline once initialized, a different technique is employed for the implementation of the coefficient pipeline. By implementing the additional coefficient pipeline in the filter structure (which is not usually the case and coefficients are loaded in parallel), the design gives the additional feature for reconfigurable coefficients that can be loaded serially during initialization cycle with just one control signal “initFilter” that is disabled during normal operation.

*Figure 4.3: Data delay line alternate DATA and NULL register implementation*
The technique employed for the coefficient pipeline is a variation on an “auto-generate” circuit [31], more usually used to generate fixed constants in a NCL circuit. It operates as follows. At the start of initialization phase “initFilter” signal goes high indicating start of the initialization. The coefficient pipe starts to fill up with the alternating sequence of DATA and NULL values required by the NCL convention. By the time the Data Delay Line is full and the filter is ready for its first valid sample, the coefficient line is already holding the required coefficients. Subsequent DATA/NULL requests are controlled by the combined completion signal from the MAC unit, shown as the “feedbackRail” signal in Fig. 4.4 together with the two T22 gates.

Whenever coefficient DATA is requested (feedbackRail=’1’), the coefficient register value is gated through the T22 (AND) pair (“Finalcoeff”), while a NULL request will immediately return ’00’ (= NULL). This technique will only work in this case because the coefficients are constant. An alternative would have been to keep the coefficient registers toggling between their DATA and NULL values but the auto-generate technique saves area by avoiding this and saves power by reducing the number of switching events in the coefficient pipe. In all other scenarios where the input data toggles, the T22 gate cannot be used as it does not exhibit the required NCL hysteresis behavior that controls...
the NULL-DATA cycle.

4.1.3 NCL threshold gate mapping

The physical mapping of the filter implementation described above uses a fully static gate library described in the previous chapter in order to measure its performance in terms of speed of operation, power and area. There are a number of alternative implementation options available that can exhibit smaller area, including various pseudo-static forms [56] as well as the PCSL technique mentioned previously in Chapter 2. The results obtained in this work can therefore be considered to be a worse case in that other techniques may offer equivalent performance and are likely to occupy smaller area.

The preliminary layouts were created using standard place and route tool (Cadence Encounter) to physically implement the designs described previously in HDL.

![Figure 4.5: Basic filter operation with externally controlled 50μS data-to-data request rate](image)

4.1.4 Filter Performance

The filter function was verified by simulation across the target supply range from 300mV to 1V at typical corner and 27°C. The output waveforms are shown in Fig. 4.5 for a supply voltage of 400mV where the input data is set to $D_I=11000...$ and coefficients are all 1s. As a result, the output becomes $[\text{filterOut}_{19:0}] = 01010101010101010110$ in dual-rail convention or $\text{filterOut}_{9:0} = 000000001$ in binary for the first sampleReq, $[\text{filterOut}_{19:0}] = 01010101010101011001$ in dual-rail convention or $\text{filterOut}_{9:0} = 000000010$ in binary for second sampleReq. It then stays constant at that value.
till $D_1N$ travels the length of the delay line (i.e., for a remaining 510 cycles). The response time once the sampleReq asks for DATA is approximately $23\mu S$ (as shown with the time stamps on "sampleReqIn" and "filterOut1" in Fig. 4.5). This test case is better explained in a Table 4.1

<table>
<thead>
<tr>
<th>data cycle no.</th>
<th>Delay stage</th>
<th>Output binary</th>
<th>NCL dual rail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1 0 .... 2 0]</td>
<td>[0000000001]</td>
<td>[01010101010101010110]</td>
</tr>
<tr>
<td>2</td>
<td>[1 1 .... 0 0]</td>
<td>[0000000010]</td>
<td>[010101010101011001]</td>
</tr>
</tbody>
</table>

Coefficient $[511:0] =$ all set to 1 throughout the simulation

The critical path delay in this case depends on the time it takes for the acknowledge signal to traverse from the 512th delay element to 1st delay element, successively acknowledging and updating the values from right to the left in the chain of registers. The critical path delay, which limits the overall sample rate of the filter, increases from around $0.6\mu S$ at 1V to $23\mu S$ at 400mV to $120\mu S$ at 300mV. In the region below 400mV the delay becomes very sensitive to supply voltage, increasing by around $5 \times$ as supply falls from 400mV to 300mV. As a result, the absolute delay of the filter will become a complex function of $V_{DD}$, supply routing losses and noise, temperature etc. A QDI asynchronous approach such as NCL will simply track these changes to achieve an average-case performance over time. To determine the maximum rate at which filter can work, auto produce and auto consume cells [31] were interfaced with the filter and the acknowledgement signal generated from the filter data delay line was used to supply the samples.

The input vector sent to the data delay line is a sequence of alternating 1’s and 0’s while all of the coefficients are set to 1. This pattern was chosen to be a worse-case for the filter as it results in the maximum number of switching events during operation and therefore the highest dynamic power. Operation with real data and coefficients will tend to exhibit lower average power/energy. Overall, the filter can operate with a sampling rate up to 20KSPS (kilo-samples per second) at 400mV, as shown by the third plot in Fig. 4.6, and 4KSPS at 300mV, which is more than sufficient for the application domain envisaged here.

The filter dissipates around $1.6\mu W$ when operating at 5KSPS with this alternating data sequence, and $1.9\mu W$ at its maximum rate of 20KSPS. The design occupies a silicon area
of around 2.16\text{mm}^2 when preliminary place and route was performed on the synthesized netlist.

![Figure 4.6: Basic filter operation showing maximum frequency of operation with data request rate controlled by acknowledgement from the design.](image)

### 4.2 Performance optimized single bit NCL Sigma–Delta filter design

This section describes the implementation of an optimized single bit filter using a 32-tap FIR filter as an example.

#### 4.2.1 Filter Design and Implementation

Following a similar process described above for basic filter design, a low pass FIR filter was designed in Matlab® with a sample rate of 8KHz, a 4KHz cutoff frequency and stop-band ripple of 66dB, resulting in a filter with 32 taps and the normalized magnitude response shown in Fig. 4.7. The impulse response (Fig. 4.8) was then interpolated with an oversampling ratio of 32 and the coefficients quantized to single bit. The blue diamonds in the impulse response are the original 32 coefficients whereas red crosses represent the interpolated samples of the original impulse response. Thus, the number of filter stages becomes 1024.

