Investigation and Optimization of Novel Stack Structures

Savan Pankajkumar Vachhani

PhD

University of York

Computer Science

December 2018

Simple living and high thinking – Gandhi

ABSTRACT

This thesis revolves around investigation and optimization of SSIA and will answer the following - **Can the performance of SSIA be significantly improved by the use of advanced design techniques, and will this make them more competitive with register file?** To answer this, a baseline performance model has been presented – and this was used to measure the enhancement of new SSIA models utilizing custom design approach (such as wider MOSFET v/s multi-finger MOSFET). The custom design has been one of the major challenges but using the custom implementation (SSIA_B) we have been able to reduce the power by 15% and area by 5% from standard cell implementation. Another custom implementation (SSIA_C) has been able to reduce the propagation delay by 37% at the cost of the higher area. A tool (SSIA Predictor + DARWIN) that can assist in the component selection of SSIA has been proposed. Using DARWIN the higher cost of the area of SSIA_C has been reduced. SSIA designs have been compared with the previous implementation by Bailey and Mullane and register files, SSIA is indeed shown to be very competitive with modern register file structures. In the final analysis, it is clearly demonstrated that significant improvements can be delivered by advanced VLSI techniques and that the performance and scalability of SSIA structures compared to register files is greatly improved. The novelty of this research lies in the thorough investigation and optimization of SSIA.

TABLE OF CONTENTS

Abstract iii

List of Figures ix

List of Tables xiii

Acknowledgements xviii

Declaration xx

CHAPTER 1 Introduction 1

1.1 Background 2

1.2 Recent Technology Trends 4

1.2.1 Miniaturization of technology: 5nm 5

1.2.2 Parallelism 5

1.2.3 Three Dimensional Integrated Circuits (3D ICs) 6

1.2.4 Stack Processors: Back to the future 7

1.2.5 A concluding remark 8

1.3 Introduction to Stack Architecture 8

1.4 Motivation and Rationale 12

1.4.1 Advantages 13

1.4.2 Clear lack of research 14

1.4.3 Promising results from previous research 14

1.4.4 Research Question 15

1.5 Thesis Structure 16

1.6 Summary 19

CHAPTER 2 Literature Review 20
2.1 Stack Processors 20
2.1.1 Stack Processors: First Generation 21
2.1.2 Stack Processors: Second Generation 25
2.1.3 Stack Processors: 21st Century 28
2.1.4 Stack Processors at York 30
2.1.5 Stack Processors at York: Superscalar Stack Issue Array (SSIA) 32
2.2 VLSI Optimization 36
2.2.1 Introduction 36
2.2.2 VLSI Optimization Techniques 37
2.2.3 Summary 40
2.3 Summary 40
CHAPTER 3 Standard Cell Implementation of SSIA 42
3.1 Introduction 42
3.2 Experimental Setup and Methodology 43
3.2.1 Fundamental Components 44
3.2.2 Multiplexer 54
3.2.3 Design Decisions 58
3.3 Fundamental Components 60
3.3.1 Area 61
3.3.2 Delay 63
3.3.3 Capacitance 65
3.3.4 Power 68
3.4 Multiplexer 70
3.4.1 Area 71

3.4.2 Delay 72
3.4.3 Alternative methodology - for multiplexer simulations 73
3.5 Superscalar Stack Issue Array (SSIA) 77
3.5.1 Area 78
3.5.2 Delay 79
3.6 Summary 82
CHAPTER 4 Custom Cell Implementation of SSIA 84
4.1 Introduction 84
4.2 Transistor Property Variation 85
4.3 Fundamental Components 89
4.3.1 Area 89
4.3.2 Delay 91
4.3.3 Capacitance 94
4.3.4 Power 97
4.4 Multiplexer 99
4.4.1 Area 100
4.4.2 Delay 100
4.5 Cost-benefit analysis 100
4.6 2g and 3g Implementation for f2 102
4.7 Equal sized NMOS and PMOS - equalNP 103
4.8 Hybrid Multiplexers 106
4.9 Comparison of different multiplexer implementations 108
4.10 System-level customisation 109
4.11 Summary 110
CHAPTER 5 Investigation of SSIA 112

5.1 Introduction 112
5.2 Methodology 113
5.2.1 Area 115
5.2.2 Delay 115
5.2.3 Power/Energy Consumption 118
5.2.4 Performance Parameters of an SSIA Version 124
5.3 Manual Selection for SSIA 124
5.3.1 Area 124
5.3.2 Delay 126
5.3.3 Power 129
5.3.4 SSIA Predictor 130
5.4 Automated selection for SSIA using DARWIN 133
5.4.1 Section DARWIN 133
5.4.2 Comparison with manual selections 136
5.4.3 Analysis of DARWIN Recommendations 140
5.4.4 Summary 143
5.5 Comparison with Previous SSIA implementations 144
5.6 Summary 150
CHAPTER 6 Evaluating Future OpportunitIes for Advanced VLSI Design Techniques 151
6.1 Dynamic-logic 151
6.2 Power gating 154
6.3 MOS Current Mode Logic (MCML) Implementation 158
6.4 Summary 159
CHAPTER 7 Conclusion and Future Work 161
7.1 A Reminder of the Research Question 161

7.2 Summary of Contributions 162
7.2.1 Develop a Baseline Performance Model 162
7.2.2 Evaluate the custom design approach 163
7.2.3 Compare the SSIA Structures 163
7.2.4 SSIA versus Register File Models 164
7.2.5 Advanced VLSI Design Techniques 164
7.3 Discussion of Contributions 165
7.3.1 Develop a Baseline Performance Model 165
7.3.2 Evaluate Custom Design Approach 165
7.3.3 Compare the SSIA Structures 166
7.3.4 SSIA versus Register File Models 166
7.3.5 Advanced VLSI Design Techniques 167
7.4 Opportunities for Future Work and Refinements 167
7.5 Concluding Remarks 168
APPENDIX A Dataset i
Abbreviations ii
Bibliography vi

LIST OF FIGURES

Figure 1-1: 1965 estimation by Gordon Moore about the number of components(transistor) per integrated function (IC chip) for the next ten years. [9]. 3

Figure 1-2: Number of transistors(000), clock speed (MHz), Power (W), and Performance/Clock (ILP) for Intel CPUs from 1970 to 2000. [3]. 4

Figure 1-3: Four common types of instruction set architecture based on the storage location of their operand [15]. 9

Figure 1-4: A block diagram for the architecture of a generic register machine [16]. 11

Figure 1-5: A block diagram for the architecture of a generic stack machine. [17]. 12

Figure 2-1: Two stills from BBC Micro episode about Transputer which are depicting parallel processing capability [33]. 27

Figure 2-2: Possible number of instruction-per-clock (IPC) in the stack code. [8]. 32

Figure 2-3: Diagram of 4-issue wide and 4-elements deep Superscalar Stack Issue Array (SSIA). 33

Figure 2-4: Propagation delay of various multiplexer implementations for issue widths 1 to 4 [1]. 34

Figure 2-5: FO4 delay data for various register files and SSIA by Bailey and Mullane with respect to the number of Read+write ports [1]. 35

Figure 3-1: Fundamental Components (a) Inverter schematic (b) Tristate buffer schematic (c) Inverter symbol (d) Tristate buffer symbol. 44

Figure 3-2: Layout and symbol of a transistor side-by-side. 45

Figure 3-3: Layouts of fundamental components (Left layout – inverter; Right layout – tristate buffer) of SSIA (with rulers). 46

Figure 3-4: A typical test bench which has an inverter as UUT and has a fanout of four. 47

Figure 3-5: Output waveforms of an inverter when supplied with a square wave signal as input. 48

Figure 3-6: Superimposed input and output waveform of an inverter showing propagation delay. 49

Figure 3-7: Equivalent Resistance –Capacitance model for an inverter. 50

Figure 3-8: The capacitance calculation for two connected inverters based on the equivalent model in Figure 3-7. 51

Figure 3-9: Screenshot of capacitance calculation using Cadence Virtuoso [64]. 52

Figure 3-10: Screenshot of power calculation using Cadence Virtuoso [64]. 53

Figure 3-11: (a) Schematic for 2g (b) Symbol of 2X1 multiplexer 55

Figure 3-12: A typical test bench which has 2X1 multiplexer as UUT and has a fanout of 4. 55

Figure 3-13: A simple box diagram if the layout of the 2X1 multiplexer is to be produced. 57

Figure 3-14: Picture showing the layout of two distinct transistors and their layout when abutted. 61

Figure 3-15: Test benches for fanout of 1, 4 and 8 commonly used in experiments. 63

Figure 3-16: Graph showing the trend of propagation delay across various fanouts and schematic, layout for fundamental components. 65

Figure 3-17: Diagram showing a transistor level schematic of the test benches to explain their capacitances at nodes. 68

Figure 3-18: 3X1, 4X1 and 5X1 Multiplexer Schematics and Symbols (Left to right). . 71

Figure 3-19: 2g (ganged component) (a) schematic (b) symbol. 74

Figure 3-20: : 3g (ganged component) (a) schematic (b) symbol. 74

Figure 3-21: Layout of 2g(top) and 3g(bottom) ganged components. 75

Figure 3-22: Diagram of 4-issue wide and 4-elements deep Superscalar Stack Issue Array (SSIA). 78

Figure 4-1: Screenshot of properties(such as width, length, numbers of fingers etc.) of n-channel MOSFET from Cadence. 86

Figure 4-2: Layout of Inverter (a)FW = 1, and (b) FW = 3. 87

Figure 4-3: Transistor (n-channel) layouts (a) Normal (b) Multi-finger (c) Wider. 88

Figure 4-4: Layout of Tri-state Buffer (a)FW = 1, and (b) FW = 4. 88

Figure 4-5: Graph showing the trend of increment in areas of wider and multi-finger inverters for each finger width/multiplier. 90

Figure 4-6: Graphs showing the comparison for wider and multi-finger implementations of inverter and tri-state buffers. 93

Figure 4-7: Diagram showing the change of capacitance on changing finger width for tri-state buffer with a fanout of four. 94

Figure 4-8: Layout of (a) inverter, and (b) inverter_equalNP. 104

Figure 4-9: Schematic of hybrid multiplexers (a) 3X1_hybrid, (b)4X1_hybrid, and (c)5X1_hybrid. 107

Figure 5-1: SSIA_A version of SSIA (depth = 8). 114

Figure 5-2: Delay dependency chart. 116

Figure 5-3: Static Power Consumption of Multiplexers. 121

Figure 5-4: Cadence screenshot of power/energy calculations of the tri-state buffer. .. 122

Figure 5-5: Simple flow chart of SSIA Predictor. 130

Figure 5-6: Simple flow chart of SSIA Predictor and DARWIN. 133

Figure 5-7: Screenshot of DARWIN (State - Reset). 136

Figure 5-8: Screenshot of DARWIN (State – Evolved for Area). 136

Figure 5-9: Graph of SSIA_A evolution for area optimization. 137

Figure 5-10: Graph of SSIA_A evolution for delay/speed optimization. 138

Figure 5-11: Graph of SSIA_C evolution for area optimization. 140

Figure 5-12: Graph of SSIA_F evolution for better area and speed. 142

Figure 5-13: SSIA and register file comparison by Bailey & Mullane (2014). 146

Figure 5-14: SSIA_A superimposed over Bailey & Mullane comparison graph. 147

Figure 5-15: Comparison graph of SSIA_A, SSIA_H and register file. 149

Figure 5-16: Comparison graph of SSIA_H and register files. 149

Figure 6-1: Dynamic-logic schematic, waveform and an example circuit [72]. 152

Figure 6-2: Schematics of tristate buffers (a) Standard tristate buffer, (b) 3T tristate buffer and (c) 4T tristate buffer. 153

Figure 6-3: Power gating diagram for a 3X1 multiplexer showing header switch and footer switch..... 155

Figure 6-4: Graphs showing propagation delay getting closer to normal with an increase in the size of a powergated transistor for multiplexers. 156

Figure 6-5: Graphs showing GND bounce and VDD bounce getting closer to normal with an increase in the size of a powergated transistor for multiplexers. 157

Figure 6-6: Graphs for 5X1 multiplexers for the investigation related to the impact of header switch and footer switch (Quadrant 1)propagation delay, (Quadrant 3)VDD swing and (Quadrant 4)GND swing. 158

LIST OF TABLES

Table 2-1: Prefix and postfix mathematical notation for expression [2*(3+3)/3]. 21

Table 3-1: Length, width and area of fundamental components inverter and tristate buffer. 48

Table 3-2: Inverter delay data. 50

Table 3-3: Fixed and variable, input and output capacitance of schematic and layout of inverter in femtofarad. 52

Table 3-4: Static and peak dynamic power consumption of inverter in nW. 54

Table 3-5: Length, width and area of a 2X1 multiplexer. 57

Table 3-6: Propagation delay data for the 2X1 multiplexer which has a fanout of four. 58

Table 3-7: Length, width and area of fundamental components and abutted tristate buffer. 62

Table 3-8: Schematic and layout propagation delay data of an inverter for fanout ranging from 1 to 8. 64

Table 3-9: Schematic and layout propagation delay data of a tri-state buffer for fanout ranging from 1 to 8. 64

Table 3-10: Fixed and variable, input and output capacitance of schematic and layout of inverter for fanout ranting from 0 to 8 in femtofarad. 66

Table 3-11: Fixed and variable, input and output capacitance of schematic and layout of tristate buffer for fanout ranting from 0 to 8 in femtofarad. 66

Table 3-12: Peak dynamic and static power consumption of an inverter in nanowatts for low to high and high to low input transitions. 69

Table 3-13: Peak dynamic and static power consumption of a tristate buffer in nanowatts for low to high and high to low input transitions. 70

Table 3-14: Calculated length, width and area of multiplexers of SSIA. 72

Table 3-15: Propagation delay of 3X1, 4X1 and 5X1 of SSIA for relevant fanouts in picoseconds. 72

Table 3-16: Calculated length, width and area of ganged components and their respective multiplexers. 76

Table 3-17: Propagation delay of 2X1 and 2X1g for fanouts 1,4 and 7 in picoseconds. 76

Table 3-18: Length, width and area of multiplexers designed using ganged components with a column for normal values for comparison. 77

Table 3-19: Propagation delay of ganged 3X1, 4X1 and 5X1 of SSIA for relevant fanouts in picoseconds with columns for normal and the difference. 77

Table 3-20: Area of SSIA and ganged SSIA for depths 4 to 8 of the stack. 79

Table 3-21: Individual delay of the component (multiplexer) of SSIA. 80

Table 3-22: Individual delay of the component (ganged multiplexer) of SSIA. 80

Table 3-23: Cumulative delay of the component (multiplexer) of SSIA. 81

Table 3-24: Cumulative delay of the component (ganged multiplexer) of SSIA. 82

Table 4-1: Area of fundamental components for various finger widths in um^2. 90

Table 4-2: Areas of wider and multi-finger inverters for comparison. 90

Table 4-3: Propagation delay of wider and multi-finger inverters for fanout ranging from 1 to 8 (schematic simulations). 91

Table 4-4: Propagation delay of wider and multi-finger tri-state buffers for fanout ranging from 1 to 8 (schematic simulations). 92

Table 4-5: Propagation delay of inverter and tri-state buffer for finger widths 1 to 4 and fanout ranging from 1 to 8 (layout simulations). 93

Table 4-6: Total input and output capacitance of inverter of finger widths 1 to 4 for fanout ranging from 0 to 8 in femtofarad. 95

Table 4-7: Total input and output capacitance of a tri-state buffer of finger widths 1 to 4 for fanout ranging from 0 to 8 in femtofarad. 96

Table 4-8: Peak dynamic and static power consumption of an inverter of finger width 1 to 4 for fanout ranging from 1 to 8 in nanowatts for low to high and high to low input transitions. 97

Table 4-9: Peak dynamic and static power consumption of a tri-state buffer of finger width 1 to 4 for fanout ranging from 1 to 8 in nanowatts for low to high and high to low input transitions. 98

Table 4-10: Calculated area of 3X1, 4X1 and 5X1 of finger widths 1 to 4 in um^2. 99

Table 4-11: Propagation delay of multiplexers (of finger widths 1 to 4) of SSIA for relevant fanouts in picoseconds in picoSeconds. 99

Table 4-12: Delay comparison relative to finger width 1 for fundamental components. 101

Table 4-13: Delay comparison relative to finger width 1 for multiplexers. 102

Table 4-14: Length, width and area of ganged components and ganged multiplexer of finger width 2 in um^2. 103

Table 4-15: Propagation delay in picoseconds of finger width 2 ganged 3X1, 4X1 and 5X1 of SSIA for relevant fanouts in picoseconds with columns for normal and the difference. 103

Table 4-16: Area of equalNP and normal components. 105

Table 4-17: Propagation delay data of 3X1 Multiplexer made using Custom Fundamental Components. 106

Table 4-18: Area of normal and hybrid multiplexers. 107

Table 4-19: Propagation delay data of hybrid multiplexers. 108

Table 4-20: Multiplexers and their possible versions. 109

Table 5-1: Configuration of SSIA_A (standard cell version). 114

Table 5-2: Cumulative Delay of SSIA_A Components. 117

Table 5-3: Static power consumption data of fundamental components. 120

Table 5-4: Static power consumption data of multiplexers. 120

Table 5-5: Energy consumption of inverter for various fanouts in femtojoules. 122

Table 5-6: Energy consumption of tri-state buffer for various fanouts in femtojoules. 123

Table 5-7: Energy consumption of various multiplexers. 123

Table 5-8: Performance parameters of SSIA_A (the standard cell version). 124

Table 5-9: Different versions of 3X1 multiplexers and their area. 125

Table 5-10: Configuration of SSIA_B (lowest area version). 125

Table 5-11: Performance parameters of SSIA_B (lowest area version). 126

Table 5-12: Cumulative delay of SSIA version made up of finger width 3 MOSFETs. 127

Table 5-13: Cumulative delay of SSIA_C components. 128

Table 5-14: Configuration of SSIA_C (lowest delay version). 128

Table 5-15: Performance parameters of SSIA_C (lowest delay version). 129

Table 5-16: Lookup Table 132

Table 5-17: Performance parameters of SSIA_A (the standard cell versions). 135

Table 5-18: Performance parameters of SSIA_B (lowest area version). 137

Table 5-19: Configuration of SSIA_D (A DARWIN Recommendation) 138

Table 5-20: Performance parameters of SSIA_D. 139

Table 5-21: Configuration of SSIA_F (A DARWIN Recommendation). 141

Table 5-22: Performance parameters of SSIA_F and SSIA_C. 141

Table 5-23: Configuration of SSIA_G (A DARWIN Recommendation) 142

Table 5-24: Performance parameters of SSIA_G. 143

Table 5-25: Cycle time and Access time for SSIA designed in UMC 90nm 145

Table 5-26: Cycle time and Access time for SSIA_A. 147

Table 5-27: Cycle time and Access time for SSIA_H. 148

Table 6-1: Rise time and fall time propagation delay of various kinds of tri-state buffers. 154

Table 6-2: Propagation delay of multiplexers in picoseconds for powergated transistor of finger widths 1, 3, 6 and 9. 156

Table 6-3: Speculated propagation delay data for the MCML multiplexer. 159

Table 6-4: Prediction for SSIA based on the speculation shown in table 6-3. 159

Table A-1: Propagation delay data of old and new implementation of SSIA i

ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude to my supervisor Dr Christopher Crispin-Bailey for his guidance, ideas, and the support throughout my studies. I thoroughly enjoyed having a discussion with him, even though it got intense at times. Dr Crispin-Bailey's expertise in the field and moral support has been invaluable, and I am especially grateful for his help in arranging the funding for PhD. This PhD studies happened because Dr Crispin-Bailey accepted my request for a summer internship way back in 2012 (made through a cold email). Thanks also go to members of my Thesis Advisory Panel (Dr James Walker, Dr Jim Austin and Dr Leandro Soares Indrusiak) for their suggestions, feedback and questions.

The funding for PhD came from various places – Department of Computer Science at York, Bestway Foundation, Sidney Perry Foundation, Vegetarian Charity, Ameobi Hardship Fund for International Students (AHFIS), and loans from friends and family – I am eternally grateful for the generosity of all of them.

This PhD studies wouldn't have started (and finished!) without my family. It is my father's unconditional faith in me that gave me the courage to take on this arduous adventure. Lows experienced during this journey would have been impossible to navigate without the support & inspiring words from my mother, my sister (Shreya) and my brother-in-law (Rashesh). My sister (Shruti) and my brother-in-law (Hardip) had to

endure the most because they are here in the UK. I can't thank them enough for all they have done for me.

Last but not the least I thank my friends – Amit and Arpita became my family in the UK; Florence, Kuntal, Moon and Zhenyu took care of me like an elder siblings would do; I couldn't have asked for a better housemates then Izabela and Pratik; Holgate Hall family (especially Christine, Helen, Konstantinos (Kosta) and Sue) has been my rock; Chaitanya and Mudita have provided very good emotional support. There are many more whom I met at York – all have been very supportive in more ways than I can mention here! My time in York has become far more special because of meeting you all, I will cherish the time we shared for the rest of my life.

DECLARATION

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

I declare that the spreadsheet mentioned in section 5.4 has been developed by Dr Crispin-Bailey and the data in it has been obtained from experiments conducted by me.

I certify any ideas, techniques, quotations, or any other works of other people included in my thesis are fully acknowledged in accordance with the standard referencing practices.

CHAPTER 1

INTRODUCTION

Superscalar Stack Issue Array (SSIA) is a novel stack structure that has the superscalar capability on top of other advantages. This thesis revolves around the investigation and optimization of SSIA, which was proposed by Bailey and Mullane [1], more details on SSIA will be presented in CHAPTER 2 of the thesis.

According to key texts of *Computer Architecture*, there are four kinds of processor based on internal storage structures, and one of these kinds uses the stack-based structure for internal storage – they are widely known as *Stack Processors*. SSIA will find its usage in the stack processors. Hennessy and Patterson, writers of one of the key texts on computer architecture, argue against the stack processors based on historical arguments [2]. One of the reasons for stack processors' limited capability is due to unavailability of parallelism in stack processors – SSIA is designed to overcome this particular limitation. Nowadays improving computer processors is becoming more and more difficult [3] , many also believe we are near to the end of semiconductor scaling [4] – more on this is in the next section (1.1). Hence it's important to look at the alternatives – more details on this in section 1.2. At the Advanced Computer Architecture Group (ACAG) of University of York, we are investigating if Stack Processor can offer any advantages [1] [5] [6] [7] [8]. One of the next things to investigate is VLSI Optimization

for stack processor to equalize the playing field with the widely popular counterpart – register-based processors. This thesis aims to make progress in the direction of VLSI Optimisation for stack processors.

1.1 Background

Gordon Moore, the co-founder of Intel and then director research & development division of Fairchild Semiconductor, predicted in 1965 that by the year 1975 the number of components in the single integrated chip (IC) will be 65,000, by doubling every 12 months. This prediction is widely known as *Moore's Law*. Later in 1975 Moore revised this statement that the numbers of components will double every 24 months. Till today this prediction remains valid, in fact, revised prediction can be considered little pessimistic because the number has roughly doubled every 18 months. The constant doubling of components has been achieved mainly by miniaturization of *semiconductor device fabrication node*. The latest technology nodes today, which are in mass production and available to consumers, are 14nm/10nm/7nm. '14nm' in the 14nm technology node is typically the size of process' gate length. It is this consistent miniaturization which allows Moore's law to remain valid till today. But this progress wasn't a straightforward task. There were many challenges that needed to be tackled to continue the miniaturization and in sustaining Moore's law. Some of these challenges are mentioned below.