The implementation and mapping followed the same steps as the basic filter, the main difference being that this filter contains additional so-called “bubble” stages (i.e., buffer...
or empty stages) to increase the throughput in the horizontal direction across the shift register. This performance optimization technique is described in the following subsection.

The single-bit FIR filter comprising 1024 data and coefficient stages was then described in NELL and the resulting logic mapped to the library of cells described in the previous chapter. Fig. 4.9 shows a part of the filter structure after mapping.
4.2.2 Performance Optimization

In addition to the shift register blocks that hold the single bit input sample values, additional buffer (empty) stages (called “bubbles” in [5]) were incorporated before each DATA and NULL stage to improve the performance of the delay chain. As suggested by [5], the throughput of the NCL pipeline can be increased if there are an optimum number of wavefront populations (DATA and NULL values) and bubble populations in the data-flow path. This is similar to adding pipeline stages in the synchronous case, although in the asynchronous case, the latency is virtually unaffected.

An increase in performance was achieved by adding 1024 empty stages in the shift register block that remained uninitialized to either a DATA or NULL wavefront. These stages serve as a copy of the data value which allows the acknowledgement signal to be generated faster thereby reducing the request cycle time. As a result, the critical path moves from being dependant on the total delay through the reverse handshaking lines, which is extremely long in these filters, to being limited only by the vertical data flow through the multiply–accumulate (MAC) stages, which has only around 10 stages before the final output.

Fig. 4.11 shows the first four snapshots of DATA–NULL cycles after the start of the operation. ‘D, N, B’ denotes the DATA, NULL and BUBBLE states whereas ‘d,n’ denotes the data request and null request respectively. The values in parenthesis are the initial values before any data is applied at the input. The delay line is tapped at stage number 4, 12, 20 and 28. When the DATA wavefront arrives at the input it will propagate through multiply and accumulate and delay chain but will wait at the output till the DATA is requested by output side even if input side provides next wavefront.
Similarly once the DATA is requested, after it is delivered, the ‘outputRegAck’ will stay in the null request state until the NULL wavefront is provided by the input side even if output side asks for new data. So it will continue to provide correct data even if part of the processing chain slows down.

Fig. 4.10 illustrates how NCL circuits themselves can control the speed of the requests for DATA/NULL depending on the state of the circuit. Here, the speed of operation of the circuit is slower initially as the filter is in initialization phase. Once all the coefficients are filled and data delay line is filled, the filter enters its full speed operation phase. This self-control behaviour ensures correct operation in case of widely varying conditions, in contrast to synchronous circuits that may propagate incorrect data values with little indication that the clock timing constraints are not being met. Thus, the self-timed behavior of NCL makes it inherently tolerant to variations in delay caused by process, voltage and temperature.

Another interesting feature that can be observed from Fig. 4.11 is how speed optimization is achieved by using additional pipeline stages. The operation of the feedback completion from MAC unit and data delay line is explained in section 4.1.2. The feedback to every tapped data delay line NCL register in this case of an optimized filter now comes from MAC unit and the bubble register. This bubble register has already copied the next wavefront from its previous stage and has already updated its feedback request signal to next wavefront (requesting NULL if the current wavefront is DATA and requesting DATA if current wavefront is NULL). So as soon as the MAC unit requests for
Figure 4.11: SWL Filter Implementation and Speed Optimization using “bubbles” - partial view

a new wavefront, the request to the NCL register updates itself as the other input to the completion feedback gate (TH22) coming from the bubble register is already updated.

For example, tap number 28 in the first snapshot holds the DATA wavefront ‘D’ and requests for NULL. But the “bubble” stage before it already has the copy of next wavefront which is supposed to be NULL in this snapshot and is already requesting the next DATA wavefront. So once the current state of DATA wavefront is consumed by the output and when it requests for a NULL wavefront, it is only a matter of one stage propagation in the horizontal direction i.e., from stage 26 to 28 and the NULL wavefront will be ready to be used. This makes the speed of operation independent of number of stages in the horizontal direction (which are determined by the oversampling ratio for SWL filters as mentioned in section 4.2.1). This does not happen in case of simple delay chain with only NULL-DATA values initialized. The speed of operation now only depends on vertical flow of data which is through the multiply/accumulate unit. This unit can be pipelined to increase the throughput further, as suggested in [49].

To demonstrate how the acknowledgment travels faster, thereby making the sample rate independent of the horizontal delay length to some extent, the basic 512 stage filter described in previous section was modified to include additional “bubble” stages and both filters are simulated to compare their performance in terms of speed of operation.
Fig. 4.6 and Fig. 4.12 shows the performance improvements achieved. The speed of operation is taken as the rate at which the acknowledgement signal (ko,8700 in case of optimized filter and ko,6654 in case of basic filter) requests DATA and its positive edge frequency is plotted in the graph. The frequency plot indicates the increase in speed from 23KHz for basic filter to 425KHz for the optimized filter measured with both the designs operating near the threshold voltage i.e., 400mV.

As can be seen from the summary chart in Fig. 4.13, the performance increase from 0.7KHz sample rate to 31KHz sample rate is achieved with the optimization of “bubble” and “wavefront” technique, but this comes at the cost of an additional 1024 register stages in the horizontal pipeline.

The sample rate in this work is measured as the difference between data-to-data request of the acknowledgement signal divided by the oversampling ratio of 32. The speed of operation increases from 425KHz to 1MHz in case of the optimized filter after the initialization time. The initialization time is the time during which all of the initial tasks such as filling up the coefficients, setting the reset for the data line etc. are performed.
4.2.3 Optimized Filter Performance

The average power, preliminary physical area and operating speed of the performance-optimized 1024-tap filter design were measured under the same conditions as the basic filter described above in Section 4.1.1. In this case, the filter was set up to give a low-pass characteristic with a cutoff around 8KHz.

Once initialized and the filter coefficients loaded, the single bit 1024 taps filter works with a data cycle-to-data cycle time delay of 1.25µS at 400mV. In other words, it can work with a maximum sampling rate of 25KHz, which is more than sufficient for the application domain envisaged in this work. For the very long shift-register structures in these filters, the addition of pipeline bubbles have a major effect, reducing the stage delay at 400mV from over 40µS to less than 1.25µS for the experimental filter. This of course has an impact on the area and the gate count has increased by \(\sim1.1 \times\) and it will also have an impact on power consumption. The design has been tested at two different PVT corners and its performance measured with a fixed sample period of 3.5µS (or around 285KHz/32 = 9KHz sample rate) and fixed supply of 0.4V. The results are described in Table 4.2, below.

```
<table>
<thead>
<tr>
<th>Corner</th>
<th>Temperature</th>
<th>Speed of Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>27</td>
<td>800KHz</td>
</tr>
<tr>
<td>SS</td>
<td>75</td>
<td>275KHz</td>
</tr>
</tbody>
</table>
```

The filter dissipates around 11µW on average at 400mV with a data-to-data request rate of 3.5µS at typical corner conditions. A preliminary place and route performed on the synthesized netlist resulted in a physical area of around 3.3mm². The worse-case delay in this case is measured between the rising edges of “sampleReqIn” and the slowest bit of the filterOut signal bus i.e., “filterOut_20”, and is approximately 15nS at 1V, rising to about 450nS at 400mV, as shown in Fig.4.14.
4.3 Verification and Debug

4.3.1 Verification

The system response was verified against the MATLAB filter response for a test input signal of 8Hz sinusoidal wave. The simulation time increases with an increase in signal frequency, so to keep the simulation time within reason, a low frequency sinusoid was chosen for verification against MATLAB. An 8-tap filter was implemented in SWL NCL and provided with delta sigma modulated single bit oversampled coefficients and test input vectors (i.e., the 8Hz sine wave) generated using MATLAB. The filter was then simulated using the Cadence Ultrasim environment at typical corner conditions with $V_{DD} = 1V$ over an amount of time sufficient to allow the outputs to settle. The output response was then converted from NCL into binary format and compared against the filtered signal response from MATLAB. The two plots shown below in Fig. 4.15 shows the correlation and confirms that the hardware is working as expected.

4.3.2 Debug

Although the development of the FIR filter was fairly straightforward due to its inherent regularity, there were two implementation issues that are worth mentioning here. Both are related to the organisation of these types of filter structures that comprise very long lines of registers with handshake feedback. Other types of filters as well as functions
such as LFSR (Linear Feedback Shift Register) units will face similar problems during synthesis.

The first arose from excessive gate load on the acknowledgement signal, which can cause excessive rise/fall times and result in slow performance, uncertainty in the switching times and problems with the handshaking. For example, in case of a 32-tap filter implementation with 4096 delay stages, the acknowledge signal from the output has to fan out to 5120 gates. Excessively slow rise times can result in an incorrect request for data. This issue is already well known in the synchronous domain and was observed in the simulations. It was resolved in the conventional manner by creating a buffer tree to divide the load on that one acknowledge line.

A further difficulty involves the way that the test bench is set up. In the current implementation, the testbench must ensure that pipeline is initialised (reset) before the coefficients are loaded. In this way, conflicts between the coefficient fill process are avoided and the correct output will result. This issue was resolved by creating a stimulus which initializes the design first and then removes the reset on data delay line to indicate start of filter operation. This is an artifact of simulation only and would not impact a real system.
4.4 Discussion and Comparison

This section discusses the results of the two filter designs described above. The filter designed in this work was simulated using Cadence® Ultrasim running within Virtuoso. The filter performance was estimated at two voltages, 1V and 0.4V, to allow closer comparison with existing designs from the literature. This discussion focuses only on the power and performance of the filter at typical corners as there are no comparable variability measurements available in literature.

Table 4.3 compares the area and performance of a number of filters from the literature, where data are available. Note that it is difficult to achieve a direct comparison due to the wide range of filter topologies, sample rates and fabrication processes used, so the comparison must be considered as indicative only. For example, the filter built for this work is a 32-tap implementation (i.e., 1024 stages after oversampling) and is designed to operate across a range of supply voltages down to 400mV. Filters reported previously are typically much smaller: in the range of 4 to 8 stages and will therefore exhibit much poorer roll-off and ripple characteristics. For this reason, Table 4.3 reports the area figures normalised to the filter order so that the limited overall area impact of the SWL technique can be seen more clearly.

Perhaps the most meaningful comparisons can be made with the work of [46] and [49] as these are NCL based filter designs, although the supply was fixed in both cases to around 1V so they are not operating in the sub-threshold region. The more complex MTNCL technique employed in [49] results in a more area-efficient filter design than the filter presented in this work. On the other hand, it can be seen that at 1V the optimised 32-tap SWL filter exhibits an almost half the critical path delay to the multi-bit filter of [46].

To achieve an approximate comparison with the power reported in [46], the figure derived for the SWL filter (1mW at 0.4V) was scaled up by $V_{DD}^2$ to an estimate of around 25mW at 2V, comparable with the ~21mW power figure reported in that work. However, it must be noted that this comparison is uncertain as the data rate at which the circuit was simulated to get this power number is not mentioned in [46] whereas the corresponding 8-tap filter design in this work was running at 20MHz.
### Table 4.3: Comparative Area-Performance-Power Results for Asynchronous Filters

<table>
<thead>
<tr>
<th>Ref</th>
<th>Supply (V)</th>
<th>Filter Order # taps</th>
<th>Tech node (nm)</th>
<th>Norm. Area (approx. transistors per order)</th>
<th>Path delay</th>
<th>Power or Energy</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>[49]</td>
<td>≈1</td>
<td>8</td>
<td>130</td>
<td>3415</td>
<td>$T_{dd}$:13.3nS</td>
<td>4678pJ per 100 data</td>
<td>following the maximum speed of the handshaking signals. So speed=75MHz</td>
</tr>
<tr>
<td>[46]</td>
<td>≈1.8-2</td>
<td>8</td>
<td>90</td>
<td>9057</td>
<td>30.6nS</td>
<td>20.8mW</td>
<td></td>
</tr>
<tr>
<td>[51]</td>
<td>1.1</td>
<td>8</td>
<td>350</td>
<td>-</td>
<td>1µS</td>
<td>5.86µW</td>
<td></td>
</tr>
<tr>
<td>[52]</td>
<td>2.5</td>
<td>4</td>
<td>180</td>
<td>28208</td>
<td>372nS</td>
<td>1.38W at 20MHz</td>
<td>filter can operate with a speed of 250MHz</td>
</tr>
<tr>
<td>[53]</td>
<td>3.3</td>
<td>32</td>
<td>350</td>
<td>1039</td>
<td>0.73µS</td>
<td>8nJ per computation</td>
<td></td>
</tr>
<tr>
<td>[30]</td>
<td>≈0.1-0.3</td>
<td>5</td>
<td>65</td>
<td>-</td>
<td>5µS</td>
<td>4.56pJ per operation</td>
<td></td>
</tr>
<tr>
<td>[45]</td>
<td>0.17</td>
<td>7</td>
<td>130</td>
<td>-</td>
<td>400µS</td>
<td>20.3nW 2.5mJ at 250mV</td>
<td></td>
</tr>
</tbody>
</table>