- For 180nm technology of 1999, the conductor in use was too resistive, so engineers replaced aluminium with copper [4].
- For 130nm technology of 2001, features were smaller than light wavelength, thus conventional lithography was replaced with computational lithography [4].

- For 90nm technology of 2004, there was a limitation in the active current – this was overcome by strained silicon [4].
- For 45nm technology, the gate leakage was limiting downscaling of technology, so the conventional metal gate was replaced by Hi-K metal gate [4].

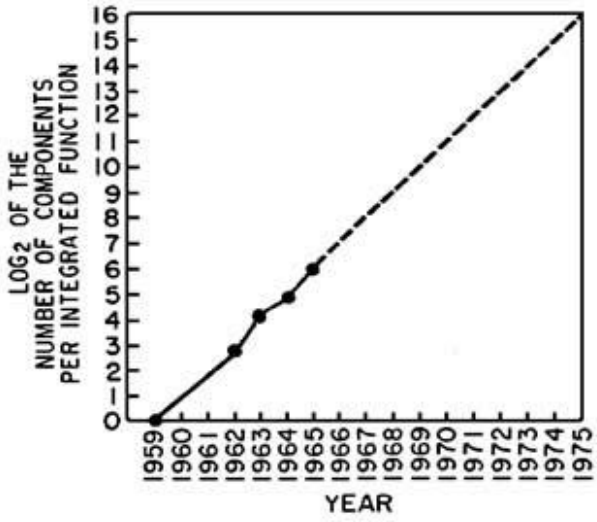


Figure 1-1: 1965 estimation by Gordon Moore about the number of components(transistor) per integrated function (IC chip) for the next ten years. [9].

Figure 1-1 is a prediction graph drawn by Moore for *Moore's Law* in his revolutionary paper published in 1965 [9]. **Figure 1-2** is a graph of CPU trend and it validates Moore's prediction (the green line – the number of transistors) [3]. It can be seen in **Figure 1-2** that clock speed (blue line) started to stall after 2004 – which was consistently increasing at the beginning. The same can be said for instruction level parallelism too (Purple line – Perf/Clock).

1.2 Recent Technology Trends

Constant efforts are being made by the semiconductor industry to sustain Moore's law (or in other words to continually enhance processors' performance), some of the recent innovations, of widespread interest, are listed below.

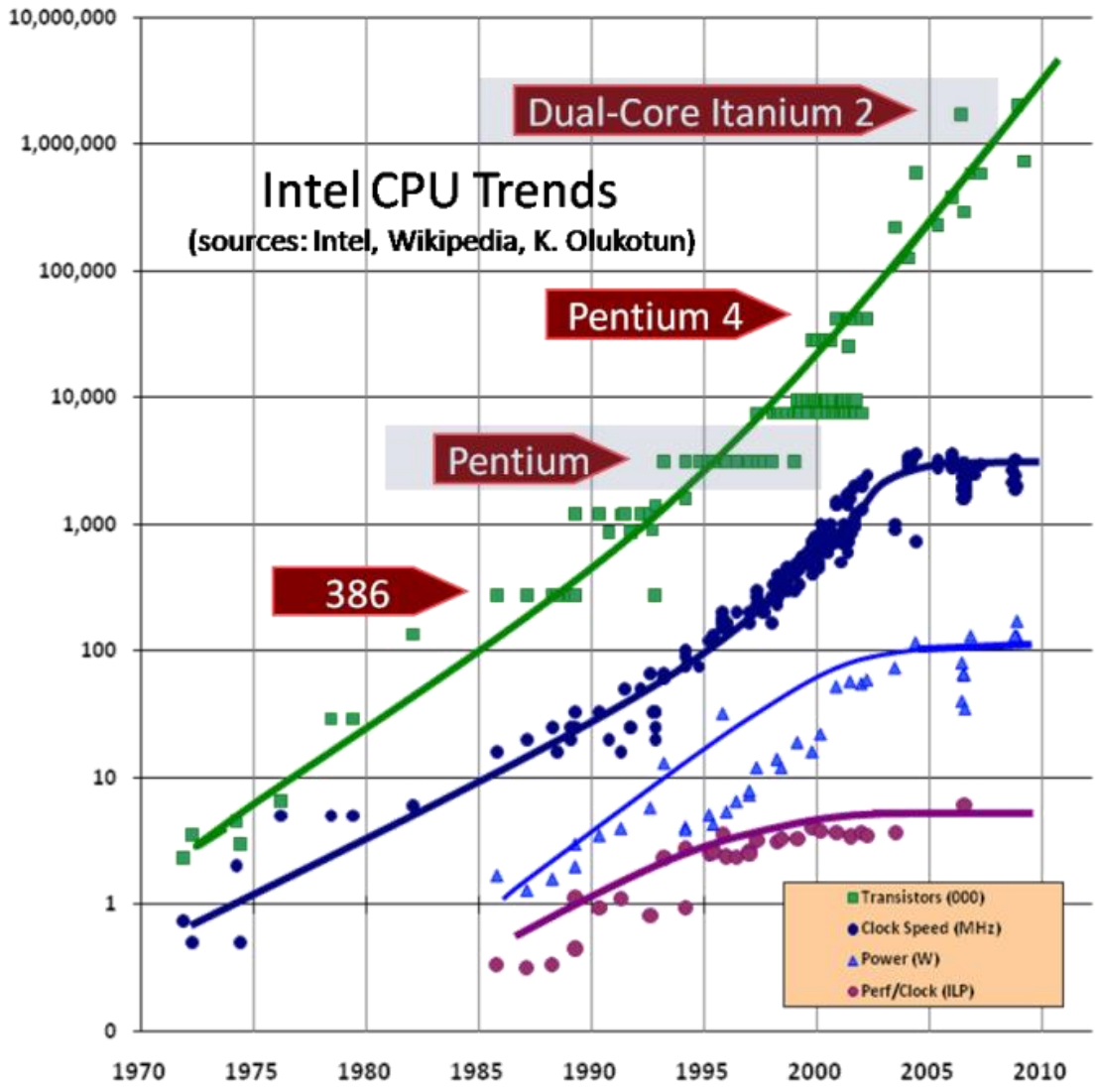


Figure 1-2: Number of transistors(000), clock speed (MHz), Power (W), and Performance/Clock (ILP) for Intel CPUs from 1970 to 2000. [3].

The following sub-sections will provide more details on the below-mentioned ideas/innovations.

- Miniaturization of technology: 5nm
- Parallelism
- Three Dimensional Integrated Circuits (3D ICs)
- Stack Processors: Back to the Future

1.2.1 Miniaturization of technology: 5nm

As mentioned earlier, Moore's law has remained valid since 1965, mainly because of this particular approach. It is widely believed that this will continue for some time. The latest development (as of September 2018) in this direction is the development of Apple's A12 Bionic processor using 7nm technology node, for the mass consumer market. Next in line is 5nm technology which is expected to arrive in 2020. Intel has confidently announced that Moore's law isn't going to end any time soon. However, Intel hadn't revealed the next big invention yet [10]. Though looking at the history this is evident, because Intel usually hadn't revealed its innovation well in advance. One of the Intel's technical manufacturing managers mentioned that Intel hadn't stopped working on its past inventions and is continually evolving them e.g. Intel is still working on the fourth generation of strained silicon, the third generation of the high-k metal gate and the second generation of tri-gate.

1.2.2 Parallelism

One of the topics of interest is parallelism – thread level parallelism. This may not be helpful in sustaining Moore's law but can certainly improve the processor's performance. Tasks performed by processors can be categorized into either serial task or

parallel task. In a heterogeneous architecture (CPU & GPU combined) the task of serial nature will be executed by CPU, whereas those of parallel nature will be better executed by GPU because of their expertise in respective areas. Therefore having a heterogeneous system of CPU and GPU can enhance the performance. Its working example is an internet browser *Internet Explorer*. IE 9 has a capability of harnessing heterogeneous system architecture and according to experts IE 9 has been 400% faster than IE 8 for certain web pages [10]. AMD is one of the leading companies working on developing heterogeneous system architecture. Another kind of parallelism is Instruction Level Parallelism (ILP) where multiple instructions can be executed simultaneously. ILP is a standard feature in register-based architectures but was believed to be impossible for stack-based architecture, but some efforts have been made in this direction [1] [5].

1.2.3 Three Dimensional Integrated Circuits (3D ICs)

In section 1.1 and 1.2.1, it was explained that cramming more and more components and scaling down of technology node is becoming more and more difficult. One way to overcome this problem is to use the third dimension, instead of planar 2D contemporary version. 3D IC is just one way of utilizing the third dimensions in semiconductor technology [11]. Apart from Moore's law being sustained, there are other advantages too of using the third dimension, such as heterogeneous integration, design and circuit security [12].

Another effort, slightly different from 3D ICs, is related to changing the building block of the IC – the transistor. IBM recently tweaked transistor, a basic element of an integrated circuit. The new transistor contains one atom thick graphene layer coating [4]. Major issue after cramming a large number of components into a chip is leakage current.

Leakage current leads to power loss and heating. The coating helps overcome the leakage current. Consequently, this technique is believed to be a saviour for Moore's law and will cut down the cost of computing. Though this transistor is yet to be used in any chip and has only been demonstrated for a simple circuit. It is expected to come out as a chip before 2020. Further, it is hoped that this technology will enable to scale clock speed of transistor to 100 GHz [5].

1.2.4 Stack Processors: Back to the future

As mentioned earlier on page 1 that here, at Advanced Computer Architecture Group (ACAG) of University of York, we are investigating if stack processors can help sustain Moore's law or in other words improve the performance of processors using an alternative architecture. Literature review suggests that stack processors are an excellent choice for small embedded systems because of their inherent advantages, mentioned later in this document [13]. It was also found that they could be attractive for many-core systems as well. As the vast majority of processors in the real world are embedded in nature stack processors appear appealing field of investigation. One bottleneck to stack processors that has resisted many attempts to overcome it is the inability to support ILP, as mentioned earlier, however the past research conducted at ACAG on stack processors busted a myth that stacks processor doesn't do parallelism and thorough investigation for Instruction level parallelism had already been done by Brendan and Mullane, and the door now seems open for ILP to be exploited in stack processors [1] [6]. Scheduling techniques, already very much used in register-based processor, was also proposed and integrated into a novel C compiler [7]. Currently, ACAG is working on NOMAD, a novel

superscalar stack processor, funded by Technology Strategy Board (now called Innovate UK) and Cybula [14].

1.2.5 A concluding remark

The above list by no means is supposed to be comprehensive and is just to give an idea about some of the efforts made by the community of researchers to enhance the processor performance. Trends presented above are the only tip of the ice-berg for technologies/ideas out there, but it's safe to say that there are plenty of ways to continually improve the performance of the processor, whether Moore's Law may sustain (or not!), as explained above in this section.

1.3 Introduction to Stack Architecture

According to Hennessy & Patterson, there are 4 major class of processor based on instruction set architecture and their internal storage [15]. Among them, the class of processors which uses stack structure for internal storage is stack-based processors and is widely known as stack processors.

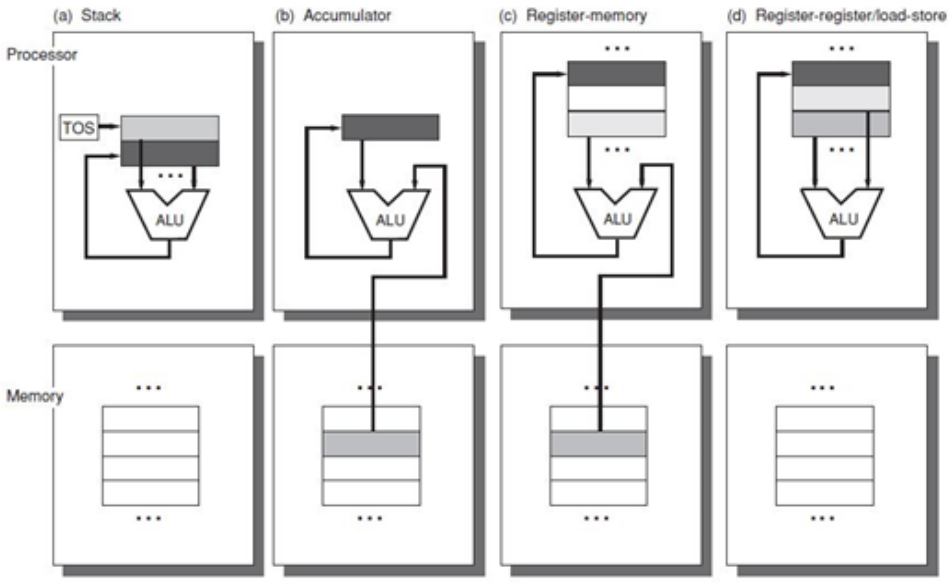


Figure 1-3: Four common types of instruction set architecture based on the storage location of their operand [15].

Stack processors have been around since the early days of computers. In fact, stack processors dominated the industry back then. Two generations of stack processor – entirely independent of each other – can be found in the literature. The first generation, like any other early days computers, were aimed at complex mathematical computation, whereas the second generation, mostly initiated by Charles Moore, covered the chip based, application specific processors. Current days' stack processors are still considered to be part of the second generation. More details on both generations have been covered in section 2.1 of the literature review chapter of the thesis. JAVA, one of the most important modern day programming languages, can also be associated with the stack. Compilers of JAVA produce an intermediate language called Java Bytecodes. Processing of Java Bytecodes requires a software interpreter called Java Virtual Machine (JVM). JVM uses stack-based architecture. Thus JVMs are the most recent or say modern days stack processors. The purpose of the JVM is to provide software compatibility across many platforms, with the hope of "write once, run everywhere."

Stack processors can be better understood by reflecting on its differences from the register-based processor. **Figure 1-4** and **Figure 1-5** represent architectures of both types of processor architecture in a highly simplified form. Both architectures have different process unit. Arithmetic and logic unit (ALU) of stack-based architecture obtains its operand from the stack. These operands are the top two elements of the stack and implicit to ALU whereas ALU of register-based architecture has to define their operands explicitly. Operands of register-based architecture can come either from registers or from

memory. The same operation is performed differently in both processors architecture. Following the example of the addition of two numbers, A and B will further clarify the differences between the two architectures. The register-based processor will use two registers R1 and R2. Initially, data will be moved to these registers and later these two registers will be added after explicit addressing. Result of addition will be stored in the accumulator register. Pseudo code for this operation is shown below.

```
MOVE R1 A
MOVE R2 B
ADD R1 R2
```

Whereas in stack processors PUSH instruction will put both numbers on the stack then instruction ADD will implicitly add both numbers and put the results on the top of the stack. Pseudo code for this operation is shown below.

```
PUSH A
PUSH B
  ADD
```

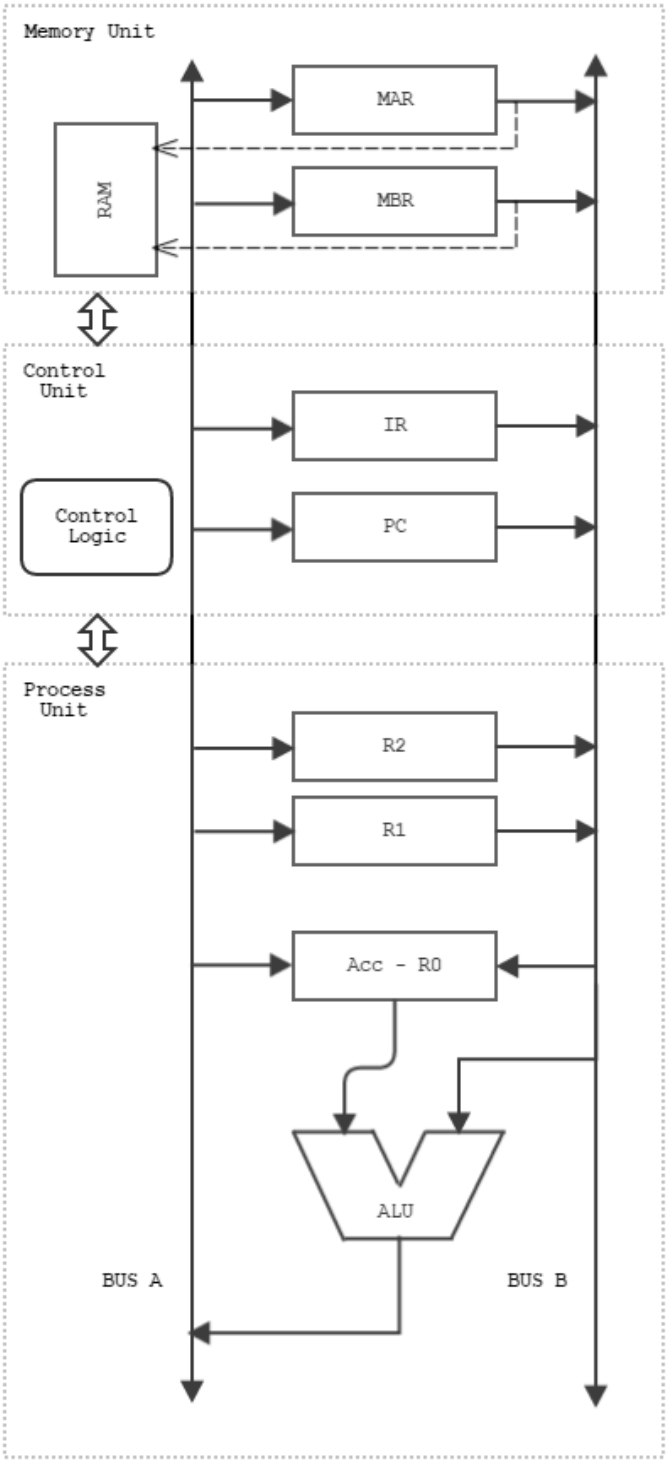


Figure 1-4: A block diagram for the architecture of a generic register machine [16].

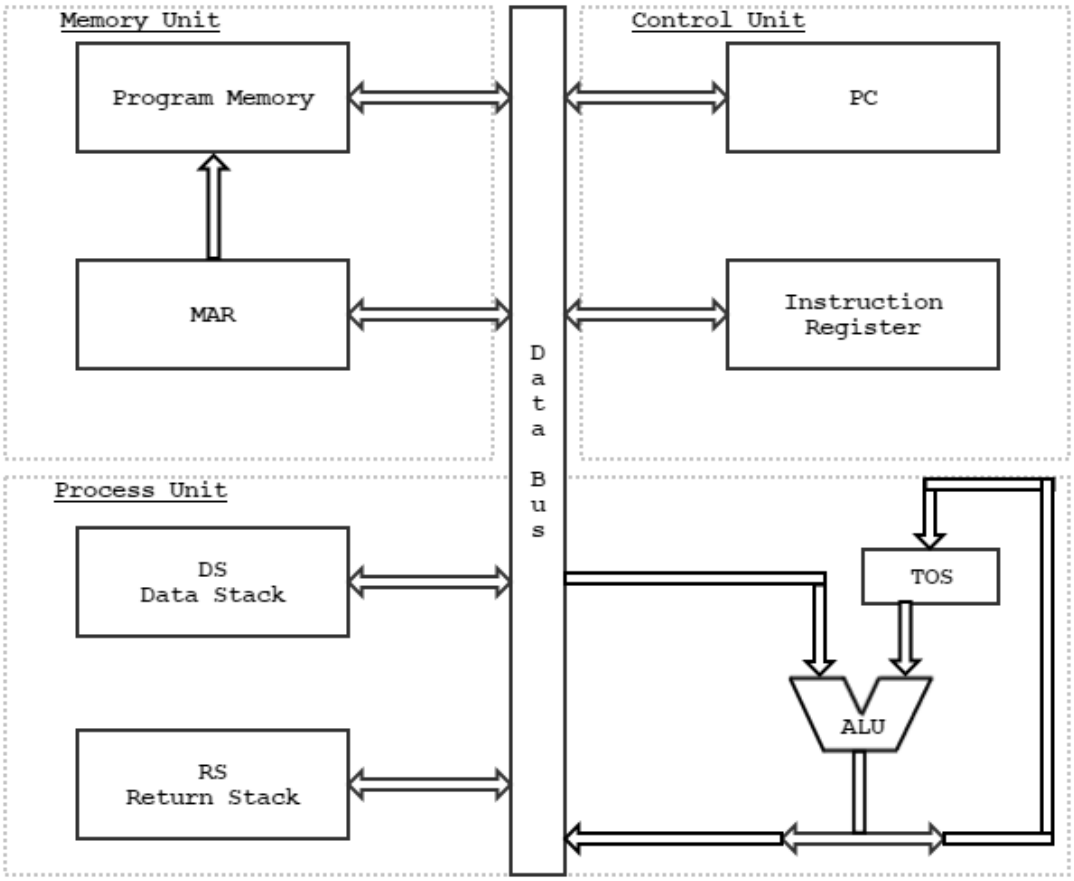


Figure 1-5: A block diagram for the architecture of a generic stack machine. [17].

1.4 Motivation and Rationale

Key motivations to research the stack processors are as follows. Firstly, due to the **advantages** of stack processors such as simpler architecture which can be advantageous in the niche area where simple and small microprocessors are required. Secondly, it appears that there is a **clear lack of research** in this area because prominent ideas such as parallelism and multi-core processing haven't been investigated for the stack-based processor. Thirdly, **previous research** at York (and elsewhere) has shown promising

prospects for stack processors. The following sub-sections goes into more details of just now mentioned key points. The last sub-section (1.4.4) states the rationale in the form of a **research question.** The research question also sets out a task that needs to be completed to answer the research question. The work in thesis keeps coming back to these tasks to make sure our efforts are in the desired direction.

1.4.1 Advantages

Stacks are considered to be the most basic structure that can be used in processing well-structured code [17]. Further, the efficiency of the stack at an expression evaluation is a well-known fact in the industry. Implicit addressing scheme of stack entails quite a few advantages – such as retrieval of an operand for the computation without specifying the location. Implicit Addressing combined with simpler instruction complexity offers added advantages, which includes small program size, better processor performance, fast procedure calls & interrupt handling, less processor & system complexity, and suitability for the embedded system [17]. As a smaller program reduces the requirement of memory and power in the processor, it also benefits the stack processor. A procedure call, interrupt handling and recursion can be handled more effectively by stack-based processor then register-based processor. System complexity & processor complexity are well maintained in stack processor compared to CISC, which has more processor complexity, and RISC, which has more system complexity. These specific advantaged makes stack processors preferable over register-based processors for application requiring embedded systems. Stack processors are also much more efficient at running certain types of the program than the register-based machine, particularly program which is well modularized [17].

1.4.2 Clear lack of research

'Computer Architecture: A Quantitative Approach' is a key text written by Hennessy & Patterson on the field that details architectural ideas and compiler technologies developed over the past 60 years. Only a few mentions of stack processors in this book give the impression that there is a clear lack of research in this area. This belief is further strengthened by the fact that investigation of Instruction Level Parallelism (ILP) for stack processor is a very recent activity (2006), whereas researchers have fully exploited ILP for register-based processors by 2004 (such inference is because Hennessy & Patterson stated that now there is very limited scope left for ILP in register-based machine). Further, it is widely believed that stack machine will involve a huge data movement between processor and memory and it doesn't have good compiler [2] but recent research by Shannon on this area has shown that a well-designed compiler can reduce this movement [7]. This belief is not surprising considering the presence of only Forth language compilers for stack processors. Thus, the dearth of research of ideas, such as ILP, multicore processing, compiler technology and VLSI optimization, is a genuine concern that should be addressed. Therefore we, at advanced computer architecture group, are wondering – What if stack processors would have received the same amount of research efforts as that of register-based processor and integration of currently available advanced techniques? Hence our motivation is to give stack processors a fair chance and ultimately estimate the capabilities of stack processors.

1.4.3 Promising results from previous research

Recent research in the area of stack processor has resulted in promising results and hence this is one of the motivations behind exploring this area further. At York, we

have made progress towards improving stack processors. Research on instruction level parallelism (ILP) and scheduling algorithm has been already mentioned in sub-section 1.2.4 [5] [6]. VLSI Optimization for stack processor has also begun [1]. Charles Moore, who was at the forefront in designing a second-generation stack processor, is investigating multi-core for stack processor [18]. Hennessy and Patterson expressed concerns that there aren't that many hardware specifically to execute Java Virtual Machine (JVM) – which has stack architecture, but Schoeberl has proposed one recently in 2005 [19]. Therefore it appears progress is being made in this direction and it's important to continue.

1.4.4 Research Question

Given that there are particular goals in this PhD project, and that these are driven by the motivation given in preceding sub-sections (and also introduction given in preceding sections) a clear research question can be posed for this thesis:

"Can the performance of superscalar stack structure be significantly improved by the use of advanced design techniques, and will this make them more competitive with register file?"

To answer the above question, there may be several valid approaches. From these, a methodology was formulated that allowed the research question to be addressed competently in the realistic timescale of a PhD and the following tasks have been set:

1. Develop a baseline performance model, the *Standard Cell* Library, which builds upon existing work, and which can form a point of comparison against any improvements achieved.

2. Evaluate the custom design approach, create a new *Custom Cell* library, and evaluate their performance relative to the above point (1).
3. Given a standard cell library, and a custom cell library, compare the SSIA structures possible in each case and evaluate how much improvement can be obtained.
4. With suitable optimal or highly optimized designs for SSIA, repeat the comparison of Bailey and Mullane for SSIA versus register file models, and determine if competitiveness has improved significantly.
5. Document and test more advanced VLSI design techniques that can be used in the future to inform future research directions, and highlight possibilities for continued work in future research.

1.5 Thesis Structure

This thesis consists of 6 more chapters and they are organized as follows:

CHAPTER 2 is a literature review chapter and has three sections. The first section reviews literature related to stack processors, starting from the very first stack processors to modern day stack processors with emphasis on stack processor research that happened at York which also includes SSIA – a superscalar stack structure on which this thesis is centred. The second section reviews literature related to VLSI design techniques. The third and the last section provided an explanation of some general terminology.

CHAPTER 3 starts with an explanation of the experimental setup and methodologies that are going to be used throughout the thesis. The chapter is intended to evaluate the performance of Fundamental Components and Multiplexers of SSIA. The evaluation is done in varying fanout condition. An important rationale for performing this

initial work is to establish a benchmark for SSIA performance, remembering that the hypothesis of this thesis is to answer the question 'Can advance VLSI design techniques improve the performance of SSIA?' The cell library developed out of this chapter is going to be referred to as a *Standard Cell* library throughout the thesis. This chapter presents the baseline data that is going to be used to measure any improvements that are made in later chapters through more sophisticated design techniques. Additionally, the baseline data includes many performance parameters (such as Area, Delay, Power, Capacitance and Fanouts), these help in painting a realistic picture of performance. The components of SSIA are used in varying fanout conditions; hence the detailed fanout analysis helps us see the trend of their performance. The baseline data also allows observing the close relationship between capacitance and propagation delay.

CHAPTER 4 explores the effect of varying transistor parameters in order to create high-performance components. The chapter starts with a comparison of wider transistors and multi-finger MOSFET implementations. The aim of the chapter is to know what the performance gains are and what the cost penalties are and thereby investigate the tradeoff and also if there is a best possible approach for SSIA implementation? The chapter also investigates the custom design approach at various design levels (Transistor, Component and System level). The important rationale - here as well - relates to the fundamental question asked in the hypothesis - Can custom design deliver worthwhile benefits over and above those achieved using the Standard Cell approach? The cell library developed out of this chapter is going to be referred to as a *Custom Cell* Library. In particular, it is found that faster components are possible at the expense of larger chip area and diminishing return trend is observed. It is also observed that it is advantageous to use

multi-finger MOSFETs to achieve higher drive strengths, but also that there is a tradeoff against the penalty of the effect of 'self-loading'. Generally, this implies a certain number of fingers offer the best performance gains. It is also observed that, contrary to the naive view, deliberately reducing the speed of some components may offer area benefits, without reducing overall system performance.

In CHAPTER 5, the impact of using the custom vs standard cell library is evaluated at a higher level. Specifically, the impact of various component designs are considered with respect to the more complex structure of the Superscalar Stack Issue Array (SSIA), and it is demonstrated that many permutations of component combinations are possible, delivering better speed, better area, better power, or indeed a compromise where some or all of these factors are balanced. Because so many permutations are possible, we contrast the manually chosen (directed design) cases, against those developed by a simple evolutionary algorithm, which attempts to find a best case combination of components for a given design goal. It is observed that suitable component combinations may achieve high speed, low area, as might be expected. However, high speeds can also be achieved with far less area cost penalty than the 'naive' solutions, by exploiting the individual members of the component library. Likewise, power and speed may be traded off beneficially. The use of evolutionary approaches also yields solutions that balance specific design priorities in ways that are not easily arrived at through manual design processes (i.e. trial and error).

In CHAPTER 6, we consider further techniques to improve the performance of SSIA, and selected techniques, such as pre-charging, power-gating and current-mode implementation, have been analyzed and preliminary results have been produced. New

avenues for SSIA enhancement have been proposed and some preliminary exploratory work on the potential of these cases has been undertaken. It is observed that speed can be improved by the use of pre-charging techniques. It is also observed that power-gating is a useful technique to save power at cost of reduced speed. Finally, it is predicted that the current mode can improve SSIA performance.

CHAPTER 7 is to summarize all the evidence and discuss the significance of work in relation to the research question set in sub-section 1.4.4. A future direction for the research has also been provided in this chapter.

1.6 Summary

This chapter provided some background to set the context of the work in section 1.1 and then the following section (1.2) discussed the recent technology trends and has sub-section dedicated to stack processor research. The third section (1.3) provided an elementary introduction to the stack processor. The fourth section (1.4) listed some of our initial motivations to investigate the stack processors in general. The fifth (1.5) section provided the structure of the thesis. The next chapter reviews the literature. Please note in advance that the data presented in the thesis, particularly in the contribution chapters (Chapter 3, 4, 5 and 6), are rounded off to suitable precision to improve the presentation of data.

CHAPTER 2

LITERATURE REVIEW

This is a literature review chapter and has three sections. The first section reviews research related to stack processors starting from the very first stack processors to modern day stack processors with emphasis on stack processor research that happened at York and SSIA – on which this thesis is centred. The second section reviews VLSI Optimizations and the last section (2.3) presents the identified gap, prospective impact of bridging the gap.

2.1 Stack Processors

This section reviews the past research related to the stack processor. The section is categorized into sub-sections. The categorization of stack processors into generation has been inspired by LaForest's thesis titled 'Second-Generation Stack Computer Architecture' [20]. The first sub-section (2.1.1) talks about stack processors during the early days of computers (1960 - 1980). The second sub-section (2.1.2) talks about the new wave of stack processors that developed with the advancement in VLSI technology (1980 - 2000). The third sub-section (2.1.3) talks about stack processors of the 21st Century and can be called 3rd generation, but it is a continuation of the 2nd generation with new ideas. The fourth sub-section (2.1.4) is focused on stack processor research

done by members of the ACAG research group. The last sub-section provides more details on the SSIA of stack processor.

2.1.1 Stack Processors: First Generation

Table 2-1: Prefix and postfix mathematical notation for expression [2*(3+3)/3].

Prefix Representation	Postfix Representation
/ * 2 + 3 3 3	2 3 3 + * 3 /
/	2
/ *	2 3
/ * 2	2 3 3
/ * 2 +	2 3 3 + 2 6
/ * 2 + 3	2 6 *
/ * 2 + 3 3	12
/ * 2 6	12 3
/ 12	12 3 /
/ 12 3	4
4	

Origin of stack principle can be attributed to Jan Łukasiewicz. Łukasiewicz proposed prefix notation, also known as *Polish Notation*, in his book *Elements of Mathematical Logic* [21]. Using his method parenthesis-free mathematical expression can be written. Taking inspiration from prefix notation Charles Leonard Hamblin proposed postfix notation [22], which became a founding principle for stack processors. In the

postfix notation, also known as *Reverse Polish Notation*, operators are put after the operands, unlike prefix notation in which operators are before both operands.

Consider following simple arithmetic expression with and without parentheses. The second expression (without parentheses) doesn't lead to intended results, hence the same expression should be written using either prefix or postfix notation, to obtain the correct answer if the parentheses-free notation is required, as shown in **Table 2-1** above.

```
2*(3+3)/3 = 4

2*3+3/3 = 7
```

The idea of prefix notation of Jan Łukasiewicz was realised in Stanislaus relay calculator by Friedrich Ludwig Bauer [23]. Bauer's relay calculator had *ZahlKeller*[1] and *OperationsKeller*[2] (more details on it are available in [24]). Bauer's ideas were later used in developing language called ALGOL 60, and according to Philip J Koopman Jr. ALGOL like languages share a close relationship with stack architecture [17].

The idea of using stack like structure was proposed by Alan Turin too, independently, and his idea was to use of stack-like data structure for subroutine related computation in Automatic Computing Engine (ACE). Much later in 2003, David E Newton wrote that Turing had proposed use of BURY (analogous to PUSH) and UNBURY (analogous to POP) operations in ACE [25].

During these early years of computer, stack principle was under active investigation and draw widespread interest from industries, universities, and independent researchers. This resulted in stack principle manifested into many sophisticated stack

[1] German word meaning Number Cellar
[2] German word meaning Operations Cellar

processors during 1960s and 1970s [17]. Some of them (popular once) are mentioned below.

Burroughs Corporation: Burroughs Corporation designed and manufactured many stack processors during the early period of computers. Burroughs was one of the major computer companies of the United States in the 1960s. Computers sold by Burroughs fell into three major categories - *Small Systems, Medium Systems* and *Large Systems*. Large Systems were mainly pure stack processors, i.e. no programmer addressable registers [26]. The first computer of large systems series was B5000. After B5000, a number of improved and transformed stack processors were designed and manufactured at Burroughs. LaForest's, after his historical review of stack processors, confirmed Koopman's belief that all the large systems computers were aimed at improving the execution efficiency of ALGOL [20]. Important industries (banking, aerospace, academic research, military, etc.) were using Burroughs computers [26]. Some of the well-known clients of the above-mentioned industries were Barclays [27], NASA, and US Army [26].

Digital Equipment Corporation (DEC): DEC produced a series of PDP 11, a 16-bit microcomputer. Like Burroughs, DEC was another major company of the United States in the 1960s. PDP 11s WERE NOT stack processor, but Koopman writes that this was an early general-purpose register machine, and as opposed to today, included the only a subset of capabilities possessed by stack processor [17].

English Electric Company: English Electric Company developed a stack processor named KDF9 [28]. KDF9 drew inspiration of stack principle from Hamblin's postfix notation. Unlike another first-generation stack computer, KDF9 used pair of

stacks, this is important, because it is major differentiation between first and second generation (discussed later in sub-section 2.1.2) of stack processor, according to LaForest. He wrote that most first generation processor uses a single stack whereas most second generation stack processors used dual-stack [20]. This way KDF9 can be considered ahead of time second generation stack processor. But, most likely, the company's focus was to compete with B5000 and IBM 360 (extremely successful computer of the time) instead of improving KDF9 [20]. Koopman wrote the following quote about KDF9 [17].

"*... [KDF9] introduced many of the features found on modern (second generation) stack computer…* "

Hewlett-Packard (HP): One of the stack processors developed by HP was *HP 3000*. LaForest, after his historical review of stack processors, wrote that HP 3000 was very similar to Burroughs' computers – with modification and improvements [20]. HP 3000 had one of the longest lifetimes, and support was available until the current decade. HP renamed this to HP e3000, this was to emphasize that the system was equipped for web and internet uses [20]. The first processor of this series was launched in 1972. As technology immensely progressed from 1972 to 2003, it seems that the size of later models would have considerably reduced from earlier models.

International Business Machine (IBM): IBM was one of the major computer companies of the 1960s, and its revenue was higher than the combined revenue of all other computer companies of the United States [29]. At IBM, two – EULER and APL Language – stack-based micro-coded emulations[3] were investigated which ran on IBM

[3] Micro-coded emulation is process of translating code for a particular machine to another machine.

360. EULER is an extension of the ALGOL language and APL is an inherently interpreted language. Both processors were direct execution engines for respective languages [17].

There were many more stack processors that were part of the first generation, more on it can be found in [17]. With VLSI technology becoming feasible the first generation started to fade, and the second generation started to take shape. The review of the second generation stack processor is provided in the following sub-section.

2.1.2 Stack Processors: Second Generation

Stack processors which came out after 1980 has been categorized as the second generation. During the 1980s the first microprocessor was already in the market and it was full of innovation for the technology industry – the first PC was launched, and research on RISC began. Similarly, a completely new wave of stack processors was generated in the 1980s. LaForest, after his historical review, wrote that the new wave of stack processors was mainly initiated by Charles Moore and documented (in form of a book titled "Stack Processors: the new wave") by Philip J Koopman Jr. [20]. Charles Moore was a freelance programmer in 1960s, around the same time Moore invented computer language *Forth* and founded *Forth Inc*. Forth language is heavily dependent on the stack structure. Program written in Forth is executed with the help of *Forth Virtual Machine* [30]. Moore wasn't satisfied with hardware used by Forth and thus he started developing custom processors for Forth [31], these custom processors' for Forth became major candidates of second-generation stack processors. Efforts in direction of developing custom hardware for Forth led him to start his second company *Novix Inc*. At Novix, Moore designed the first prototype NC4000. It was improved and launched with

the name NC4016 in 1985. Its further improvement led to NC6016 which was licensed to Harris semiconductor and marketed with name RTX200 in 1987. Later Moore collaborated with Russel Fish and proposed Sh-Boom in 1988.

During the 1980s one of the major debate was about RISC v/s CISC. Amidst this debate slightly less known approach was MISC. Charles Moore wrote the following [32]

"*… the principle of simplicity (in RISC) was not enforced enough to realize the full benefit from this principle. In the MISC architecture, we like to explore the power of simplicity to its limit …*"

Moore's idea about achieving the above was through stack architecture. Another notable processor, among the MISC, is INMOS Transputer, explained in the next paragraph. Moore also invented MuP21. It was first of a family of chips termed MISC in 1995. Around the same time, he also invented P series stack processors in VLSI using his CAD tool OKAD. Internet was a new thing around that time, so he developed i21 using OKAD for internet usage. Later in 1998 he collaborated with Jeff Fox and co-founded his third company *Ultra Technology*, there he developed F21. More details related to this is available in Koopman's book about stack processors and some unpublished information is available at Jeff Fox's online repository [17] [32].

Transputer's popularity was attributed to its parallel computing ability. Transputer was among one of the early architecture to exploit parallel computing. Transputer was designed and produced by INMOS. INMOS was an early computer company (of the 1980s) based out of Bristol in the United Kingdom. Transputer derived its name from Transistor and Computer. Inventors believed that Transputer will become the building block (just like a transistor) for the yet to come computers of the future and hence the

name. It seems the same belief would have demanded simplicity and hence the stack architecture. As already mentioned, Transputer was popular because of the ability of parallel processing. John Catsoulis wrote that Transputer was among the first few processors which exploited parallel processing. Alan Clement, in his book about Computer Architecture, wrote that modern parallel processing is inspired by the Transputer.

Figure 2-1: Two stills from BBC Micro episode about Transputer which are depicting parallel processing capability [33].

Above screenshots are taken from an episode about Transputer on BBC Micro Live. The episode presented the simplified working of Transputers. The animation shows a particular butterfly roaming around two different screens. Both screens are connected to their respective Transputers and both coordinates and communicate with each other and then produce the animation [33].

Koopman's Processors: Alongside documenting Moore's work Koopman designed WISC 16, WISC32 /RTX 32P [17]. These processors were of great use for stack-based processing. These were the successor of Forth-language processors but also supported other programming languages.

FPGA based stack processors: Quite a few stack processors of the second generation are based on FPGAs. Various inventors, around the globe independent of each other, came out with FPGA implementation for stack processors. They are documented by Jeff Fox in his online repository [31].

NORMA: Burroughs developed a stack processor, named NORMA (Normal Order Reduction MAchine) in 1986. NORMA was a graph reduction processor [34].

4stack: In 1996 Bernd Paysan published his Diploma thesis titled *Implementation of 4stack Processor in Verilog*. In 1993, Paysan started an investigation to fulfil the shortcomings of stack processors compared to superscalar RISC processor. During this investigation, he came up with novel 4stack architecture. He went on to develop complete instruction set architecture for 4stack and simulator to test benchmarks. Conducted experiments concluded that 4stack processor was very well capable of exploiting parallelism. So for his thesis, he went on to describe ISA in RTL and Structural Verilog. He also documented the synthesis for the design target ASIC process [35].

The second generation of stack processors continues into the 21st century, but with new ideas, and this has been reviewed in the next sub-section (2.1.3).

2.1.3 Stack Processors: 21st Century

Charles Moore's collaboration with Jeff Fox, to form *Ultra Technology*, developed Forth processor 'F21' in 1998. F21 was successor/extended-version of MuP21, mentioned earlier in sub-section 2.1.2, and its important feature was the presence of video processor. Moore further modified F21 to produce 'C18' in 2001. C18 was intended for parallel processing. Unfortunately, Ultra Technology ceased its operation in 2002 but Moore continued his work. In order to investigate multicore capabilities and founded

Async Arrays in 2003, which was renamed *Intellsys* in 2006. There Moore developed multicore Forth processors SEAFORTH 24 and SEAFORTH 40, where 24 and 40 are a respective number of cores present. In 2009, Moore founded another company called *Green Arrays*. There also he is developing multicore Forth processors – GA4, GA32 and GA144 are some of them [26].

In 2005, Soo Yuen Jien investigated high-level language support and low-level instruction execution for stack processor. In particular, he investigated superscalar and speculative execution of an instruction in stack processors. Jien also went on to design simulator for its novel stack processor SAFA for evaluating its benchmark [36].

In 2008, Charles Eric LaForest designed a simplistic modern stack processor named *Gullwing*. He compared Gullwing with DLX RISC, the processor used by Hennessy & Patterson in their book on computer architecture. Experiments conducted by him concluded theoretical facts that stack processors are efficient at executing recursive code and not very useful in iterative code [20].

In 2014, Subbarao designed a low-power microprocessor based on stack architecture at Lund University. It was designed using 65nm technology node and also occupies the comparatively low area [13].

In recent times, quite a few FPGA implementations of stack processors were made, but there is hardly any published material available on them now. Few examples of this implementation are P series processors developed by EForth Academy Taiwan, G16 Yoda developed by Opencores.org, MSL16 – a Forth Chip – developed by Philip Leong and Microcore [32].

As already mentioned, at the Advanced Computer Architecture Group (ACAG) of University of York, we are investigating if Stack Processor can offer any advantages, therefore it's important to look at past efforts of our group and hence the next sub-section (2.1.4) is focused on stack processor research that happened at York.

2.1.4 Stack Processors at York

The research on stack processor at Advanced Computer Architecture Group of the University of York is led by Christopher Crispin-Bailey. Crispin-Bailey's PhD research at Teesside University and thesis published in 1996 is about optimization technique for stack processor. In 1990s computer architecture research was around RISC vs CISC debate. But Crispin-Bailey, in his thesis, wrote that concepts like instruction complexity and alternative architecture are also equally important [37], and that's where his investigation of stack processor began. Crispin-Bailey went on to review stack processor's behaviour and high-level language issues and thus design concepts of stack processor [38] [39]. He identified techniques for improvement, developed mathematical models for major system components, and also proposed a revised model for stack processor hardware. Crispin-Bailey's thesis concluded that after application of investigated techniques stack processor performance can be increased. It can even match register-based processors. Crispin-Bailey continued his research on stack processor at York. At York, first he investigated interrupts and its impact on stack spill for stack processors, and then he started investigating opportunities of parallelism and scheduling techniques for stack-code. In 2002, Huibin Shi, under the guidance of Crispin-Bailey started PhD research. Shi experimented with Instruction Level Parallelism (ILP) techniques commonly applied to register-based processor specially loop unrolling,

superblock formation and branch prediction [6]. Shi's thesis concluded that there are a few ILP techniques which can be applied and those can ultimately improve the performance. Soon after, in 2005, Mark Shannon started investigating register allocation (stack scheduling) for stack processors. 'Register allocation' is a technique of keeping commonly used variables on the stack, so that those can be reused easily. During that time register allocation techniques for register-based processors, such as graph colouring, were already in use. The specific aim of Shannon was to produce a C compiler for Ubiquitous Forth Object (UFO) stack machine. UFO was designed by Crispin-Bailey and Stephen Pelc. It was one of the projects of AMADEUS, an initiative funded by DTI to develop better component technologies for computing environments [40]. Shannon built upon the simple stack scheduling techniques proposed by Koopman and proposed two new scheduling algorithms. He also implemented all three algorithms in novel C Compiler, intended for a stack machine, and compared for performance. Development of C Compiler is also an important contribution as most of the previous compilers of stack machines were for Forth [7].

The recent project at York is Non-Standard Operand Mechanism Architecture Demonstrator (NOMAD). NOMAD is a collaboration between the University of York and Cybula Ltd. NOMAD is a superscalar stack processor developed with special consideration for its power consumption. It uses a novel approach of nomadic operands. Most recently, Antonio Arnone has used NOMAD for comparing power consumption in CPU v/s Core. In a broad sense, he investigated if there is a benefit of applying accelerator core techniques in stack structure [8]. One aspect of Arnone's research tried to find out the number of instruction-per-clock (IPC) that can be executed in stack code

on custom hardware – the relevant graph is shown in **Figure 2-2**. This can also be interpreted as available parallelism in stack code. **Figure 2-2** also shows an example of scenarios with and without any code optimization. In **Figure 2-2**, it can also be seen that, out of 1600 cases which were tested, most cases (nearly 80%) have IPC of 3, 4 or 5. This finding is particularly useful to research presented in this thesis. This finding proves that parallelism in stack code is possible, and hence something like SSIA which can exploit the parallelism will be extremely useful. SSIA was proposed by Crispin-Bailey and Mullane in 2014 and that's the focus of next sub-section.