**This work**

| This work | 0.4       | 32       | 130 | 22136 | $T_{dd}$:1.25µS | - | 11.8µW | optimized |
|           | 0.4       | 16       | 130 | 22079 | $T_{dd}$:150nS | - | optimized |

| 0.4 | 16       | 130 | 19990 | $T_{dd}$:43µS | 1.6µW at 5KHz, 1.9uW at 20KHz | basic |
| 0.4 | 8        | 130 | 21900 | $T_{dd}$:40nS | 1mW at 20MHz | optimized |

**Synchronous**

| 0.4 | 32       | 180 | 4900 | - | 32.7µW at 286KHz |
| 0.4 | 16       | 180 | - | 8µW at 5KHz |

- indicates no data available, $T_{dd}$: DATA to DATA delay
The high speed asynchronous 4 tap FIR filter based on Reduced Slack Pre-Charged Half Buffer techniques in [52] can operate up to 250MHz. However it dissipates an extremely high 1.38W at 20MHz with 2.5V supply. In comparison, the SWL filter dissipates much lower power at 20MHz (estimated at approximately 40mW if scaled to 2.5V) and uses slightly fewer transistors. Note however, while the maximum operating speed achieved in [52] by employing RSPCHB technique is much greater than the SWL design, there is little point in comparing these power numbers at higher voltages as the point of this work it to achieve robust, low-power/energy design at low supply voltages.

In [45], an asynchronous filter was designed to operate within the sub-threshold supply range of 170mV–350mV, consuming 2.5nJ at its most energy efficient point between 250mV and 300mV. The maximum operating speed of the filter is around 100KHz at 350mV. In contrast, the 32-tap optimized SWL NCL filter designed in this work exhibits an average delay of 450nS at 400mV, resulting in a Power–Delay Product (PDP) of \( \sim 5.3\text{pJ} \) at this supply point. It is evident that the lack of complex multiplication stages in the SWL filter more than offsets the increased number of register stages and allows the SWL filter to achieve high operating speeds at low energy.

As a final comparison, two approximately equivalent *synchronous* SWL filters with 16 taps and 32 taps were designed to analyse the area/power impact at a specific operating rate, set by the maximum achievable clock period of 3.5\( \mu \)S in the case of the synchronous Boolean design, and therefore chosen to be 3.5\( \mu \)S (data to data) in case of NCL. Again, the lack of complex multiplication stages in the SWL filter more than offsets the increased number of stages and results in the asynchronous filter requiring 5× less power for 16 taps and 2.7× less power than the equivalent SWL clocked synchronous design for this 32-tap filter with identical throughput.

It has to be noted that the clocked synchronous filter was designed using an IBM cmrf7sf process (as opposed to cmrf8sf) due to the availability of a suitable standard cell library in that process. However, the average power numbers and energy per transition for a representative group of the cells were found to be within 5% when identically dimensioned cells were designed in both technologies and compared. Table 4.4 shows the average power and energy per transition for one TH22 gate to reinforce the point that the two technologies exhibit results that are similar enough to be considered interchangeable for this work.
Table 4.4: Average power and energy per transition comparison of TH22 for CMRF7SF and CMRF8SF IBM technology

<table>
<thead>
<tr>
<th></th>
<th>cmrf8sf (130nm)</th>
<th>cmrf7sf (180nm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1V</td>
<td>300mV</td>
</tr>
<tr>
<td><strong>$P_{AVG}$ (over 2nS)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>single transition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1-0) / (0-1)</td>
<td>4$\mu$W</td>
<td>842pW / 836pW</td>
</tr>
<tr>
<td><strong>Energy/transition</strong></td>
<td>8.3fJ</td>
<td>0.7fJ</td>
</tr>
<tr>
<td>for transistor Width=500n Length=180n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.5 Summary

This chapter has described the implementation of the proposed SWL NCL digital filter and its verification in terms of both functional correctness and performance. The filter implementation has been discussed in detail in terms of its various components such as its data delay line, coefficient line and feedback implementation technique. An area efficient architectural technique has been developed to implement the variable coefficient line. The basic filter design has been tested for its performance and has been found that the minimum order filter (i.e. 16 tap) needed by applications such as a heart waveform monitor, for example, runs comfortably at the rate of 4 kilo samples per second consuming approximately 5× less power than equivalent synchronous circuit at 400mV.

In order to expand the area of application further into the speech domain, an optimized filter implementation technique was developed and its performance evaluated across two process corners. The 32-tap filter developed runs at sampling period of 3.5$\mu$S with a supply voltage of 400mV. The design has undergone thorough timing checks to uncover implementation problems such as excessive loading effects and faulty test vector sequences. A buffer tree was created to resolve a loading issue on high fan-out nets that initially caused incorrect operation. Finally, like conventional design procedures, the test-bench has been separated into two parts: initialization and functional verification.

An equivalent synchronous design was created in an almost identical technology node to compare the proposed design technique with similar performance parameters. The proposed implementation is also compared with the other filter implementations available in the literature. It has been found that the lack of large multiplier structures in the SWL filter more than compensates for the larger number of stages necessary in these oversampled filter organizations.
Chapter 5

Summary, Conclusions and Future Work

5.1 Summary

This research described in this thesis was intended to determine whether combining the highly pipelined organization of Short Word Length filter systems with the robust operation of Null Convention Logic operating with low supply voltages (at or near the transistor threshold value), would support the development of low-power DSP systems.