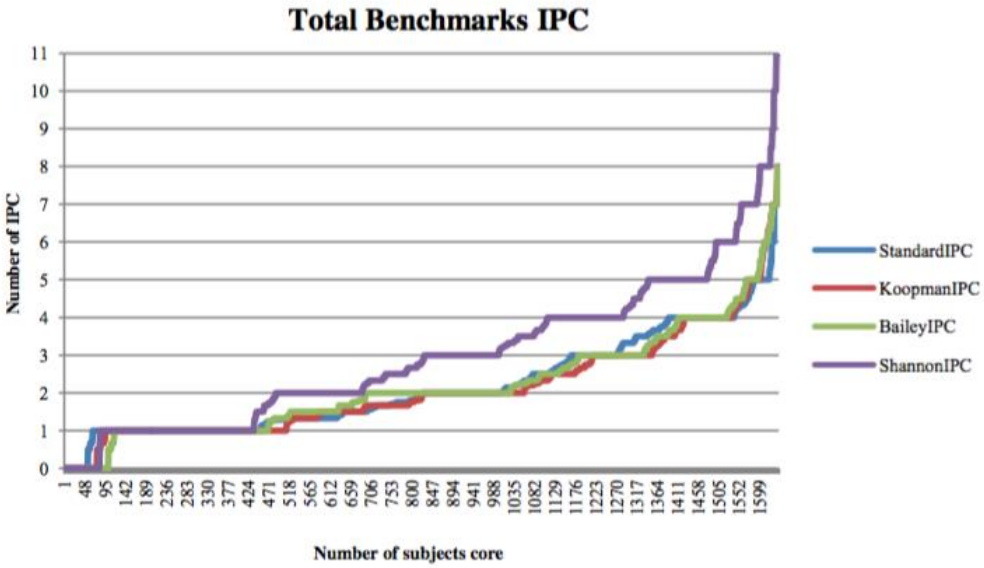


Figure 2-2: Possible number of instruction-per-clock (IPC) in the stack code. [8].

2.1.5 Stack Processors at York: Superscalar Stack Issue Array (SSIA)

Superscalar Stack Issue Array (SSIA) is an alternative operand management structure and a novel stack structure that has the superscalar capability on top of other

advantages. It is widely assumed that stack are serial in nature, and rightly so, but a technique can be invented through which stack can be used in parallel, and one such technique is used in SSIA and it uses tags. This tag match logic and memory are realised using multiplexers and latches. All of the above concepts were presented as a tutorial at HiPEAC 2012 and slides have been made public and can be downloaded from here [41].

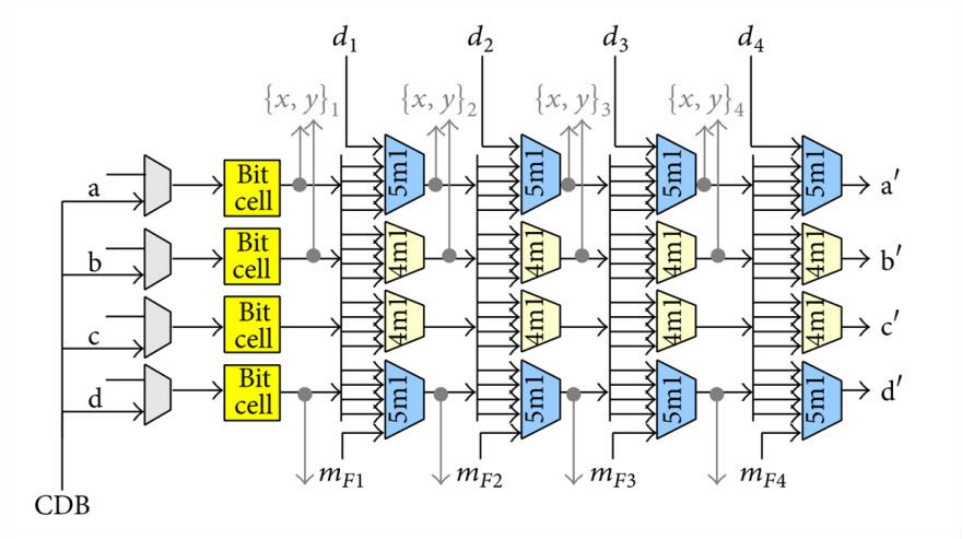


Figure 2-3: Diagram of 4-issue wide and 4-elements deep Superscalar Stack Issue Array (SSIA).

SSIA consists of multiplexers of a varying number (3, 4 and 5) of inputs. Crispin-Bailey and Mullane demonstrated in their paper that the use of various cascade logic styles to build multiplexers and ganged tristate models were viable techniques, but that ganged tri-state buffer (TNN model in **Figure 2-4**) had much superior performance, almost twice as fast as the slowest combinational logic option [1]. The relevant graph is shown in **Figure 2-4** below. The research presented in this thesis also validates the tri-state buffer approach to be superior later in sub-sub-section 5.2.3.1 on page number 119.

More investigation on making multiplexers using tri-state buffer found that other researchers also confirmed the similar advantages in the past, independently [42].

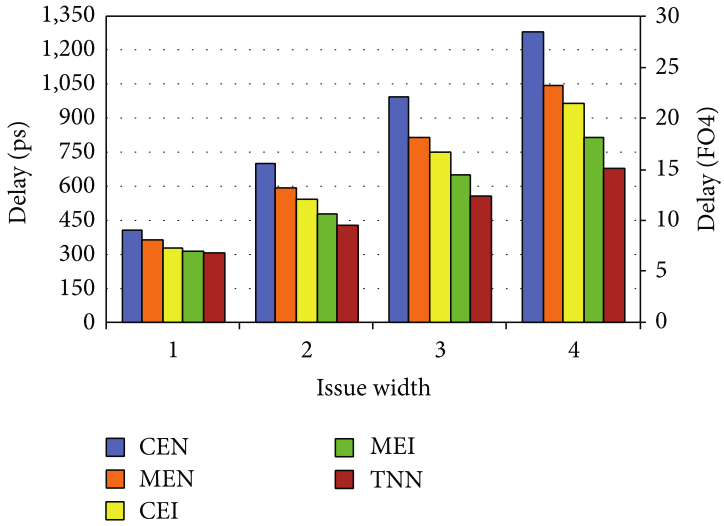


Figure 2-4: Propagation delay of various multiplexer implementations for issue widths 1 to 4 [1].

The full SSIA in operation hasn't been observed in the past but that's something that can be pursued as future research and has been mentioned as future work in section 7.4. Meanwhile, the comparison to SSIA delay predictions against those of equivalent superscalar register files (shown in **Figure 2-5** below) showed that firstly SSIA can be competitive with register files in general, but also that for high issue widths, SSIA rapidly becomes less competitive. We can summarize some important key points here:-

- Ganged tri-states buffer was demonstrated to be the optimal choice for a standard cell library in terms of delay
- SSIA can compete with register files for modest issue widths, with similar performance

- SSIA suffers from scalability issues for the delay – at high issue widths, it becomes less competitive than register file.

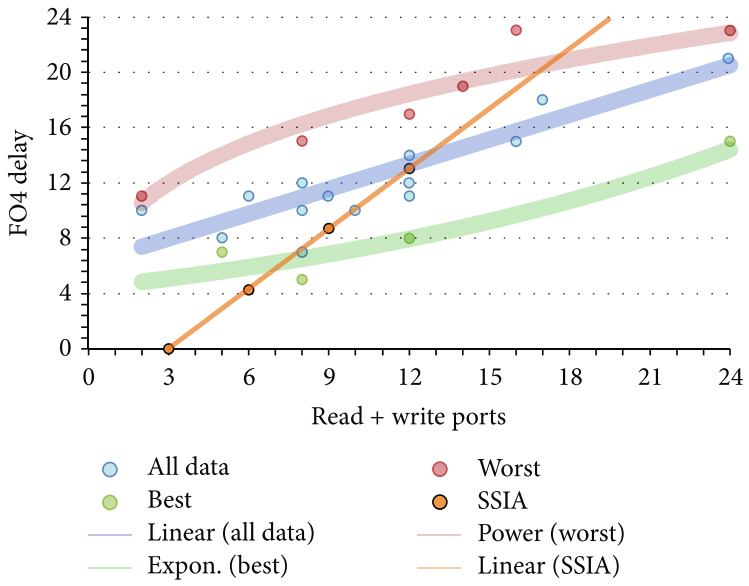


Figure 2-5: FO4 delay data for various register files and SSIA by Bailey and Mullane with respect to the number of Read+write ports [1].

However, these results were gained from a standard cell library, without the benefit of any more advanced VLSI techniques, whereas the register file data is largely based on highly optimized designs using semi or full custom VLSI design approaches. Hence advanced VLSI design technique has been one of the main goals of the research presented in this thesis, as mentioned earlier in the previous chapter. Please note that the research presented in this thesis revisits this graph based on the new data later in section 5.5. There it will be possible to see that third of the above three observation, which is a limitation related to scalability issues at higher issue width has been overcome.

This section started with a review of the first ever stack processors produced in the 1960s to stack processors of today. The next section reviews the literature related to advanced VLSI optimization techniques.

2.2 VLSI Optimization

This section reviews the research related to VLSI Optimization. The first sub-section provides an introduction to VLSI and also provides some context. The second sub-section provides a review of VLSI Optimization techniques.

2.2.1 Introduction

VLSI stands for 'Very Large Scale Integration'. VLSI refers to chip design technology that contains hundreds of thousands of transistor in a single chip. Nowadays there are millions of transistors in a single chip, which is sometimes termed ULSI (Ultra Large Scale Integration) but the term VLSI and ULSI are used interchangeably. Before VLSI we had SSI, MSI and LSI. SSI stands for 'Small Scale Integration' and has more than ten transistors in a single chip. MSI stands for 'Medium Scale Integration' and has more than a hundred transistors in a single chip. LSI stands for 'Large Scale Integration' and has more than a thousand transistors in a single chip. From SSI to VLSI, optimization of Area, Delay and Power have been a key concern for designers. These factors have been a focus in this thesis as well (e.g. in sections 3.3, 3.4, 3.5, 4.3, 4.4, and 5.3). Nowadays, at VLSI level, the complexity is extremely high. That's why the design process involves the help of computers – CAD designs – and Electronic Design Automation (EDA) tools. The conventional approach is to use standard cell design techniques to reduce production time. An automated approach is used to optimize the design, but this process comes with drawbacks, such as area and delay aren't the best

possible. Custom design (or semi-custom) design approach can be used to counter this drawback at the cost of time and efforts. That approach has been used in this thesis too. The idea of VLSI optimization stems from past research related register files. The register files have been heavily optimized and plenty of research has been done in improving them using advanced VLSI Optimization techniques [1]. These VLSI Optimization techniques have been discussed in the next sub-section and these techniques will be used later in the contribution chapters (CHAPTER 3, CHAPTER 4, and CHAPTER 6) to improve the SSIA design.

2.2.2 VLSI Optimization Techniques

- Full-custom Design
- Multi-finger MOSFET
- Dynamic Logic
- Domino Logic
- Power-gating
- MOS Current Mode Logic (MCML)

In this sub-section, above VLSI optimization techniques have been covered. Please note that the above list isn't an exhaustive list, because of the broad nature of the topic, there are other ways and other techniques too [43].

Full-custom Design: Full custom design requires a designer to spend a lot of time making sure the area of the layout is most compact ensuring the smaller size of the final design. The design must reflect the higher performance too. The full-custom design will significantly raise the *Design Expenses* component of below equation (**Eq. 2-1**) but the hope is that design will be very good that there will be demand for the product raising the

Production Volume component of the **Eq. 2-1** and thereby compensating the extra cost [44]. The components (such as inverter, tri-state buffer, and multiplexer) required in the design are available off-the-shelf from Standard Cell Library in most cases, as it makes the business sense. A typical profit equation for IC manufacturing is shown below. Earlier the product is released more is the profit. Delay in rolling out product can significantly affect the profit so it makes sense to not spend a large amount of time in doing full-custom design. But there has been a lot of research out there which compares the standard cell and full custom design approach and confirms the advantage and cost mentioned above [45].

$$\begin{bmatrix} Total\ Cost \\ of\ an\ IC \end{bmatrix} = \frac{Design\ Expenses}{Production\ Volume} + \begin{bmatrix} Manufacturing \\ Cost\ per\ IC \end{bmatrix} \qquad \textbf{Eq. 2-1}$$

Multi-finger MOSFET: The change in an aspect ratio of the transistor can have a significant impact on the performance [46]. Another important parameter of the transistor is finger-width and it has been proven that this parameter also has a significant impact on the performance [47]. Folding transistor in multiple-fingers (such as 2, 3 and 4) is beneficial. More numbers of fingers mean more number of channels for current and thus more drive strength. As the channels are in parallel, the channel resistance also becomes parallel, thereby reducing the equivalent resistance. But this comes at the cost of a large area. In this thesis, we have explored the impact of multi-finger MOSFET in CHAPTER 4.

Dynamic Logic: Dynamic logic offers a fast and power efficient alternative to commonly used static logic. Static logic has both pull up and pull down networks in the digital circuit. Dynamic logic uses a pre-charging transistor instead of the whole pull up

the network. This way the output of the digital circuit is always high. If inputs are so that output needs to be low then pull down network pulls down the output. This way dynamic digital circuit overcome the slow rising time to generate high output and saves the constant static power consumption between the power supplies. Pre-charging is also a similar technique which uses a transistor to pre-charge the output. Like pre-charging, there can also be a pre-discharging digital circuit which sets the default output low.

Domino Logic: Static CMOS circuits don't require a special circuit to cascade the different logic gates, but dynamic CMOS circuits require domino logic. Without the use of domino logic circuitry, outputs of cascaded dynamic CMOS circuit fails and falls like cascaded dominos. Thus it is called domino logic.

Power Gating: Power gating is a technique of turning off the inactive parts of the circuit. This reduces the leakage current and unnecessary power consumption of the circuit. There is a technique similar to this that is called *Clock Gating* that too saves the power by pruning the clock.

MOS Current Mode Logic (MCML): MCML is another kind of logic style, as opposed to CMOS logic style. This logic style boasts of low noise and greater noise immunity. Current mode style can achieve better speed without the high cost of the area as well. Also, the power dissipation is superior compared to CMOS at the high operating speed [48]. Many researchers have compared both of the logic styles with other logic styles to prove the above benefits [49] [50] [51]. Further, various techniques that optimize logic gates especially inverter and tri-state buffer have already been published [52] [53] [54] [55]. There are other similar techniques such as pseudo-NMOS, Source-

Coupled Logic (SCL), Positive Feedback SCL (PFSCL) etc. which can achieve similar benefits.

2.2.3 Summary

This section provided a review of VLSI Optimization techniques which have been used in the thesis, starting with Full-custom and multi-finger MOSFET which is the focus of Chapter 3 and Chapter 4. This review was followed by a brief explanation of Dynamic, Domino and Current Mode logic techniques. The explanation has been brief as it wasn't the main focus of thesis and it is just for Chapter 6 which is to evaluate future opportunities for advanced VLSI design techniques for SSIA.

2.3 Summary

Section 2.1.4, 2.1.5 covered research related to stack processor happening at York. VLSI Optimization for Stack Processor seems like an area which is unexplored and can add another dimension (after compilers for Stack Processor, and ILP in Stack Processors). This was the major driving force to choose VLSI Optimization for Stack Processor. The previously done investigation of SSIA was done using standard cell design techniques and using 90nm technology node, so the clear progression from this seemed to be a full-custom design approach [45]. As a full-custom design approach is a time and labour intensive, the semi-custom design approach has been used [44] [47]. Inverter and tri-state buffers are the fundamental components of SSIA and there is plenty of research out there that deals with making inverter and tri-state buffers better [56] [49] [52] [53] [54] [57]. The first step for us seemed to evaluate different aspect ratios [46] and multi-finger MOSFET implementations [58]. This has been investigated and results have been presented in CHAPTER 3 and CHAPTER 4, please note as advanced

technology node (65nm) was also available at the start of research hence the 65nm was used for experiments. CHAPTER 6 dives deeper and evaluated more advanced VLSI Optimization techniques for inverter and tri-state buffer reviewed in the previous section [59] [60] [61] [62]. The main benefit of this work is that this work will establish the benchmark for SSIA performance and implementation better that benchmark will also be established. This will benefit all the application which has the potential to use SSIA such as future stack processor (e.g. NOMAD) [14]. This is the end of the literature review chapter and next chapters present the contributions made by this research.

CHAPTER 3

STANDARD CELL IMPLEMENTATION OF SSIA

3.1 Introduction

This chapter starts with an explanation of the experimental setup and methodologies that are going to be used throughout the thesis. The chapter is intended to evaluate the performance of the fundamental components and multiplexers of Superscalar Stack Issue Array (SSIA). As the SSIA components have varying fanout condition the performance evaluation is done for fanout ranging from 1 to 8. An important rationale for performing this initial work is to get a baseline data of SSIA the performance, remembering that the research question (mentioned in section 1.4.4) is to find out whether the performance of SSIA can be improved. The fundamental components and multiplexers designed in this chapter have been put together in a single cell library, and the library has been referred to as a 'Standard Cell' library throughout the thesis.

Contributions made through this chapter are presented in the list below.

- Standard Cell library - created in 65nm CMOS process technology.
- The fundamental components have been quantified in terms of performance parameters such as area, propagation delay, power, and capacitance.
- The impact of fanout has been evaluated.

- Multiplexers have been designed using the fundamental components and they have also been quantified in terms of performance parameters such as area and delay.
- Multiplexers have been simulated using an alternative methodology and the results from both methodologies have been compared.
- First order projection of SSIA performance, designed using Standard Cell library, has been presented.

3.2 Experimental Setup and Methodology

This section covers the design (of fundamental components and multiplexers circuits and experiments), simulation, associated experimental setup and methodologies employed to calculate the performance parameters. It is done using examples of an inverter, tri-state buffer and 2-input (2X1) multiplexer. All the circuit implementations are done using the 65nm technology of TSMC[4] and Cadence Virtuoso. The simulations are done in Analog Design Environment (ADE) of Cadence.

[4] TSMC – Taiwan Semiconductor Manufacturing Company

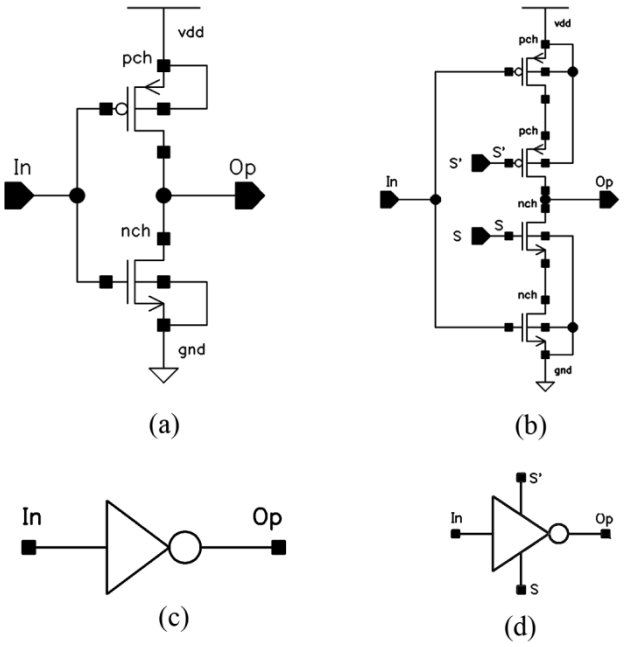


Figure 3-1: Fundamental Components (a) Inverter schematic (b) Tristate buffer schematic (c) Inverter symbol (d) Tristate buffer symbol.

3.2.1 Fundamental Components

Inverter and tri-state buffer are designed using *Complementary MOSFET (CMOS)* technology. CMOS contains both NMOS and PMOS transistors which are arranged in a complementary fashion. The size of PMOS is twice that of NMOS to ensure equal rise time and fall time. Another important parameter of MOSFETs is finger width [58]. For Standard Cell library finger width has been kept at one. In later chapters, when we use the custom approach, size and finger width have been changed. **Figure 3-1** shows the CMOS implementation of inverter and tri-state buffer. CMOS implementation of inverter contains one PMOS and one NMOS transistors and CMOS implementation of the tri-state buffer contains two PMOS and two NMOS transistors. PMOS is labelled '*pch*' and

NMOS is labelled '*nch*'. The input pin is denoted as '*In*' and Output pin is denoted as '*Op*'. The tri-state buffer contains select signal '*S*'. '*S"* should always be opposite of S to achieve the functionality of tri-state buffer. **Figure 3-1** also contains symbols of the fundamental components.

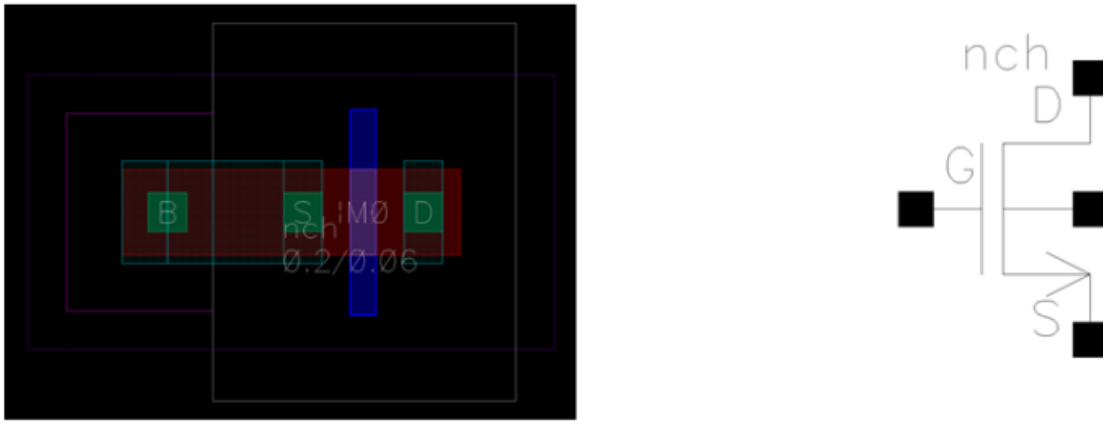


Figure 3-2: Layout and symbol of a transistor side-by-side.

Schematics are enough for simulation and quantifying the performance trend, however, the layout allows us to get more precise simulation data and hence layout simulation data is also obtained. The more realistic output is due to the fact that it takes the parasitic capacitance into consideration. Additionally, layout design allows us to customise the power rail and pin placements as per our own requirements. **Figure 3-2** shows layout and symbol of a MOSFET (NMOS to be specific). Here, B means Bulk Substrate, S means Source, D means Drain, and G means Gate. **Figure 3-3** shows the layout drawings of inverter and Tristate Buffer. Layouts presented in the thesis have passed the Design Rule Check (DRC) and Layout Versus Schematic (LVS) checks. It can be noted in the below figure that pitch of inverter is larger. It is kept intentionally large so

that its height (cadence refers this as a width) matches to that of a tri-state buffer, to keep the fundamental components modular. All of the components follow the same power rail regime (i.e. same width/height). Pins are also placed accordingly so that it's easier to connect other components. Its advantages will become clear when layouts of the more complex component are presented later (such as **Figure 3-21** in section 3.4).

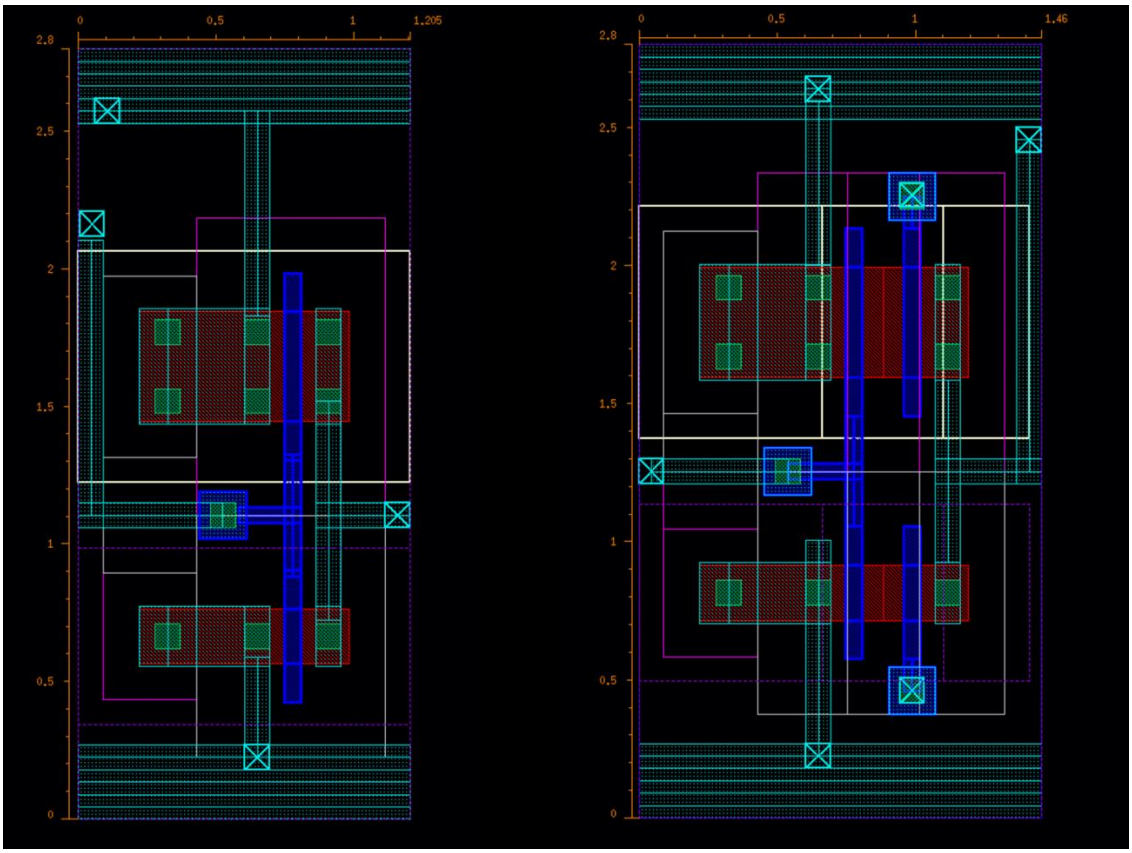


Figure 3-3: Layouts of fundamental components (Left layout – inverter; Right layout – tristate buffer) of SSIA (with rulers).

Figure 3-4 shows a typical test bench used in the experiments. It can be seen that, initially, the ideal signal is passed through an inverter. This is to ensure that a realistic slew rate can be achieved. The UUT in the test bench means the 'Unit Under Test'. For fundamental components, UUT can be either inverter or a tri-state buffer, and for

multiplexers, UUT can be any multiplexers from 2-input mux (2X1) to 8-input mux (8X1). But only 3X1, 4X1 and 5X1 are most important as SSIA uses only those multiplexers, as explained earlier in section 2.1.5. UUTs can have any number of fanout, ranging from 1 to 8. In this example, UUT is an inverter and fanout is four (FO4). In some cases, the tri-state buffer is used for fanout to simulate condition close to reality.

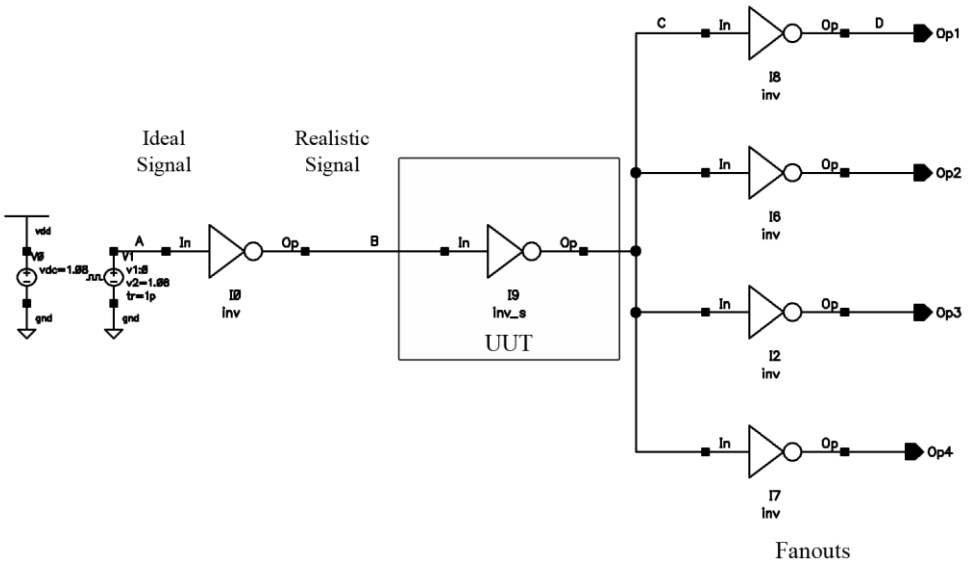


Figure 3-4: A typical test bench which has an inverter as UUT and has a fanout of four.

The performance of fundamental components is quantified using the following four parameters and this section explains how their measurement is done.

- Area – Layout/chip area,
- Delay - Propagation Delay
- Power - Power Consumption
- Capacitance – Capacitance at Input or Output node of the device.

Area

Area of the component has been measured from their layout. The length and width of the component have been calculated using 'ruler' functionality of Cadence Virtuoso and area is the product of both. The length and width of inverter and tri-state buffer can be seen in **Figure 3-3**. Area data is presented in **Table 3-1**. The data in the table has rounded off to make it presentable and the same practice has been followed throughout the thesis.

Table 3-1: Length, width and area of fundamental components inverter and tristate buffer.

Components (↓)	Length (um)	Width (um)	Area (um^2)
Inverter	2.8	1.2	3.4
Tristate	2.8	1.5	4.2

Delay

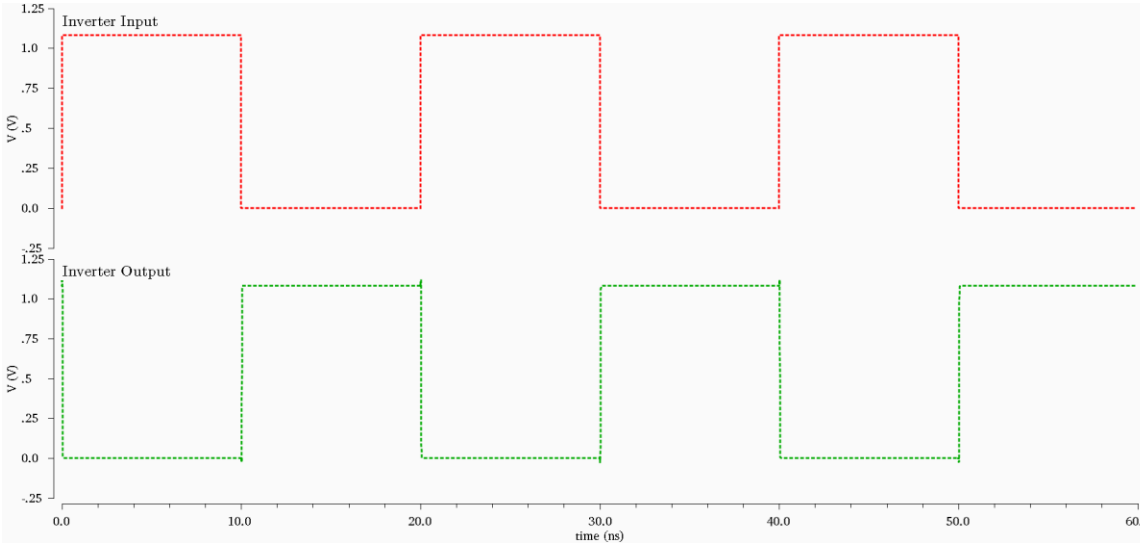


Figure 3-5: Output waveforms of an inverter when supplied with a square wave signal as input.

Propagation delay (t_p) has been measured using the waveforms generated from the simulation of the component. **Figure 3-5** shows a typical input and output waveforms. Propagation delay is clearly visible when Input and Output waveforms are superimposed, as shown in **Figure 3-6** below. We measure the delay at 50% of the V_{high} of the input signal and the overall propagation delay of the component is calculated using **Eq. 3-1**, where t_{HL} is the delay during high to low transition and t_{LH} is the delay during low to high transition. It can also be seen in **Figure 3-6** that delay of layout simulation is higher compared to that of the schematic, this is due to the presence of parasitic capacitance. **Table 3-2** shows the data of example that we have considered just now.

$$t_p = \frac{t_{HL} + t_{LH}}{2} \qquad \textbf{Eq. 3-1}$$

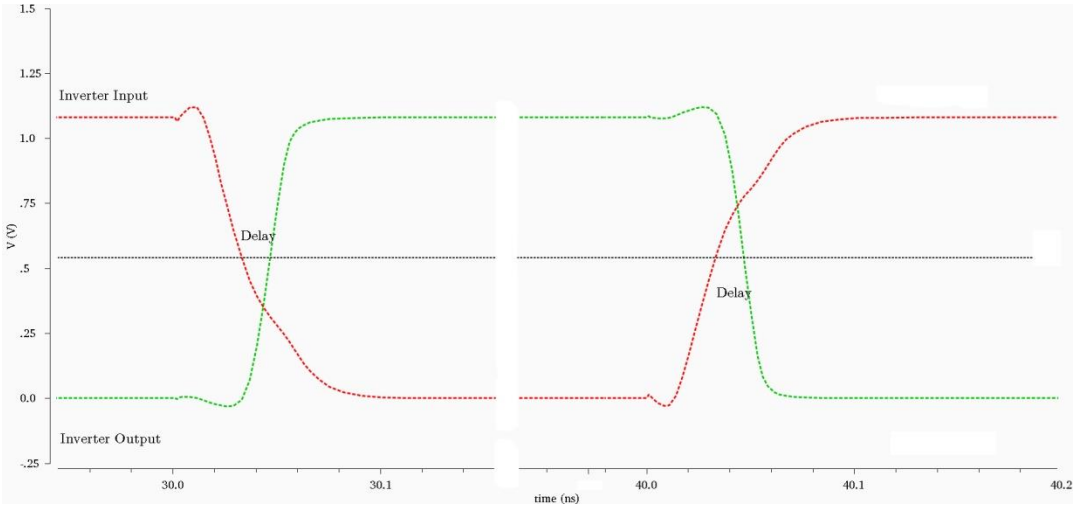


Figure 3-6: Superimposed input and output waveform of an inverter showing propagation delay.

Table 3-2: Inverter delay data.

Component (Fanout)	Schematic	Layout
Inverter (FO4)	23.7 ps	43.8 ps

Capacitance

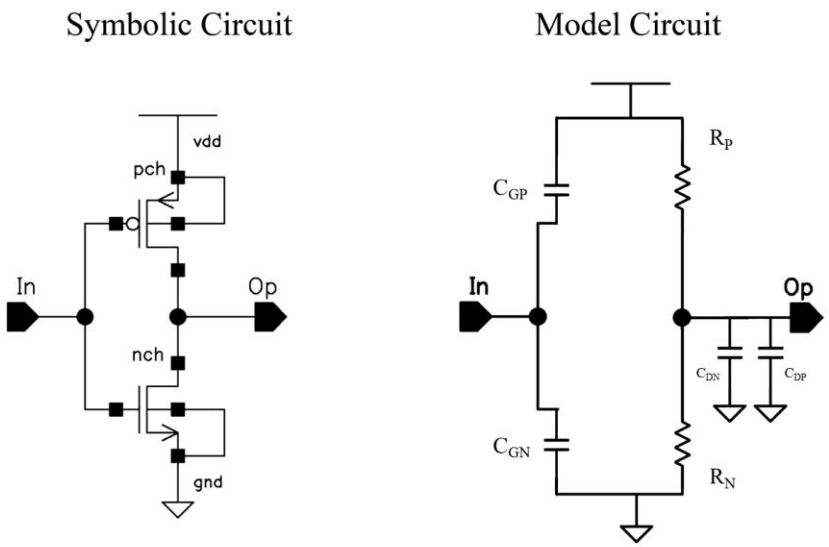


Figure 3-7: Equivalent Resistance –Capacitance model for an inverter.

The propagation delay is mainly due to the time spent on charging and discharging the parasitic capacitance. An inverter can be seen as an RC circuit of parasitic capacitance and resistance as shown in **Figure 3-7**. Transistor has various kinds of parasitic capacitance as shown in **Figure 3-7** and **Figure 3-8**. Gate Capacitance (C_G) is between the gate and the underlying channel. Source/Drain Capacitance (C_D) is between Source and Drain. C_{GN} and C_{GP} are Gate Capacitance associated with NMOS and PMOS

respectively. Similarly, C_{DN} and C_{DP} are Source/Drain Capacitance associated with NMOS and PMOS respectively. C_{Load} is an equivalent capacitance at the node

ADE simulation calculates the load capacitance C_{Load} when 'captab' option of Cadence Virtuoso is enabled. To understand our capacitance calculation, consider a typical test bench of **Figure 3-4**. We calculate C_{Load} at Node B and Node C. Node B Capacitance is represented as an Input Capacitance of UUT and Node C Capacitance is Output Capacitance for simplicity. The parasitic capacitance of MOSFETs has both linear and non-linear components. The non-linear component is the direct result of depletion regions in the MOSFETs. As the depletion region of MOSFETs is dependent on applied V_{ds} the region itself is variable and hence the parasitic capacitance is also variable [63]. Cadence also generates the individual components (Fixed and Variable) of C_{Load} and they are presented in **Table 3-3**. The values are in femtofarad.

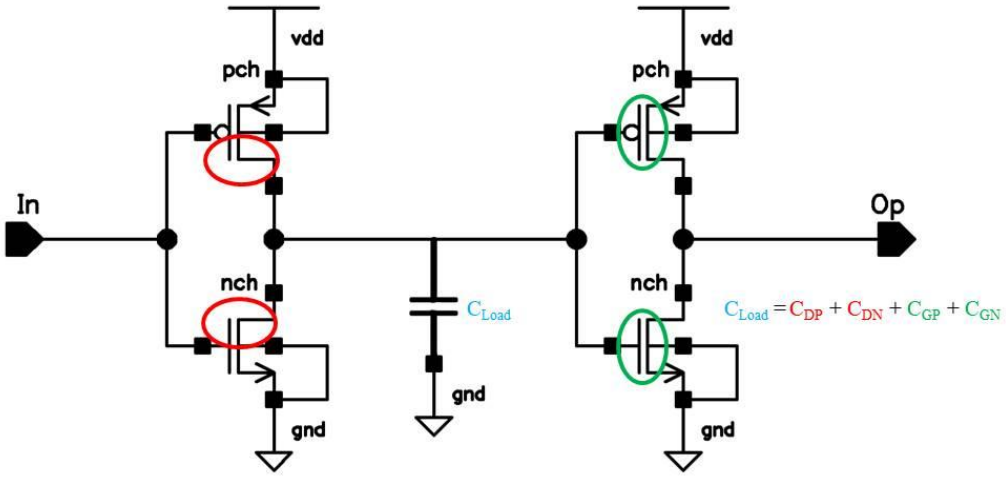


Figure 3-8: The capacitance calculation for two connected inverters based on the equivalent model in Figure 3-7.

Table 3-3: Fixed and variable, input and output capacitance of schematic and layout of inverter in femtofarad.

Inverter	Schematic			Layout (fF)		
	Fixed	Variable	Total	Fixed	Variable	Total
Input (fF)	0.2	0.2	0.4	0.8	0.2	1.0
Output (fF)	1.0	1.1	2.1	2.7	1.1	3.8

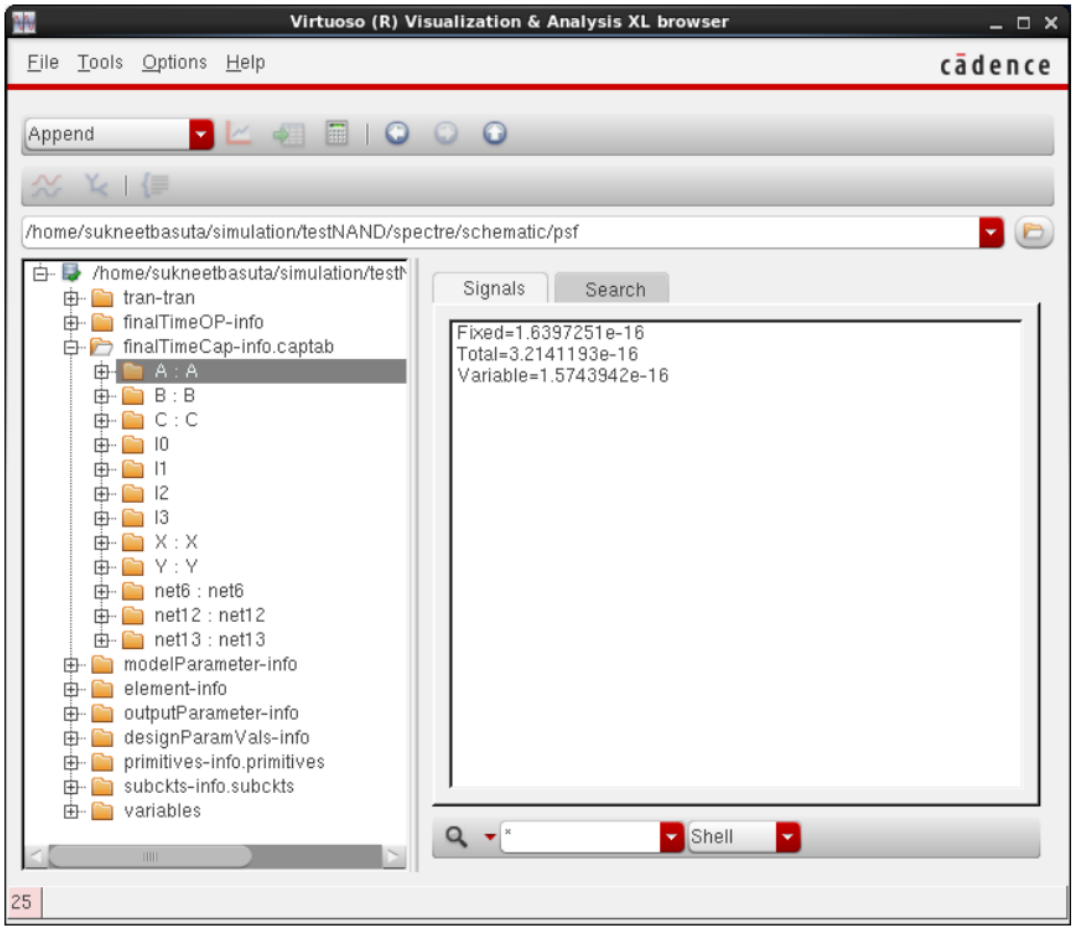


Figure 3-9: Screenshot of capacitance calculation using Cadence Virtuoso [64].

Power

The process of calculating power consumption is similar to that of Capacitance. The power consumption data is also generated during ADE simulation and has been noted from there (See **Figure 3-10**). The power consumption of UUT is dependent on its output state as well as the output transition. We have measured two kinds of power consumption data - static power consumption and the peak value of dynamic power consumption. This is presented in **Table 3-4**.

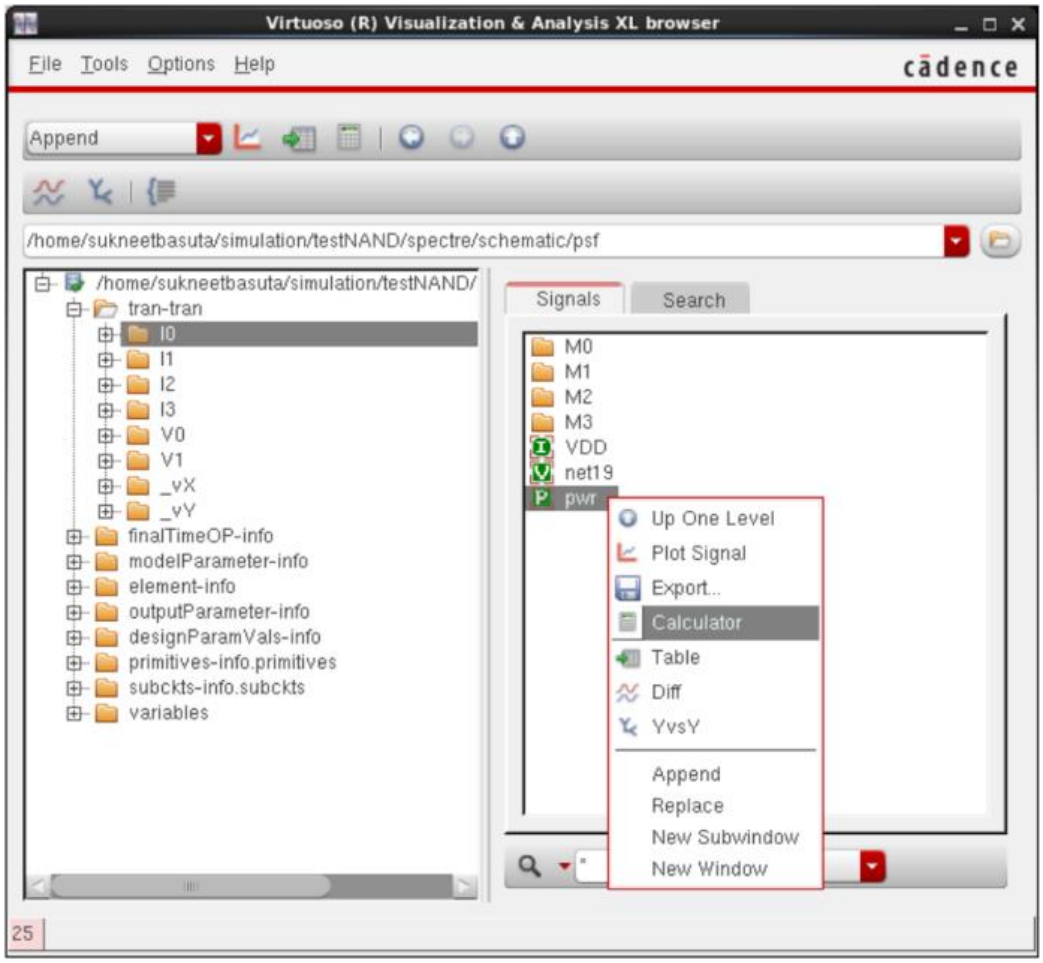


Figure 3-10: Screenshot of power calculation using Cadence Virtuoso [64].