The work was motivated by the simple idea that low power/energy operation can be achieved by running CMOS circuits at or near the threshold voltage of their transistor components. As the effects of process, voltage and temperature (PVT) variability are much more extreme in this operating region, asynchronous techniques in general, and Null Convention Logic in particular, have been used to provide a robust platform, largely immune to the effects of the PVT-induced problems.

Further, Short Word Length filter systems that operate on single bit ΣΔ encoded data have also been suggested as a suitable design technique for low-power systems, although it was not clear whether the requirement for significant oversampling rates and therefore a much larger number of stages would prevent the development of truly low power systems.
It has been shown that the large area overheads of the NCL asynchronous approach can be offset to some degree by the greatly reduced complexity of the short word length filter multiply–accumulate (MAC) architecture, while the use of NCL asynchronous design accommodates the increased variability arising from operation with a supply voltage that is near or sub-threshold.

To achieve this analysis, a single bit binary FIR ∑Δ digital filter has been implemented in a standard 130nm CMOS process using NULL Convention Logic. The filter was set up to achieve a sampling frequency target of 8KHz, suitable for speech and other low bandwidth applications over a range of supply voltages down to 400mV. The performance, area and power of the filter were measured and compared against an equivalent synchronous filter design using a similar process.

It was found that the average power at the typical process corner is consistently lower than for the equivalent synchronous case when measured across various filter orders. Finally, while the physical area occupied by the asynchronous filter components is always larger than the equivalent synchronous system, the overall structure is very regular with mostly local (adjacent) connectivity. As a result of this regularity and the low complexity of the interconnection network, the area overhead may be reduced in the place and route step.

5.2 Conclusions

In this work, a short word length digital filter has been presented that is suitable for low bandwidth applications. The range of applications can include physiological monitoring (EEG, ECG or muscle activity), speech filtering in hearing aids etc. Using the Null Convention Logic asynchronous techniques, the filter exhibits stable performance at a very low supply voltage range near the transistor threshold. Conventional synchronous Boolean techniques are unsuited to sub-threshold operation due to the need for large safety margins on the clock and uncertainty around setup and hold margins in the face of wide delay variability.

NCL, as a QDI asynchronous technique, can be set up to be “correct by construction” with few changes required to the circuit structure to allow to operate in this region. In the case of the standard cell libraries that were set up to support this work, these
changes were limited to the need to “tune” the physical dimensions of their transistors specifically to work in the near–threshold region, especially in the case of the gates with feedback.

The use of SWL techniques with NCL has been shown to result in significant power savings, notwithstanding the large number of filter stages required by this coarse quantization (i.e., down to one bit). Compared to an equivalent SWL filter in clocked synchronous mode, the NCL filter was shown to consume ~5× less power at an equivalent throughput, making it ideally suited to the envisaged low-power applications. Even prior to optimization, the sub-threshold filter was found to operate reliably up to a sample rate of about 20KSPS. The optimized NCL-based FIR filter is able to operate reliably down to the near threshold voltage region of 400mV with a sample rate of about 25KHz, indicating that, once it has been initialized (i.e., its coefficients loaded), the filter works comfortably within the required sampling rate of 8KHz at 400mV at both the typical and slow process corners. The average power-delay product of the 32-tap optimised design (at 400mV supply) was as low as 5.3pJ, illustrating that the approach is capable of achieving high performance at very low energy levels.

While the focus of this research was on the power/energy and performance of the SWL filter structures, it is possible to make a few general observations on the area of these circuits. On the face of it, the large number of stages required by these SWL filters implies very large area overheads. However, this has to be offset against the lack of the large, multi-bit multiplier structures that dominate the area of conventional filters. Even so, in the filters studied in this work, the area of the NCL circuits was significantly larger than their equivalent synchronous systems. This tends to come about for two main reasons. Firstly, the NCL standard cells are intrinsically large and secondly, mult-line (dual) signal paths are required to support the null convention, more-or-less doubling the number of required functional blocks. There is a potential reduction in component count due to the increased equivalent (Boolean) functionality of the NCL threshold logic, but in practice this contribution tends to be small. However, while no rigorous analysis was undertaken, it was observed during the place and route flow that the regular structure and short word representation clearly assisted in constraining the overall area of the filter. The interconnection nets are simpler, typically with low fanout to immediately adjacent cells. The repeated TH22 latch structures can be placed close to each other, minimizing interconnection length and thus the load parasitics. This contributed to
the ability of the optimized NCL FIR filter to operate to more than 20MHz, while maintaining low energy operation.

Based on the data presented and analysed in this thesis, it can be seen that the combination of asynchronous NCL techniques with the simple, locally connected and highly pipelined organization of SWL systems will support ultra-low-power DSP applications. In particular, digital filtering can be performed using SWL NCL operating over a wide range of voltages, down to the sub-threshold range, and across the full spread of process variability and temperature. Even at a very low supply voltage (400mV) the filters still met the sampling requirements for applications such as physiological data and speech processing.

5.3 Future Work

All the results in this thesis were obtained by assuming an external sample rate controlled by a notional clock, i.e., from a test-bench. In order to get the complete benefit of the SWL approach, the NCL filter scope would need to be widened to encompass the sampling process as well. While there have been many prior proposals for asynchronous Analog to Digital Converter (ADC) systems few, if any, have incorporated the entire processing chain. The ability to directly process (on demand) irregular and/or “bursty” signals may potentially result in extremely low power operation as they would operate (consume switching power) only on demand.

Noise margin is an important parameter for any CMOS circuit and especially for those operating at very low voltages, near and below threshold. As the supply reduced, so is the noise margin of the gates. There is already evidence to show that the monotonic signal cascade of NCL does not produce the same current spike behavior as its synchronous counterparts, with lower current peaks spread more widely in time. It will therefore tend to produce less intrinsic power supply noise and will require smaller values of bypass capacitance to significantly reduce power supply noise. While the characterisation of the noise margins of these NCL circuits was beyond the scope of the current research, it is important that it be fully explored before these techniques can be exploited in an industrial context.
Further, although area was also not a focus of this work, it is clear that the NCL approach does result in high area overheads. There are a number of alternative implementation options available that can exhibit smaller area, including various semi-static forms [56] as well as the PCSL technique mentioned previously. The results obtained in this work can therefore be considered to be a worse case in that other techniques may offer equivalent performance and are likely to occupy a smaller area.