In the table, four different values of power consumption have been presented. The first row is for the peak value of power consumption when input value transitions from low to high. This value comes out to be 59.5nW. The second row is for the peak value of power consumption when input value transition in the other direction and this value is slightly higher at 59.6nW. The next two rows are for static power consumption that means when the input to the inverter is constant i.e either GND (ground) or VDD (Source Voltage). More on power consumption has been discussed later in section 5.2.3. Their dynamic power calculation is presented in terms of energy consumed during a transition.

Table 3-4: Static and peak dynamic power consumption of inverter in nW.

Type of Power Consumption	**Value (nW)**
Peak Dynamic Power - I/P = Low → High	59.5
Peak Dynamic Power - I/P = High → Low	59.6
Static Power - I/P = High	1.2
Static Power - I/P = Low	1.0

3.2.2 Multiplexer

The remainder of this section explains the methodologies used for multiplexer using an example of the 2X1 multiplexer. The approach of designing multiplexers which uses tri-state buffers and inverter has been explained earlier in section 2.1.5 of the literature review chapter.

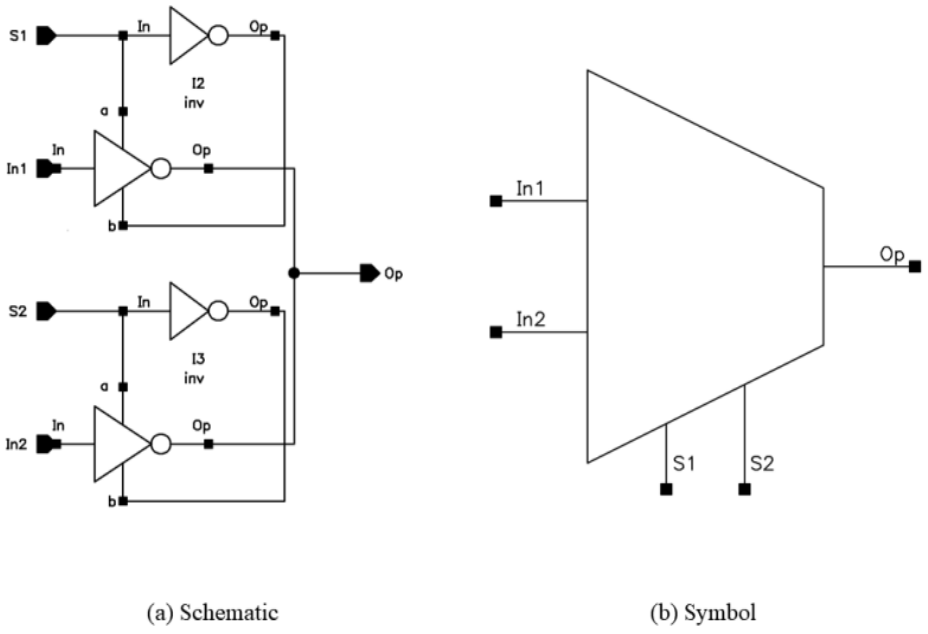


Figure 3-11: (a) Schematic for 2g (b) Symbol of 2X1 multiplexer

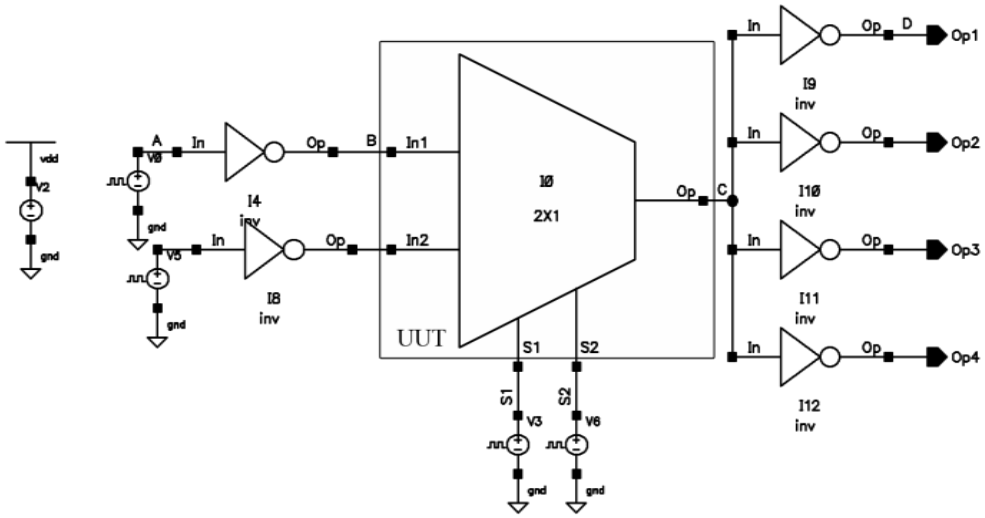


Figure 3-12: A typical test bench which has 2X1 multiplexer as UUT and has a fanout of 4.

Figure 3-11 shows schematic and symbol of the 2X1 multiplexer. In the schematic of 2X1 multiplexers, there are two select signals in each tri-state buffer. Those two signals are just the opposite of each other. Hence, ideally, there is only one unique select signal per tri-state. Test benches for multiplexer experiments, shown in **Figure 3-12**, remain similar to the previously explained typical test bench (**Figure 3-4**) but now the UUT is 2X1 multiplexer instead of the fundamental component. The performance of multiplexer is quantified using two parameters – Area and Delay.

Area

A multiplexer is made up of fundamental components. It contains tri-state buffers - equivalent to its's number of inputs (n) - and an inverter. In our approach, when it comes to layout, all the components are aligned horizontally and hence the length of the multiplexer is the length of any single component (all components have equal length) and width will be the addition of widths of each component as shown below in **Figure 3-13** and **Eq. 3-2**.

$$Multiplexer_{Width} = n \cdot tristate_{Width} + inverter_{Width} \qquad \textbf{Eq. 3-2}$$

Area (length X width) calculated using this first order projection for the 2X1 multiplexer is presented below in **Table 3-5**. Please note that the actual area would be slightly larger because of the area required for wiring them together. If that extra space is not left than it results in Design Rule Violation errors. The same conditions apply for the area data presented later in CHAPTER 4, and CHAPTER 5.

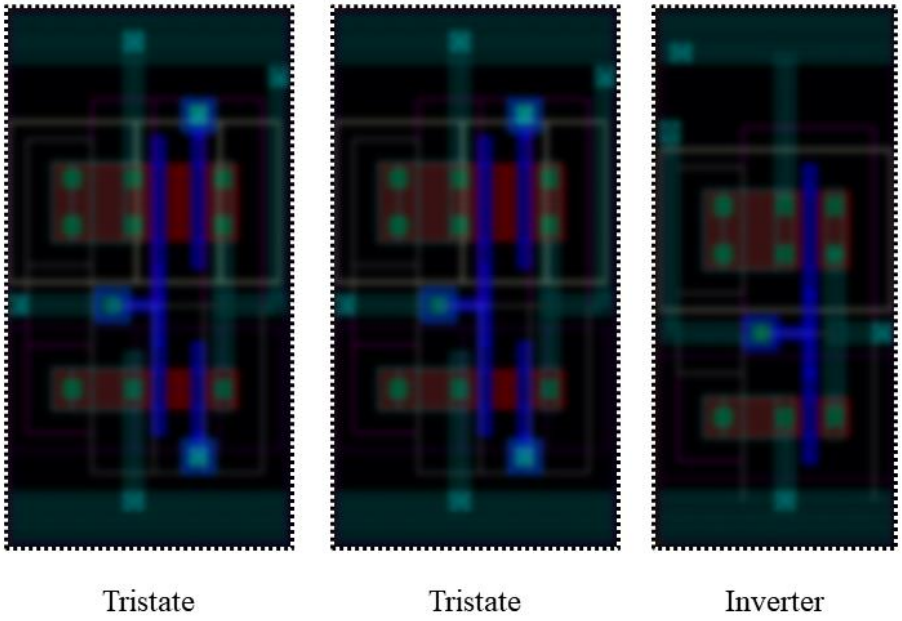


Figure 3-13: A simple box diagram if the layout of the 2X1 multiplexer is to be produced.

Table 3-5: Length, width and area of a 2X1 multiplexer.

Component	Length (um)	Width (um)	Area (um^2)
2X1	2.8	4.1	11.5

Delay

The propagation delay of the multiplexer is the total delay occurred when a signal travels from node B to node C in a test bench similar to a one shown in **Figure 3-12**. The process of obtaining schematic data is similar to that of the fundamental components. There are two possible ways of collecting the post-layout simulation data. One involves doing the full layout similar to what is done for fundamental components, this is a very

time intensive process and hence the other approach has been used. In this approach, Cadence Virtuoso settings are changed. Due to the change Cadence Virtuoso considers the layout of fundamental components for simulation. But this approach ignores the wiring, the wiring that connects the tri-state buffers to the inverter. At this stage, the impact of wiring is ignored. This is to allow many permutations of the circuit to be designed without having to do a full layout for every case – which is a labour intensive task as mentioned earlier. But to gauge how significant this simplification is, with respect to the accuracy, a further set of experiments has been performed, these are presented in section 3.4 of the chapter.

Table 3-6: Propagation delay data for the 2X1 multiplexer which has a fanout of four.

Component	**Delay**
2X1 FO4	101 ps

3.2.3 Design Decisions

In this sub-section list of explanation has been provided for some of the key design decision and pros and cons have been discussed wherever necessary.

- CMOS logic technology – In this thesis, we have used CMOS technology to construct our components. Alternatives are BJT, TTL or NMOS based technology. It has been proven that CMOS has high noise immunity and low power consumption. Although there are more transistors one of the pair is always off.
- Multiplexer implementation using Tri-state buffer – There are many ways of designing multiplexers, e.g. 5X1 multiplexers can be designed using one 3X1 and

two 2X1 multiplexers as well. But previous research investigated that it's better to design multiplexers using tri-state buffers [1].

- The pitch of inverter deliberately kept longer – It is visible in **Figure 3-3** that layout of the inverter is not as compact as a tristate buffer. This is deliberately done. Because this will allow the future design of multiplexers easier if their pitch is the same, as shown in a box diagram of 2X1 in the previous section (3.2.2) in **Figure 3-13**.
- Testbench Design – One of the choices we have made to obtain the data in the thesis is to not use the ideal signal. This has been done by passing the input signal through an inverter initially. This can be seen in all the testbenches in this thesis, e.g. as shown in **Figure 3-4**.
- Layout simulation – Schematic as well as layout both the design have been simulated and both the data has been presented. There are an advantage and disadvantage of using this approach.
    - Pros
        - Simulation results are more realistic.
        - Impact of parasitic capacitance can be observed
    - Cons
        - This approach is time intensive due to the time required to draw layouts.
        - Although data is more realistic, the actual value can still be a little bit different as layout models are not 100% accurate.

3.3 Fundamental Components

A simple evaluation of fundamental components (as well as a multiplexer) of SSIA has been done in the past [1]. It was done using the 90nm technology of UMC, but that evaluation was limited to only propagation delay, and that too just for schematics and considered only very simple fanout scenario. In this thesis, advanced technology has been used and fanout ranging from 1 to 8 have been considered. Additionally, the test benches have been designed carefully to be more precise about the evaluation and both schematic and layout measurements have been done. In this section, each parameter of performance has been given their own separate sub-section and their contribution & significance have been discussed.

3.3.1 Area

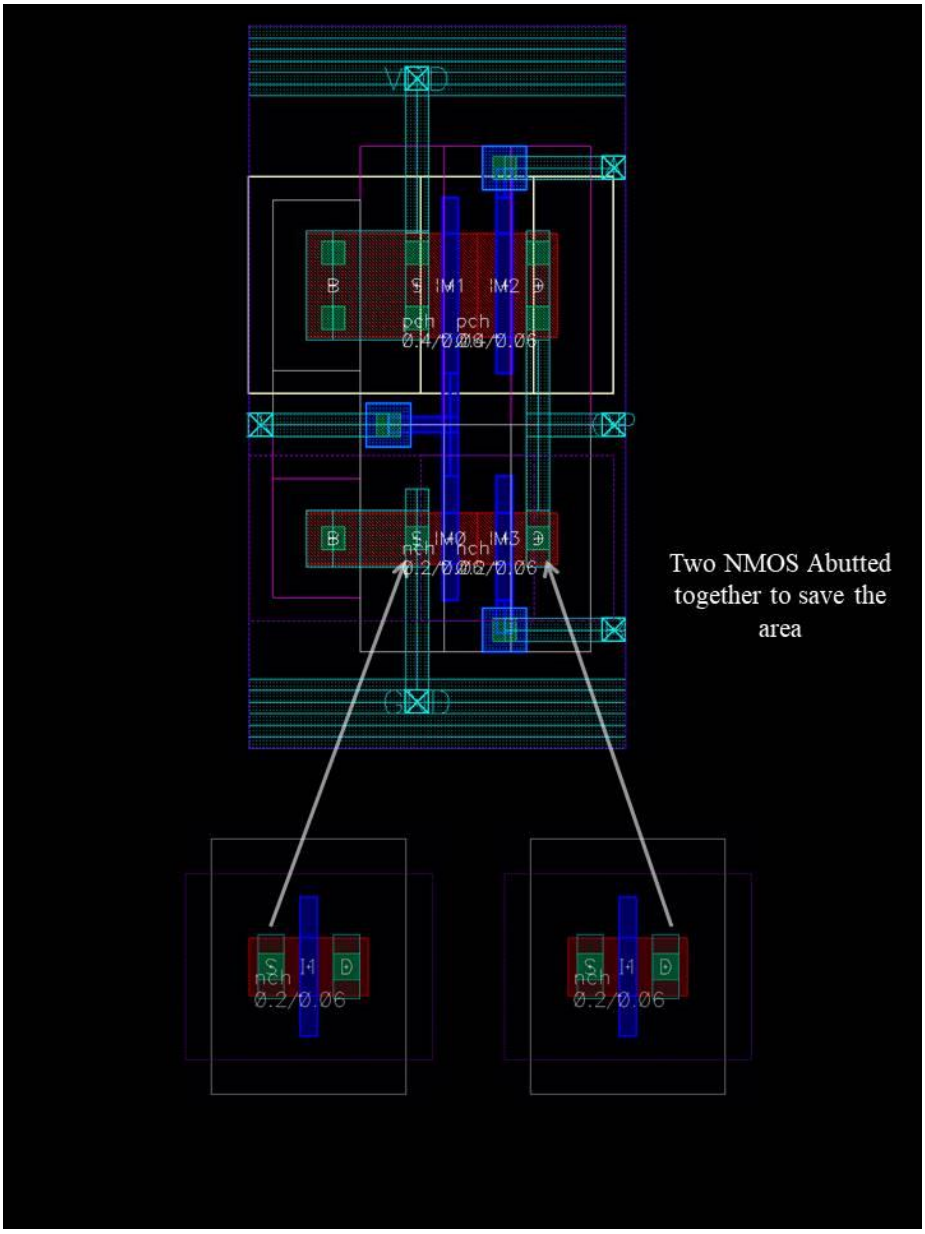


Figure 3-14: Picture showing the layout of two distinct transistors and their layout when abutted.

The first parameter of performance that we have considered is the area that a fundamental component will occupy on the chip. Areas of both fundamental components – inverter and tri-state buffer – have already been presented in section 3.2. It is presented

again in **Table 3-7** below. Although optimizations are presented in later chapters, one simple optimization has been done even for 'Standard Cell' design – Abutting [65]. Abutting can minimize the area. In our case, two MOSFETs of tri-state buffers are abutted as shown in **Figure 3-14** below. Cadence Virtuoso has an in-built abutting capability and that has been used. It can be seen that the area of tri-state has been reduced significantly by the use of the abutting technique.

Table 3-7: Length, width and area of fundamental components and abutted tristate buffer.

Components (↓)	**Length (um)**	**Width (um)**	**Area (um^2)**
Inverter	2.8	1.2	3.4
Non- abutted tri-state buffer	2.8	2.3	6.4
Abutted tri-state buffer	2.8	1.5	4.1

3.3.2 Delay

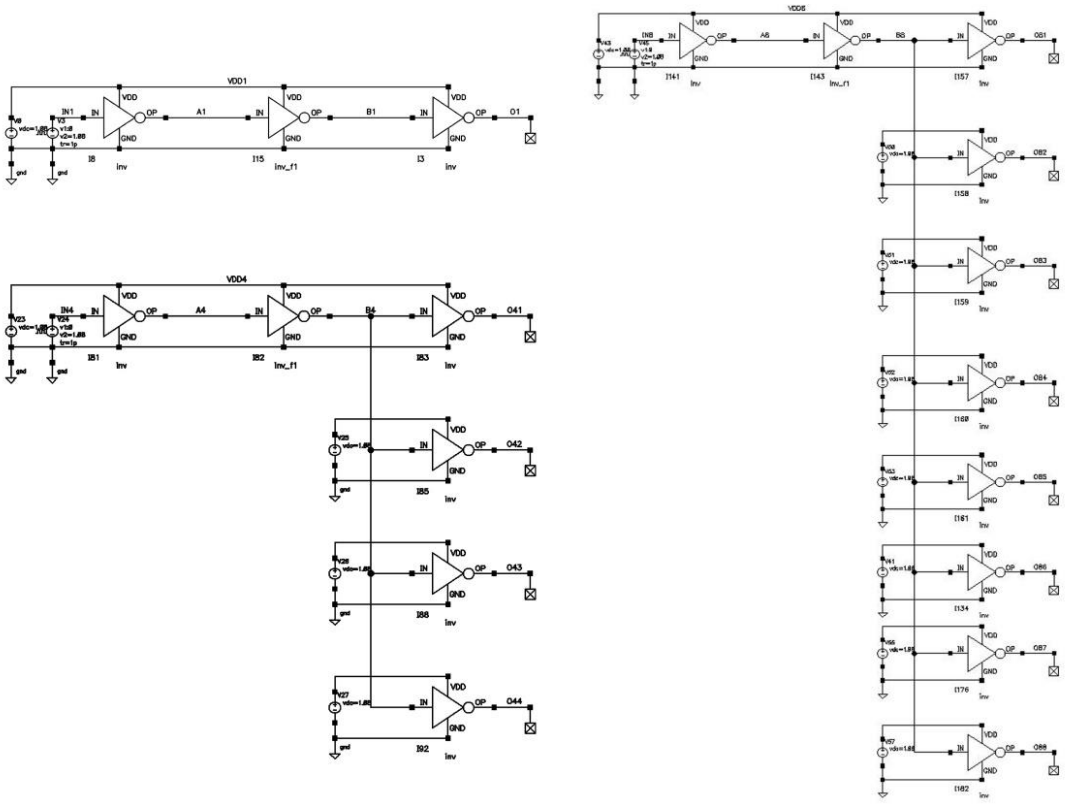


Figure 3-15: Test benches for fanout of 1, 4 and 8 commonly used in experiments.

The second parameter of performance is the propagation delay. Fundamental components having fanout ranging from 1 to 8 have been simulated. **Figure 3-15** shows some of these test benches. Delay data for fanout ranging from 1 to 8 has been collected presented in **Table 3-8**. This data has been plotted, and the resulting graph has been shown in **Figure 3-16**. In the table, it can be seen that as the fanout increases propagation delay also increases, and the graph suggests propagation delay shares a linear relationship with fanout. Looking at schematic data and layout data it can be observed that delay of 'FO4 schematic' is equivalent to that of 'FO1 layout' proving the point that actual layout

data is quite far from that of schematic data and signifies the importance of post-layout simulation data.

Table 3-8: Schematic and layout propagation delay data of an inverter for fanout ranging from 1 to 8.

Fanouts (↓)	Schematic (ps)	Layout (ps)
FO1	13	23
FO2	17	30
FO3	20	37
FO4	24	44
FO5	27	51
FO6	31	57
FO7	34	64
FO8	38	71

Table 3-9: Schematic and layout propagation delay data of a tri-state buffer for fanout ranging from 1 to 8.

Fanout (↓)	Schematic (ps)	Layout (ps)
FO1	31	41
FO2	39	54
FO3	46	68
FO4	53	81
FO5	60	94

FO6	67	108
FO7	74	122
FO8	82	136

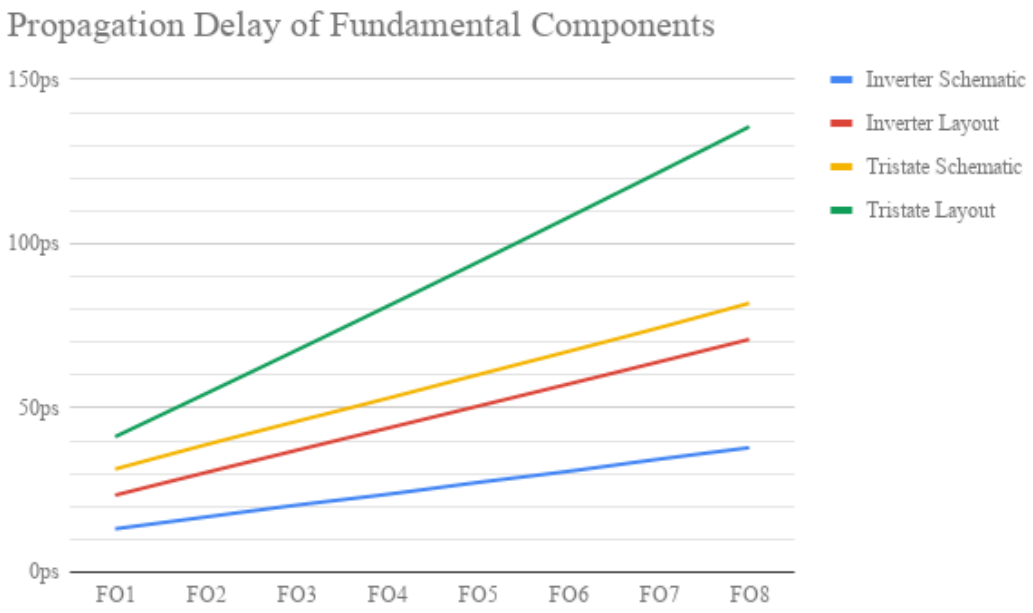


Figure 3-16: Graph showing the trend of propagation delay across various fanouts and schematic, layout for fundamental components.

3.3.3 Capacitance

The third parameter of performance is Capacitance. Input and output capacitance data of inverter and tri-state buffer is presented below in **Table 3-10**. Both components – fixed and variable – of total capacitance have been presented. The table can be seen as an extended version of **Table 3-3**.

Table 3-10: Fixed and variable, input and output capacitance of schematic and layout of inverter for fanout ranting from 0 to 8 in femtofarad.

Fanout (↓)	Schematic			Layout		
	Fixed	**Variable**	**Total**	**Fixed**	**Variable**	**Total**
	Input Capacitance (fF)					
All F/O	0.2	0.2	0.4	0.8	0.2	1.0
Fanout (↓)	**Output Capacitance (fF)**					
	Fixed	**Variable**	**Total**	**Fixed**	**Variable**	**Total**
NOC	0	0	0	0.2	0	0.2
FO1	0.2	0.3	0.5	0.8	0.3	1.1
FO2	0.5	0.5	1.0	1.5	0.5	2
FO3	0.8	0.8	1.6	2.1	0.8	2.9
FO4	1.0	1.1	2.1	2.7	1.1	3.8
FO5	1.3	1.3	2.6	3.4	1.3	4.7
FO6	1.5	1.6	3.1	4.0	1.6	5.6
FO7	1.8	1.9	3.7	4.7	1.9	6.6
FO8	2.0	2.2	4.2	5.3	2.2	7.5

NOC = No Connection and hence zero fanout.

Table 3-11: Fixed and variable, input and output capacitance of schematic and layout of tristate buffer for fanout ranting from 0 to 8 in femtofarad.

Fanout	Schematic			Layout		
	Fixed	**Variable**	**Total**	**Fixed**	**Variable**	**Total**

(↓)	Input Capacitance(fF)					
All F/O	0.2	0.2	0.4	0.7	0.2	0.9
Fanout (↓)	**Output Capacitance(fF)**					
	Fixed	**Variable**	**Total**	**Fixed**	**Variable**	**Total**
NOC	0	0	0	0.2	0	0.2
FO1	0.2	0.3	0.5	0.8	0.2	1.0
FO2	0.5	0.5	1	1.5	0.4	1.9
FO3	0.8	0.7	1.5	2.1	0.7	2.8
FO4	1.0	1.0	2.0	2.7	0.9	3.6
FO5	1.3	1.2	2.5	3.4	1.1	4.5
FO6	1.5	1.5	3.0	4.0	1.3	5.3
FO7	1.8	1.8	3.6	4.6	1.5	6.1
FO8	2.0	2.0	4.0	5.3	1.8	7.1

NOC = No Connection and hence zero fanout.

At first, it was a surprise to see that capacitance of tri-state buffer is slightly lower compared to the inverter. It can be attributed to the presence of an extra transistor in-series in the tri-state buffer. Two equivalent capacitors in series halve the overall value and then it is added to calculate overall C_{Load}. **Figure 3-17** explains this idea further.

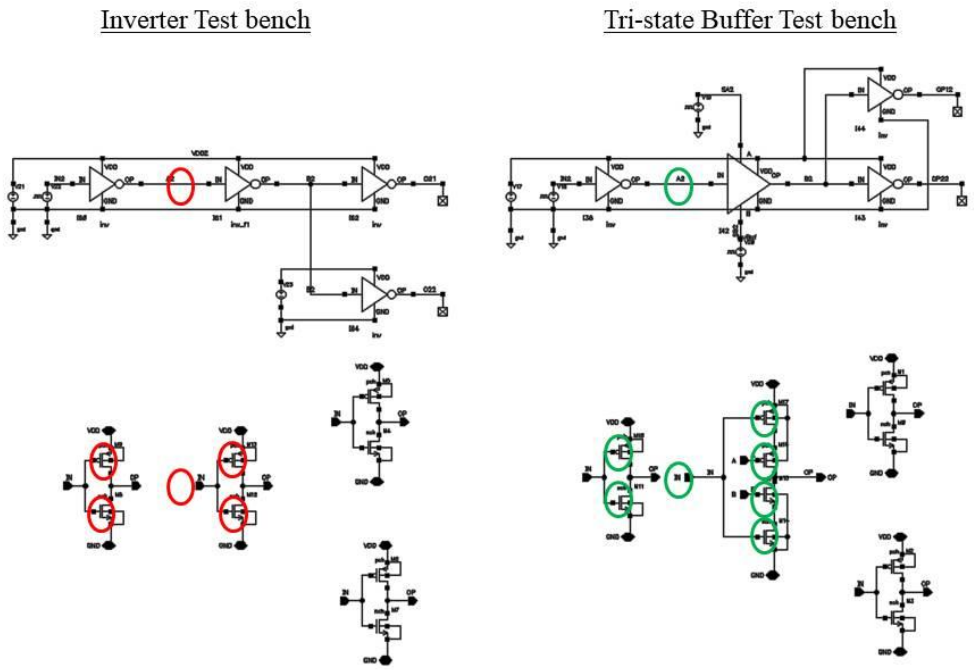


Figure 3-17: Diagram showing a transistor level schematic of the test benches to explain their capacitances at nodes.

3.3.4 Power

Fourth and the last parameter of performance is power consumption. Post-layout power consumption data of inverter and tri-state buffer for fanout ranging from 1 to 8 is presented below. Power consumption is slightly different for high to low and low to high transition and therefore both sets of data are presented below. Power is consumed when there is no activity in the circuit i.e. static power consumption is also presented below.

Table 3-12: Peak dynamic and static power consumption of an inverter in nanowatts for low to high and high to low input transitions.

Peak Value of Dynamic Power		
Fanout (↓)	**I/P = Low → High (nW)**	**I/P = High → Low (nW)**
FO1	47	46
FO2	53	52
FO3	57	56
FO4	59	60
FO5	62	62
FO6	64	64
FO7	66	65
FO8	67	66
Static Power		
Fanout (↓)	**I/P = Low (nW)**	**I/P = High (nW)**
All fanout	1.2	1.0

Table 3-13: Peak dynamic and static power consumption of a tristate buffer in nanowatts for low to high and high to low input transitions.

Peak Value of Dynamic Power		
Fanout (↓)	**I/P = Low → High (nW)**	**I/P = High → Low (nW)**
FO1	36	32
FO2	37	33
FO3	38	34
FO4	38	35
FO5	39	35
FO6	39	36
FO7	40	36
FO8	40	36
Static Power		
Fanout (↓)	**I/P = Low (nW)**	**I/P = High (nW)**
All fanout	3.2	3.2

3.4 Multiplexer

This section presents data of multiplexer and fanout that are relevant for SSIA only and these are 3-input (3X1), 4-input (4X1) and 5-input (5X1) multiplexers. 3X1 has a fanout of 3, 4X1 has a fanout of either 4 or 5, and 5X1 has a fanout of 5. Multiplexers having input ranging from 2 to 8 have been simulated to find out the trend and have been presented in the **Table A-1** of APPENDIX A. Multiplexer methodology has been

explained previously using the example of 2-input (2X1) multiplexer in section 3.2.2. **Figure 3-18** shows schematics and symbols of 3X1, 4X1 and 5X1 Multiplexers.

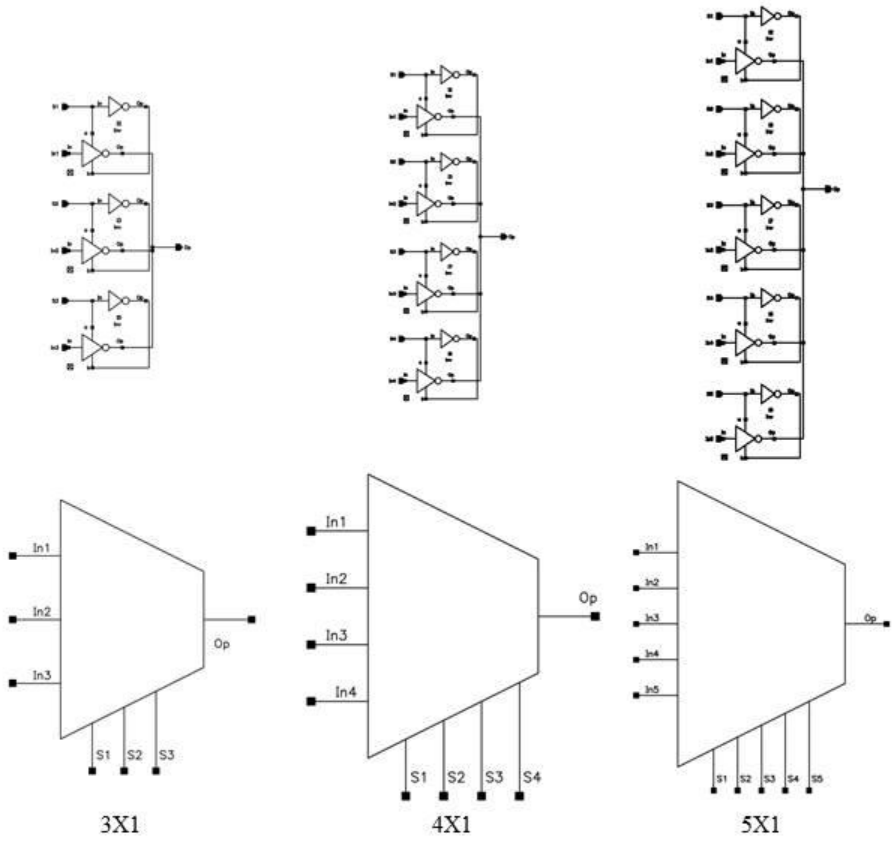


Figure 3-18: 3X1, 4X1 and 5X1 Multiplexer Schematics and Symbols (Left to right).

Like fundamental components, a simple evaluation of multiplexer has already been done in the past [1], but our evaluation is better because it is more in-depth, as mentioned previously in section 3.3. Details of two parameters of performance – area and delay – is presented in following subsections. The alternative methodology for multiplexer simulation, mentioned earlier, has also been explained in this section.

3.4.1 Area

The first parameter of performance that we have considered is the area that a fundamental component will occupy on the chip. The width of multiplexers can be

calculated using the **Eq. 3-2** and then using the methodology explained in section 3.2.2. Areas of 3X1, 4X1 and 5X1 multiplexers have been calculated and are presented below in **Table 3-14**. As mentioned earlier in section 3.2 the area data is slightly optimistic as the impact of internal wiring has been ignored at this stage.

Table 3-14: Calculated length, width and area of multiplexers of SSIA.

Component (↓)	**Length (um)**	**Width (um)**	**Area (um^2)**
3X1	2.8	5.6	15.7
4X1	2.8	7.0	19.7
5X1	2.8	8.5	23.8

3.4.2 Delay

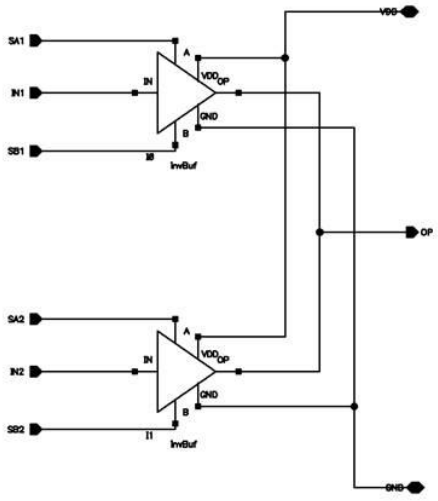
The second parameter of performance is the propagation delay. The propagation delay of the multiplexers used in SSIA is shown below in **Table 3-15**.

Table 3-15: Propagation delay of 3X1, 4X1 and 5X1 of SSIA for relevant fanouts in picoseconds.

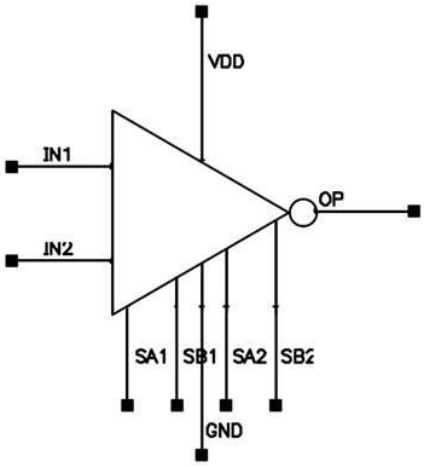
Components – Fanout (↓)	**Delay (ps)**
3X1 – FO3	110
4X1 – FO4	131
4X1 – FO5	137
5X1 – FO5	151

3.4.3 Alternative methodology - for multiplexer simulations

As mentioned previously, our previous simulation methodology for multiplexer ignores the impact of internal wiring. Hence, two new components – 2g and 3g – have been designed by ganging transistors together. The idea is that by ganging tri-state buffer together wires interconnect between cells is introduced. If this wire effect is large, then the delay data will change significantly. If the wire effect is small, then the delay data will not change much (which would suggest that the simplified methodology is acceptable). The schematic, symbol and layout of 2g and 3g are presented in figures below. Pin placement in the layout, shown in **Figure 3-21**, needed the care to avoid design rule violations.



(a) Schematic



(b) Symbol

Figure 3-19: 2g (ganged component) (a) schematic (b) symbol.

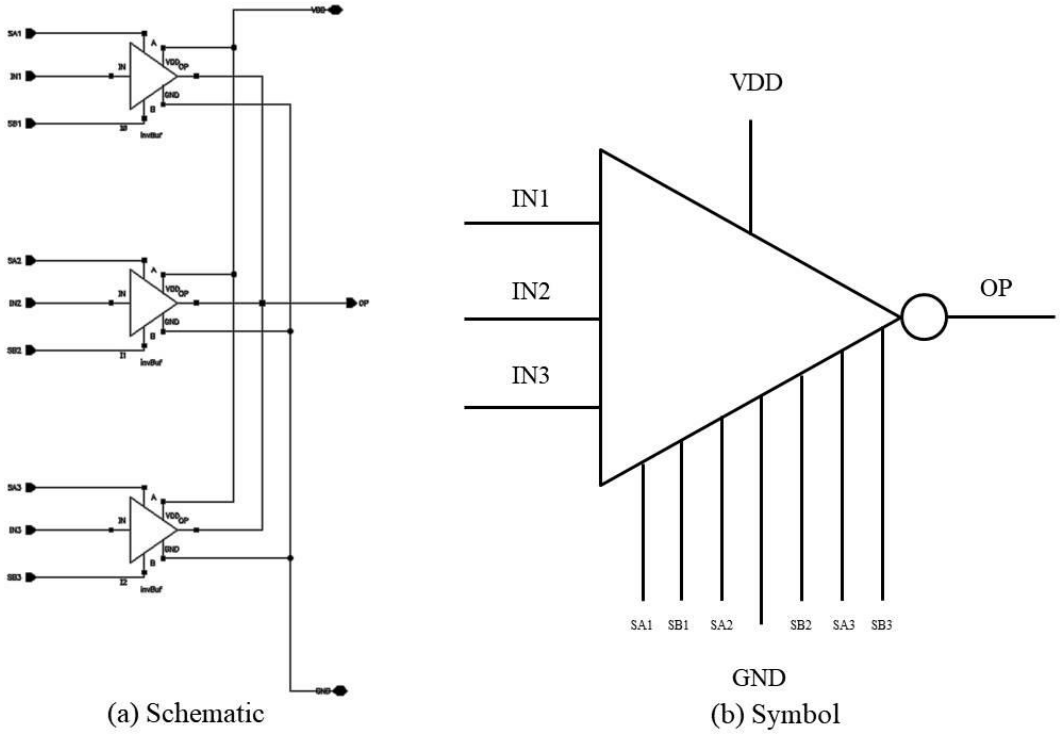


Figure 3-20: : 3g (ganged component) (a) schematic (b) symbol.

Areas of components 2g and 3g are measured from the layout shown in **Figure 3-21** and have been presented in **Table 3-16**. The table also includes the area of 2X1 designed normally (2X1) as well using 2g ($2X1_g$). $2X1_g$ has been simulated in 3 different fanout conditions and delay data measured have been presented in **Table 3-17** below.

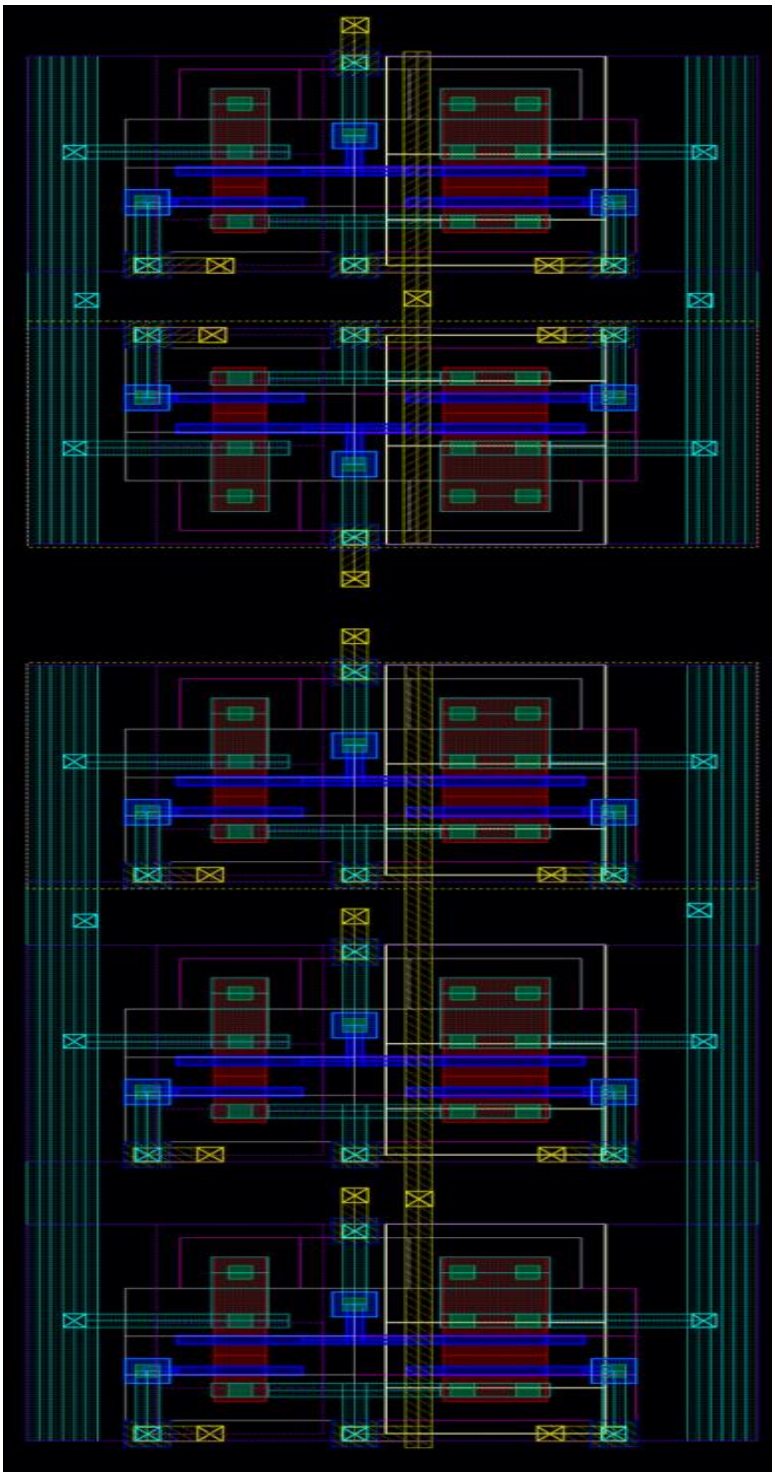


Figure 3-21: Layout of 2g(top) and 3g(bottom) ganged components.

Table 3-16: Calculated length, width and area of ganged components and their respective multiplexers.

Component (↓)	**Length (um)**	**Width (um)**	**Area (um2)**
2g	2.8	3.4	9.5
3g	2.8	5.3	14.8
$2X1_g$	2.8	4.6	12.9
2X1	2.8	4.1	11.5

Table 3-17: Propagation delay of 2X1 and 2X1g for fanouts 1,4 and 7 in picoseconds.

Fanouts (↓)	**$2X1_g$ (ps)**	**2X1 (ps)**
FO1	82	79
FO4	104	101
FO7	122	119

More multiplexers, designed using this approach, were added to *Standard Cell* library. New 3X1 has been designed using 3g and an inverter, new 4X1 has been designed using two 2g cells and an inverter, and new 5X1 has been designed using 2g, 3g and an inverter. Area and delay of these components have been presented in tables below. From this data impact of wiring, the area can also be inferred. It has been calculated that Normal Multiplexer reports 10%-15% smaller area than more realistic Ganged Multiplexer but this is the difference is likely to be less for bulkier multi-finger implementations of the next chapter.

Table 3-18: Length, width and area of multiplexers designed using ganged components with a column for normal values for comparison.

Component (↓)	**Length (um)**	**Width (um)**	**Area (um2)**	**Normal**
$3X1_g$	2.8	6.5	18.2	15.7
$4X1_g$	2.8	8.0	22.4	19.6
$5X1_g$	2.8	9.9	27.7	23.8

Table 3-19: Propagation delay of ganged 3X1, 4X1 and 5X1 of SSIA for relevant fanouts in picoseconds with columns for normal and the difference.

Component - Fanout (↓)	**Normal (ps)**	**Ganged (ps)**	**Difference (ps)**	**Percentage**
$3X1_g$ - FO3	110	116	6	5%
$4X1_g$ - FO4	131	134	3	2%
$4X1_g$ - FO5	137	141	4	3%
$5X1_g$ - FO5	151	159	8	5%

As the observed change is less than 5%, it can be concluded that although not very precise, the normal methodology can be used without being too far from real value.

3.5 Superscalar Stack Issue Array (SSIA)

Although SSIA design is not very complicated, doing full custom layout can be time & labour intensive, and hence first-order projection has been used. By first-order projection we mean the overall performance has been stipulated based on the

performance of individual components. First-order projection won't be far from actual data because the test bench has been designed carefully to be similar to actual condition, e.g. consider issue width 2 of SSIA. All tri-state buffers receive an input from an inverter and output of tri-state is fed to an inverter which has either 4 or 5 tri-state buffer as a fanout. Our test bench contains all the just mentioned components and hence we believe that our prediction will be very close to actual data.

The performance of SSIA has been quantified using two performance parameters: area and delay. Both parameters have their own separate sub-section below where their contribution & significance have been discussed.

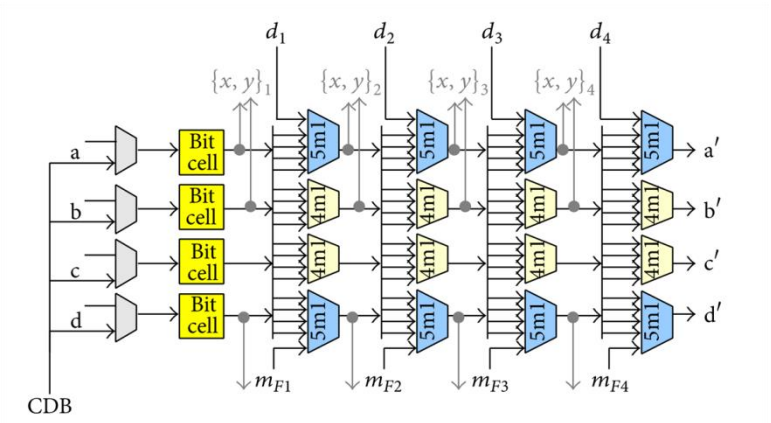


Figure 3-22: Diagram of 4-issue wide and 4-elements deep Superscalar Stack Issue Array (SSIA).