For example, if the sample data delay line is implemented as a separate block of four registers (i.e., DATA-BUBBLE-NULL-BUBBLE) in a fully dynamic style, the transistor count can be substantially reduced as it will eliminate the transistors needed to build ‘hold DATA’ and ‘hold NULL’ states. The gate count can potentially more than halve using a dynamic style compared to full-static implementation. If the logic sense is allowed to invert at each stage, the output inverter can be removed from each of the TH22 registers while still maintaining the same final output (i.e., after each block of four). Of course, this has to be carefully analysed as the nodes are now dynamic and their charge leakage rates will matter greatly. Also, the feedback path needs to be designed carefully and will now comprise alternating NOR and NAND equivalent gates. As in all dynamic circuits, the hold periods will ultimately determine the minimum sustainable application throughput.

Ideally then, future work arising from this research would include considerations at both the circuit level and the systems level, by incorporating the sampling process into the scope of the signal processing chain and further optimising and characterizing the register circuits. Adding an asynchronous control processor such as described in [57] would result in a completely clock-less signal processing flow that would be well suited to the characteristics of low-power digital signal processing applications.
References


Appendix A:
NCL TH gates: Schematic, Layout and Performance

This section presents the schematics layouts and performance data (in terms of propagation delay) developed in this work for a set of NCL gates required by the proposed filter.

- T13

**Figure 1:** T13 Schematic and Layout
Figure 2: T13 Propagation Delay and Switching Threshold across supply voltage, input slew and output load

• TH33

Figure 3: TH33 Schematic and Layout
Figure 4: TH33 Propagation Delay and Switching Threshold across supply voltage, input slew and output load

• TH23

Figure 5: TH23 Schematic and Layout
Figure 6: TH23 Propagation Delay and Switching Threshold across supply voltage, input slew and output load
Appendix B: 
HDL code for the basic SWL NCL Filter

This section presents the NELL code for the proposed SWL NCL filter technique. NELL is a Verilog-like language that manages many of the key characteristics of NCL.

module reg1b_initNULL(output dual dout(null), output ko, input ki, input dual din);  
dout=ki&din;  
ko = dout;  
endmodule

module reg1b_init(output dual dout(0), output ko, input ki, input dual din);  
dout=ki&din;  
ko = dout;  
endmodule

module reg10b_initNULL(output dual dout[10](null), output ko, input ki, input dual din[10]);  
dout=ki&din;  
ko = dout;  
endmodule
module copy(output dual y, input dual x);
y=x; //copies the input to the output
endmodule

module copy10b(output dual y[10], input dual x[10]);
y=x; //copies the input to the output
endmodule

module booleanAND(output dual Z, input dual X, input dual Y);
Z/0=(X/0&Y/0)|(X/0&Y/1)|(X/1&Y/0);
Z/1=(X/1&Y/1);
endmodule

module halfAdder(output dual Sum, output dual Cout, input dual A, input dual B);
Sum/0=(A/0&B/0)|(A/1&B/1);
Sum/1=(A/0&B/1)|(A/1&B/0);
Cout/0=(A/0&B/0)|(A/0&B/1)|(A/1&B/0);
Cout/1=(A/1&B/1);
endmodule

module fullAdder(output dual Sum, output dual Cout, input dual A, input dual B, input dual Cin);
Sum/0 = (A/0&B/0&Cin/0)|(A/0&B/1&Cin/1)|(A/1&B/0&Cin/1)|(A/1&B/1&Cin/0);
Sum/1 = (A/0&B/0&Cin/1)|(A/0&B/1&Cin/0)|(A/1&B/0&Cin/0)|(A/1&B/1&Cin/1);
Cout/0= (A/0&B/0&Cin/0)|(A/0&B/0&Cin/1)|(A/0&B/1&Cin/0)|(A/1&B/0&Cin/0);
Cout/1= (A/0&B/1&Cin/1)|(A/1&B/0&Cin/1)|(A/1&B/1&Cin/0)|(A/1&B/1&Cin/1);
endmodule

module filter_512(output dual filterOut[10], output pipeOutAck, input sampleReqIn(null), input pipeReqIn(null), input dual Din, input dual coeff[512]);
dual dataIn[1023], dataOut[1023];
dual multiplierOut[512], outputRegOut[10];
dual addStage1Sum[256], addStage1Cout[256];
dual addStage2Sum[256], addStage2Cout[128], addStage2Carry[128];
dual addStage3Sum[192], addStage3Cout[64], addStage3Carry[128];
dual addStage4Sum[128], addStage4Cout[32], addStage4Carry[96];
dual addStage5Sum[80], addStage5Cout[16], addStage5Carry[64];
dual addStage6Sum[48], addStage6Cout[8], addStage6Carry[40];
dual addStage7Sum[28], addStage7Cout[4], addStage7Carry[24];
dual addStage8Sum[16], addStage8Cout[2], addStage8Carry[14];
dual addStage9Sum[9], addStage9Cout, addStage9Carry[8];
rail dataAck[1023], outputRegAck;
rail dataAckMod[512](null);

//delay line registration:: registers in horizontal direction

pipeOutAck=dataAck[0];

copy(dataIn[0],Din);
reg1b_InitNULL(dataOut[0],dataAck[0],dataAckMod[0],dataIn[0]);

for tapCounter=1:511 copy(dataIn[tapCounter*2-1],dataOut[tapCounter*2-2]);
reg1b_Init(dataOut[tapCounter*2-1],dataAck[tapCounter*2-1],dataAck[tapCounter*2-1],
dataIn[tapCounter*2-1]);
copy(dataIn[tapCounter*2],dataOut[tapCounter*2-1]);
reg1b_InitNULL(dataOut[tapCounter*2],dataAck[tapCounter*2],dataAckMod[tapCounter],
dataIn[tapCounter*2]);

//dataAckMmod generation:: depends on horizontal and verticle flow of data
for multAckCounter=0:510
dataAckMod[multAckCounter] = (dataAck[multAckCounter*2+1] & outputRegAck);

dataAckMod[511] = (pipeReqIn & outputRegAck);

// coeff and data multiplication
for andCounter=0:511
booleanAND(multiplierOut[andCounter] , coeff[andCounter] , dataOut[1022-2*andCounter] );

// stage1 addition :: 1bit half adder needed to add two consecutive regOuts. so total 256
1bit-halfadders to add 512 multiplication outputs

for stage1Counter=0:255
halfAdder(addStage1Sum[stage1Counter],addStage1Cout[stage1Counter],
multiplierOut[stage1Counter*2],multiplierOut[stage1Counter*2+1]);

// stage2 addition :: stage1 has generated 256 2bit numbers..so 128 2bit-ripple carry
adders are needed. LHS adder is HA and RHS adder is FA

for stage2Counter=0:127
halfAdder(addStage2Sum[stage2Counter*2+0],addStage2Carry[stage2Counter*2+1+0],
addStage1Sum[stage2Counter*2+0], addStage1Sum[stage2Counter*2+1]);
fullAdder(addStage2Sum[stage2Counter*2+1],addStage2Cout[stage2Counter],
addStage1Cout[stage2Counter*2+0],addStage1Cout[stage2Counter*2+1],
addStage2Carry[stage2Counter*1+0]);
// stage3 addition :: stage2 generates 128 3bit numbers..so 64 3bit-ripple carry adders needed. LHS adder of each is HA and rest two adders of each 3bit adder are FA

for stage3Counter=0:63
halfAdder(addStage3Sum[stage3Counter*3+0],addStage3Carry[stage3Counter*2+0],
addStage2Sum[stage3Counter* 4+0],addStage2Sum[stage3Counter* 4+2]);
fullAdder(addStage3Sum[stage3Counter*3+1],
addStage3Carry[stage3Counter*2+1],addStage2Sum[stage3Counter* 4+1],
addStage2Sum[stage3Counter* 4+3],addStage3Carry[stage3Counter*2+0]);
fullAdder(addStage3Sum[stage3Counter*3+2],addStage3Cout[stage3Counter],
addStage2Cout[stage3Counter*2+0],addStage2Cout[stage3Counter*2+1],
addStage3Carry[stage3Counter*2+1]);

// stage4 addition :: stage3 generates 64 4bit numbers..so 32 4bit-carry ripple adder needed. LHS is HA and rest 3 are FA

for stage4Counter=0:31
halfAdder(addStage4Sum[stage4Counter*4+0],addStage4Carry[stage4Counter*3+0],
addStage3Sum[stage4Counter* 6+0],addStage3Sum[stage4Counter* 6+3]);
fullAdder(addStage4Sum[stage4Counter*4+1],addStage4Carry[stage4Counter*3+1],
addStage3Sum[stage4Counter* 6+1],addStage3Sum[stage4Counter* 6+4],
addStage4Carry[stage4Counter*3+0]);
fullAdder(addStage4Sum[stage4Counter*4+2],addStage4Carry[stage4Counter*3+2],
addStage3Sum[stage4Counter* 6+2],addStage3Sum[stage4Counter* 6+5],
addStage4Carry[stage4Counter*3+1]);
fullAdder(addStage4Sum[stage4Counter*4+3],addStage4Cout[stage4Counter],
addStage3Cout[stage4Counter*2+0],addStage3Cout[stage4Counter*2+1],
addStage4Carry[stage4Counter*3+2]);
//stage5 addition :: stage4 generates 32 5bit numbers...so 16 5bit-carry ripple adder needed. LHS is HA and rest 4 are FA

for stage5Counter=0:15
halfAdder(addStage5Sum[stage5Counter*5+0],addStage5Carry[stage5Counter*4+0],
addStage4Sum[stage5Counter* 8+0],addStage4Sum[stage5Counter* 8+4]);
fullAdder(addStage5Sum[stage5Counter*5+1],addStage5Carry[stage5Counter*4+1],
addStage4Sum[stage5Counter* 8+1],addStage4Sum[stage5Counter* 8+5],
addStage5Carry[stage5Counter*4+0] );
fullAdder(addStage5Sum[stage5Counter*5+2],addStage5Carry[stage5Counter*4+2],
addStage4Sum[stage5Counter* 8+2],addStage4Sum[stage5Counter* 8+6],
addStage5Carry[stage5Counter*4+1] );
fullAdder(addStage5Sum[stage5Counter*5+3],addStage5Carry[stage5Counter*4+3],
addStage4Sum[stage5Counter* 8+3],addStage4Sum[stage5Counter* 8+7],
addStage5Carry[stage5Counter*4+2] );
fullAdder(addStage5Sum[stage5Counter*5+4],addStage5Cout[stage5Counter],
addStage4Cout[stage5Counter*2+0],addStage4Cout[stage5Counter*2+1],
addStage5Carry[stage5Counter*4+3]);

//stage6 addition :: stage5 generates 16 6bit numbers...so 8 6bit-carry ripple adders needed. LHS is HA and rest 5 are FAs

for stage6Counter=0:7
halfAdder(addStage6Sum[stage6Counter*6+0],addStage6Carry[stage6Counter*5+0],
addStage5Sum[stage6Counter*10+0],addStage5Sum[stage6Counter*10+5]);
fullAdder(addStage6Sum[stage6Counter*6+1],addStage6Carry[stage6Counter*5+1],
addStage5Sum[stage6Counter*10+1],addStage5Sum[stage6Counter*10+6],
addStage6Carry[stage6Counter*5+0] );
fullAdder(addStage6Sum[stage6Counter*6+2],addStage6Carry[stage6Counter*5+2],
addStage5Sum[stage6Counter*10+2],addStage5Sum[stage6Counter*10+7],
addStage6Carry[stage6Counter*5+1] );
fullAdder(addStage6Sum[stage6Counter*6+3],addStage6Carry[stage6Counter*5+3],
addStage5Sum[stage6Counter*10+3],addStage5Sum[stage6Counter*10+8],
addStage6Carry[stage6Counter*5+2] );
addStage5Sum[stage6Counter*10+3],addStage5Sum[stage6Counter*10+8],
addStage6Carry[stage6Counter*5+2] );
fullAdder(addStage6Sum[stage6Counter*6+4],addStage6Carry[stage6Counter*5+4],
addStage5Sum[stage6Counter*10+4],addStage5Sum[stage6Counter*10+9],
addStage6Carry[stage6Counter*5+3]);
fullAdder(addStage6Sum[stage6Counter*6+5],addStage6Cout[stage6Counter],
addStage5Cout[stage6Counter*2+0],addStage5Cout[stage6Counter*2+1],
addStage6Carry[stage6Counter*5+4]);

//stage7 addition :: stage6 generates 8 7bit numbers..so 4 7bit-carry ripple adders
//stage7Counter=0:3
for stage7Counter=0:3
halfAdder(addStage7Sum[stage7Counter*7+0],addStage7Carry[stage7Counter*6+0],
addStage6Sum[stage7Counter*12+0],addStage6Sum[stage7Counter*12+ 6]);
fullAdder(addStage7Sum[stage7Counter*7+1],addStage7Carry[stage7Counter*6+1],
addStage6Sum[stage7Counter*12+1],addStage6Sum[stage7Counter*12+ 7],
addStage7Carry[stage7Counter*6+0] );
fullAdder(addStage7Sum[stage7Counter*7+2],addStage7Carry[stage7Counter*6+2],
addStage6Sum[stage7Counter*12+2],addStage6Sum[stage7Counter*12+ 8],
addStage7Carry[stage7Counter*6+1] );
fullAdder(addStage7Sum[stage7Counter*7+3],addStage7Carry[stage7Counter*6+3],
addStage6Sum[stage7Counter*12+3],addStage6Sum[stage7Counter*12+ 9],
addStage7Carry[stage7Counter*6+2] );
fullAdder(addStage7Sum[stage7Counter*7+4],addStage7Carry[stage7Counter*6+4],
addStage6Sum[stage7Counter*12+4],addStage6Sum[stage7Counter*12+10],
addStage7Carry[stage7Counter*6+3]);
fullAdder(addStage7Sum[stage7Counter*7+5],addStage7Carry[stage7Counter*6+5],
addStage6Sum[stage7Counter*12+5],addStage6Sum[stage7Counter*12+11],
addStage7Carry[stage7Counter*6+4]);
fullAdder(addStage7Sum[stage7Counter*7+6],addStage7Cout[stage7Counter],
addStage6Cout[stage7Counter*2+0],addStage6Cout[stage7Counter*2+ 1],
addStage7Carry[stage7Counter*6+5]);

//stage8 addition :: stage7 generates 4 8bit numbers..so 2 8bit-carry ripple adders needed. LHS is HA and rest 7 are FAs

for stage8Counter=0:1
 halfAdder(addStage8Sum[stage8Counter*8+0],addStage8Carry[stage8Counter*7+0],
 addStage7Sum[stage8Counter*14+0],addStage7Sum[stage8Counter*14+ 7]);
 fullAdder(addStage8Sum[stage8Counter*8+1],addStage8Carry[stage8Counter*7+1],
 addStage7Sum[stage8Counter*14+1],addStage7Sum[stage8Counter*14+ 8],
 addStage8Carry[stage8Counter*7+0 ]);
 fullAdder(addStage8Sum[stage8Counter*8+2],addStage8Carry[stage8Counter*7+2],
 addStage7Sum[stage8Counter*14+2],addStage7Sum[stage8Counter*14+ 9],
 addStage8Carry[stage8Counter*7+1 ]);
 fullAdder(addStage8Sum[stage8Counter*8+3],addStage8Carry[stage8Counter*7+3],
 addStage7Sum[stage8Counter*14+3],addStage7Sum[stage8Counter*14+10],
 addStage8Carry[stage8Counter*7+2] );
 fullAdder(addStage8Sum[stage8Counter*8+4],addStage8Carry[stage8Counter*7+4],
 addStage7Sum[stage8Counter*14+4],addStage7Sum[stage8Counter*14+11],
 addStage8Carry[stage8Counter*7+3]);
 fullAdder(addStage8Sum[stage8Counter*8+5],addStage8Carry[stage8Counter*7+5],
 addStage7Sum[stage8Counter*14+5],addStage7Sum[stage8Counter*14+12],
 addStage8Carry[stage8Counter*7+4]);
 fullAdder(addStage8Sum[stage8Counter*8+6],addStage8Carry[stage8Counter*7+6],
 addStage7Sum[stage8Counter*14+6],addStage7Sum[stage8Counter*14+13],
 addStage8Carry[stage8Counter*7+5]);
 fullAdder(addStage8Sum[stage8Counter*8+7],addStage8Carry[stage8Counter],
 addStage7Cout[stage8Counter*2+0],addStage7Cout[stage8Counter*2+ 1],
 addStage8Carry[stage8Counter*7+6]);
//stage9 :: stage8 generates 2 9bit numbers..so 1 9bit-carry ripple adder needed. LHS is HA and rest 8 are FAs

halfAdder(addStage9Sum[0],addStage9Carry[0],addStage8Sum[0],addStage8Sum[8]);
fullAdder(addStage9Sum[1],addStage9Carry[1],addStage8Sum[1],
addStage8Sum[9],addStage9Carry[0]);
fullAdder(addStage9Sum[2],addStage9Carry[2],addStage8Sum[2],
addStage8Sum[10],addStage9Carry[1]);
fullAdder(addStage9Sum[3],addStage9Carry[3],addStage8Sum[3],
addStage8Sum[11],addStage9Carry[2]);
fullAdder(addStage9Sum[4],addStage9Carry[4],addStage8Sum[4],
addStage8Sum[12],addStage9Carry[3]);
fullAdder(addStage9Sum[5],addStage9Carry[5],addStage8Sum[5],
addStage8Sum[13],addStage9Carry[4]);
fullAdder(addStage9Sum[6],addStage9Carry[6],addStage8Sum[6],
addStage8Sum[14],addStage9Carry[5]);
fullAdder(addStage9Sum[7],addStage9Carry[7],addStage8Sum[7],
addStage8Sum[15],addStage9Carry[6]);
fullAdder(addStage9Sum[8],addStage9Cout, addStage8Cout[0],
addStage8Cout[1],addStage9Carry[7]);

// output register

dual outputRegIn[10] = addStage9Sum[0],addStage9Sum[1],addStage9Sum[2],addStage9Sum[3],
addStage9Sum[4],addStage9Sum[5],addStage9Sum[6],
addStage9Sum[7],addStage9Sum[8],addStage9Cout;
reg10b_initNULL(outputRegOut,outputRegAck,sampleReqIn,outputRegIn);
copy10b(filterOut,outputRegOut);

endmodule