3.5.1 Area

The first parameter of performance that we have considered is the area. Area of the SSIA will be sum total of the area of each individual component, it can be calculated using data from **Table 3-14** and **Eq. 3-3**, and it is mentioned below in **Table 3-20** for various depths of SSIA. There are four columns, or Issue Widths (IW_n = 4), in SSIA and

each issue width contains two 5-input (5X1) multiplexers and two 4-input (4X1) multiplexers. 3-input (3X1) multiplexers aren't shown in **Figure 3-22** but its number will be four less than the depth (D) of the stack. Normal multiplexer data has been used instead of the ganged multiplexer at this stage and also SSIA level abutting can also be employed for reducing the area but that is beyond the scope of this thesis.

$$A = IW_n * [2 * 5X1_{area} + 2 * 4X1_{area} + (D - 4) * 3X1_{area}] \qquad \textbf{Eq. 3-3}$$

Table 3-20: Area of SSIA and ganged SSIA for depths 4 to 8 of the stack.

Depth of Stack	**Area(um^2)**	**$Area_g$ (um^2)**
4	347	401
5	410	474
6	473	546
7	536	619
8	598	692

Please note that the area data of SSIA is rather optimistic as the impact of internal wiring has been ignored, as mentioned earlier in the sub-sections 3.2.2, and 3.4.1.

3.5.2 Delay

The second parameter of performance is the propagation delay. Delay of the SSIA will be worst-case delay an input signal incurs traversing through SSIA. Following four tables present the delay data of SSIA implemented using components of a standard cell library. Each cell of a tables can be considered as a component of SSIA (i.e. the rows of the tables are rows of SSIA and columns of the tables are columns; the first row of the table is row of 5-input (5X1) multiplexer; the second and the third row of the table is a

row of a 4X1 multiplexer; the fourth row is a row of a 5X1 multiplexer; all the remaining rows of the table are rows of a 3X1 multiplexer) and their content is the delay incurred. **Table 3-21** and **Table 3-22** show the individual delay. **Table 3-23** and **Table 3-24** show cumulative total delay at that point in SSIA.

Table 3-21: Individual delay of the component (multiplexer) of SSIA.

	IW_1 (ps)	**IW_2 (ps)**	**IW_3 (ps)**	**IW_4 (ps)**
S_0	151	151	151	151
S_1	137	137	137	137
S_2	131	131	131	131
S_3	151	151	151	151
S_4	110	110	110	110
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
S_{D-1}	110	110	110	110

IW_i = i^{th} issue width of SSIA; S_j = j^{th} element of stack; D = Depth of stack;

Table 3-22: Individual delay of the component (ganged multiplexer) of SSIA.

	IW_1 (ps)	**IW_2 (ps)**	**IW_3 (ps)**	**IW_4 (ps)**
S_0	159	159	159	159
S_1	141	141	141	141

S_2	134	134	134	134
S_3	159	159	159	159
S_4	116	116	116	116
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
S_{D-1}	116	116	116	116

IW_i = i^{th} issue width of SSIA; S_j = j^{th} element of stack; D = Depth of stack;

Table 3-23: Cumulative delay of the component (multiplexer) of SSIA.

	IW_1 (ps)	**IW_2 (ps)**	**IW_3 (ps)**	**IW_4 (ps)**
S_0	151	301	452	602
S_1	137	288	439	589
S_2	131	281	432	582
S_3	151	301	452	602
S_4	110	261	411	562
S_5	110	220	371	522
S_6	110	220	331	481
S_7	110	220	331	441
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
S_{D-1}	110	220	331	441

IW_i = i^{th} issue width of SSIA; S_j = j^{th} element of stack; D = Depth of stack;

Table 3-24: Cumulative delay of the component (ganged multiplexer) of SSIA.

	IW_1 (ps)	**IW_2 (ps)**	**IW_3 (ps)**	**IW_4 (ps)**
S_0	159	317	476	634
S_1	141	300	300	617
S_2	134	293	451	610
S_3	159	317	476	634
S_4	116	274	433	591
S_5	116	231	390	548
S_6	116	231	347	505
S_7	116	231	347	462
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
S_{D-1}	116	231	347	462

IW_i = i^{th} issue width of SSIA; S_j = j^{th} element of stack; D = Depth of stack;

Now the worst case scenario for propagation delay would be when the signal travels through 5X1 multiplexer for all the issue widths and that is 602ps for normal design and 634ps for ganged design.

3.6 Summary

This chapter presented the baseline data that is going to be used to measure any improvements that are made in later chapters through more sophisticated design techniques. Additionally, the baseline data included many performance parameters (such as area, propagation delay, power, capacitance and fanout), which means quite precise

performance data and accurate quantification. The components within SSIA are used in varying fanout condition and hence the detailed fanout analysis of this chapter was helpful as it shows the trend of the performance. The baseline data also allows us to observe and confirm the relationship between capacitance and propagation delay.

The chapter began with an explanation of the experimental setup and methodologies (Section 3.2). The section used examples of the inverter, tri-state buffer and 2X1 multiplexer and had separate sub-sections for explaining fundamental components and multiplexer methodologies. Section 3.3 presented the performance parameters of the fundamental component and also showed how they change with fanout and section 3.4 repeated the same process for multiplexers. Section 3.4 also explained the alternative methodology to evaluate the impact of wiring in performance. Section 3.3 and 3.4 presents the baseline data that can be used to measure any improvements that will be made in later chapters. Section 3.5 predicted the performance of SSIA if implemented using the standard cell library. Next chapter is going to collect similar data and show the enhancement that can be using the custom design approach.

CHAPTER 4

CUSTOM CELL IMPLEMENTATION OF SSIA

4.1 Introduction

This chapter investigates the custom design techniques to improve the SSIA performance [45]. This chapter mainly explores the effect of varying transistor properties on the performance [47]. The main aim of the chapter is to know what the performance gains are and what the cost penalties are and thereby investigate the associated trade-offs. Investigation in the first few sections of the chapter is similar to the previous chapter - in terms of methodology, fanout conditions and test benches. The last section investigates the custom design approach at various design levels (Transistor, Component and System level). The important rationale for performing this work is to get the pieces of evidence to prove the hypothesis by answering the following question - *What are the advantages of using custom cell design for SSIA?*

The fundamental components and multiplexers designed in this chapter have been put together in a single cell library, and the library has been referred to as a 'Custom Cell' library throughout the thesis.

Contributions made through this chapter are presented in the list below.

- Custom Cell library - created in 65nm CMOS process technology.

- A comparison of wider v/s multi-finger MOSFET implementation of fundamental components.
- An evaluation for performance, similar to the previous chapter, has been carried out for fundamental components made up of multi-finger MOSFETs. (As the above-mentioned comparison found multi-finger implementation advantageous).
- An Investigation, with a similar methodology to the previous chapter, for multiplexers implemented using the Custom Cell library, has been carried out.
- Novel components employing custom design approach at various design levels have been created.

4.2 Transistor Property Variation

Figure 4-1 is a screenshot of Cadence Virtuoso Property Editor and shows the various properties of a MOSFET. From those properties 'Width' and 'Number of fingers' have been picked and have been changed for conducting the analysis presented in this chapter. This idea is the result of inspiration from similar work done previously by others (changing finger widths) and associated benefits [47] [46].

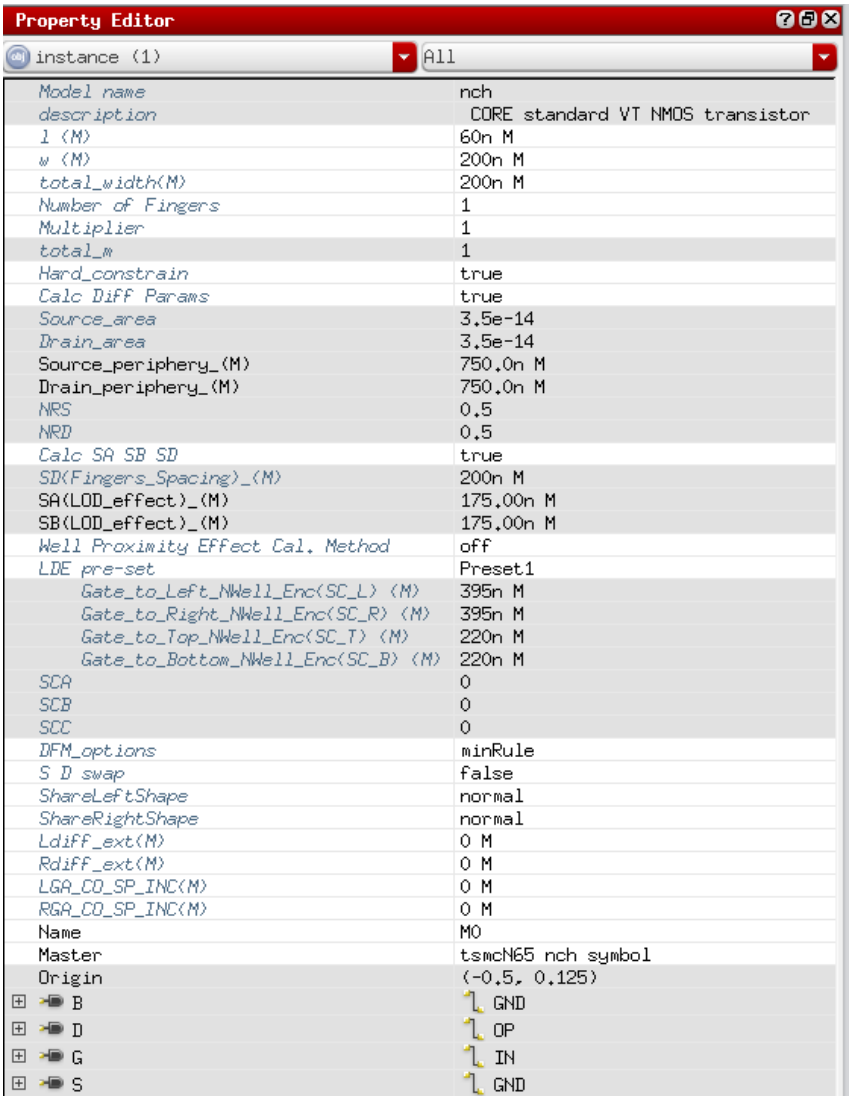


Figure 4-1: Screenshot of properties(such as width, length, numbers of fingers etc.) of n-channel MOSFET from Cadence.

Doubling the ‘width’ or ‘finger width’ has almost the same increment in the area but the reduction in propagation delay is different (explained later in sub-sections 4.3.1 and 4.3.2). The reduction in delay, in other words, improvement in speed, is higher in the case of multi-finger implementation. This is due to the fact that parasitic resistances of the transistor become parallel [47]. It also keeps the height/width same which is useful in keeping layout modular. Normal transistor symbol and layout has been shown earlier in

Figure 3-2. **Figure 4-3** shows the normal transistor layout alongside layouts of multi-finger and wider transistors. **Figure 2-3** and **Figure 4-4** shows inverter and tri-state buffer made using multi-finger MOSFETs.

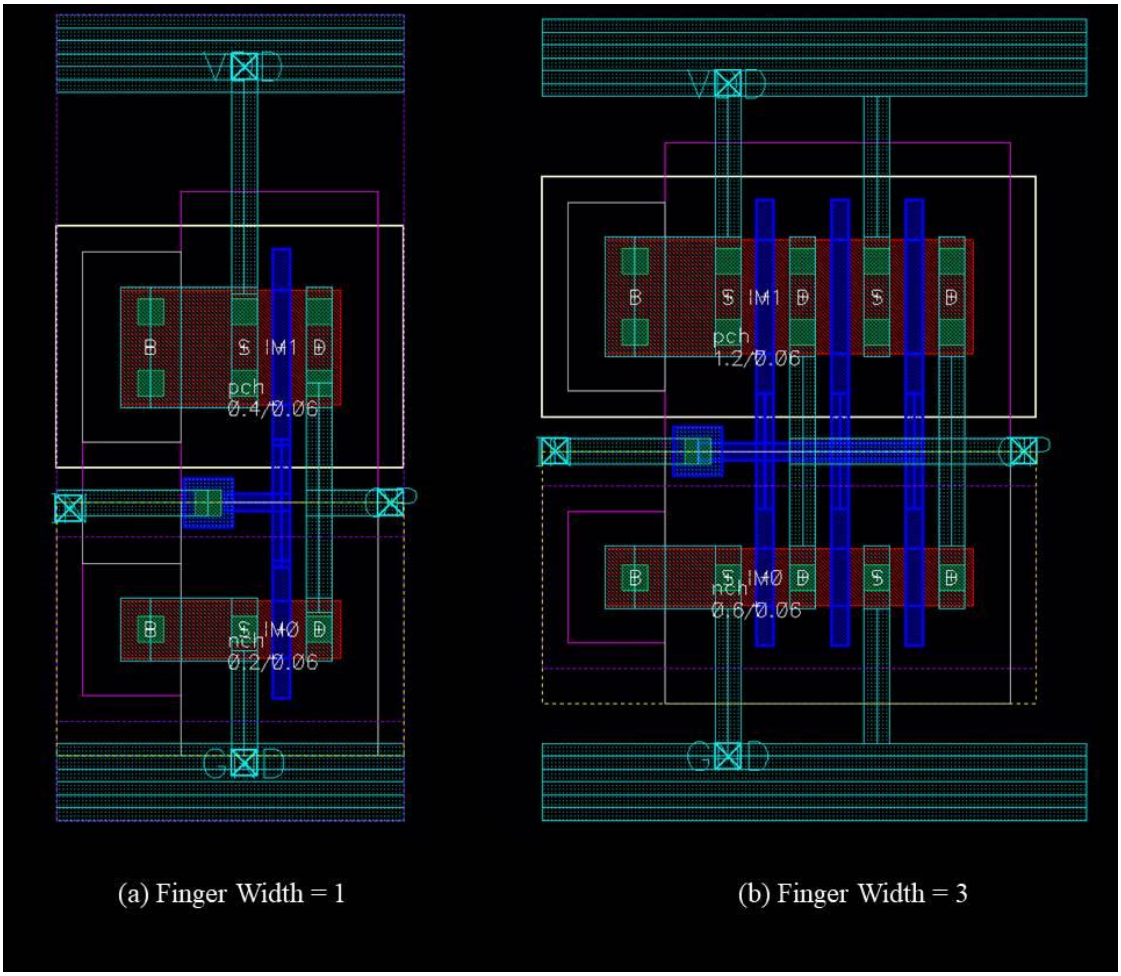


Figure 4-2: Layout of Inverter (a)FW = 1, and (b) FW = 3.

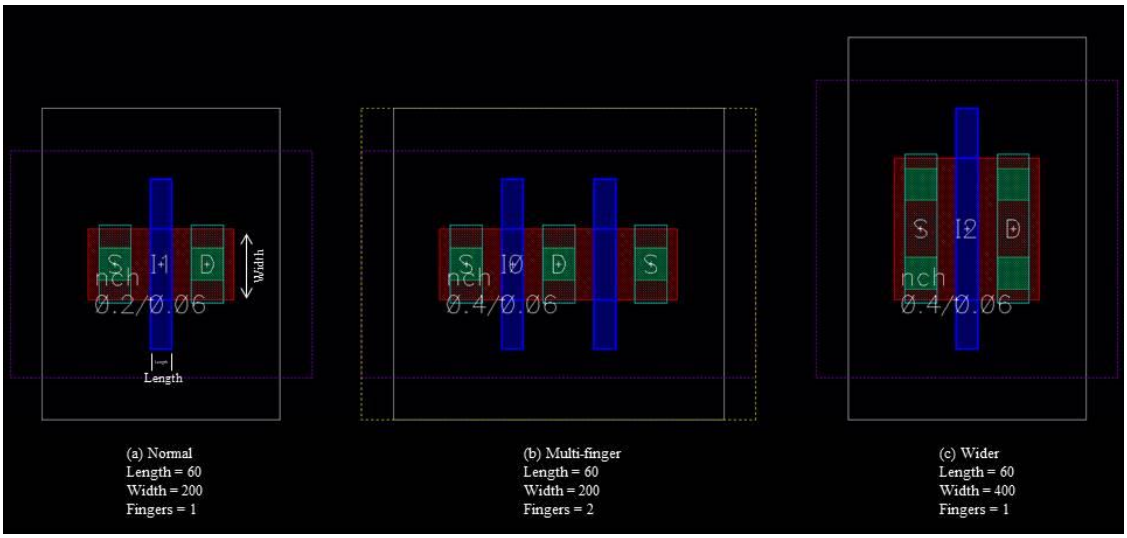


Figure 4-3: Transistor (n-channel) layouts (a) Normal (b) Multi-finger (c) Wider.

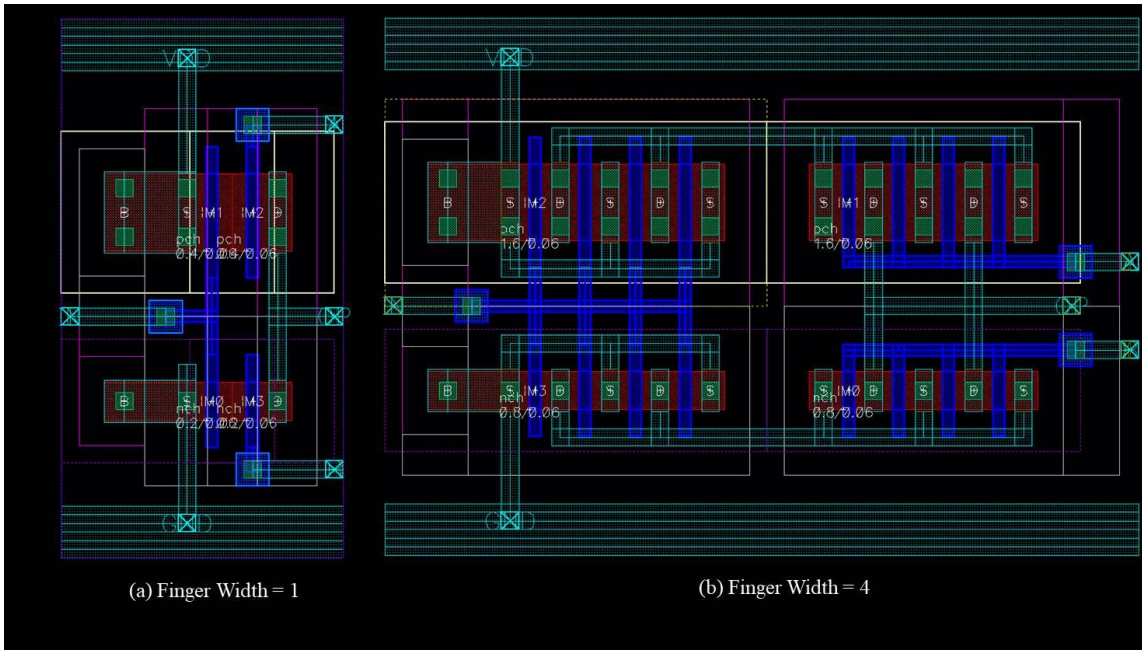


Figure 4-4: Layout of Tri-state Buffer (a)FW = 1, and (b) FW = 4.

4.3 Fundamental Components

The evaluation of fundamental components in this section is similar to section 3.3. The same methodology is used, fanouts ranging from 1 to 8 have been considered, and the same care has been taken in designing test benches. But the analysis in this chapter has been extended by collecting data for various finger widths, instead of only basic components. The evaluation of the area and delay also includes a comparison of wider v/s multi-finger implementation. The decision of choosing multi-finger implementation has been made after this comparison. The overall analysis of this section and the following section 4.4 will be helpful in choosing components for SSIA in later chapters. In this section, each parameter of performance has been given their own separate sub-section and their contribution & significance have been discussed.

4.3.1 Area

The first parameter of performance is the area. **Figure 2-3** and **Figure 4-4** have already shown an inverter and a tri-state buffer of finger widths 1 and 3 respectively. Their respective area has been calculated using the method described in section 3.2 of the previous chapter. **Table 4-1** presents the area data of inverter and tri-state buffer. The table contains values for finger widths one to four. **Table 4-2** presents the area data of 'wider' and 'multi-finger' inverter and **Figure 4-5** is a visual representation of this data. Each increment, in both the implementations, increases the area by approximately 25%. This cost is similar for both the implementations and can also be seen in the graph in **Figure 4-5**. If the area is the only priority then it is better to use wider implementation.

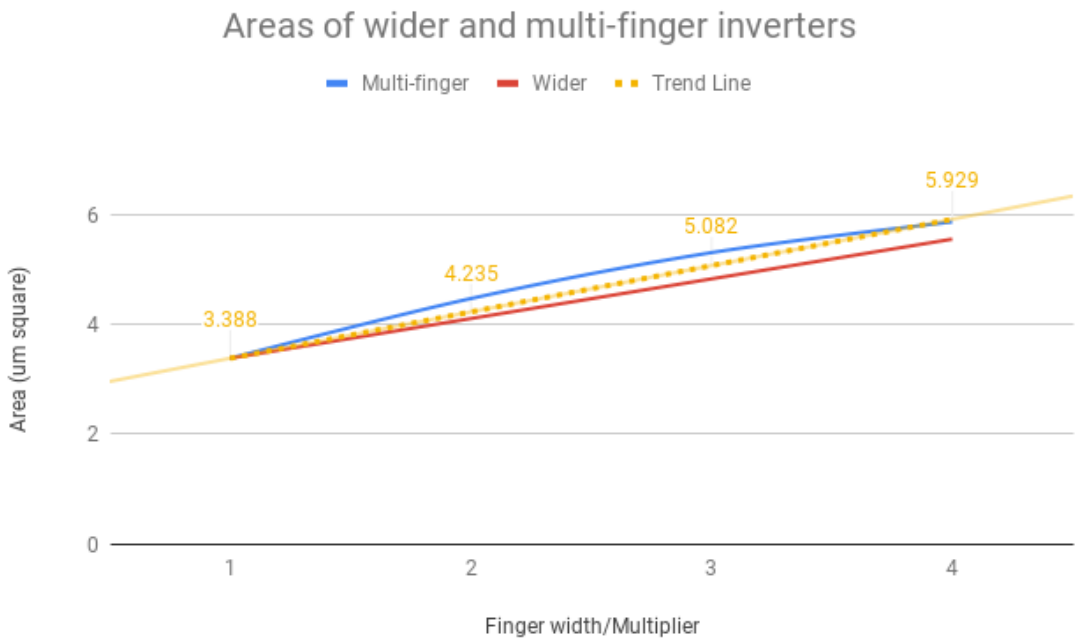


Figure 4-5: Graph showing the trend of increment in areas of wider and multi-finger inverters for each finger width/multiplier.

Table 4-1: Area of fundamental components for various finger widths in um^2.

Component (↓)	**FW = 1 (um^2)**	**FW = 2 (um^2)**	**FW = 3 (um^2)**	**FW = 4 (um^2)**
Inverter	3.4	4.5	5.3	5.9
Tri-state buffer	4.1	8.1	9.4	11.0

FW = Finger width;

Table 4-2: Areas of wider and multi-finger inverters for comparison.

Implementation (↓)	**Normal (um^2)**	**2 x Normal (um^2)**	**3 x Normal (um^2)**	**4 x Normal (um^2)**
Multi-finger	3.4	4.5	5.3	5.9

Wider	3.4	4.1	4.8	5.6

4.3.2 Delay

The second parameter of performance is the propagation delay. **Table 4-3** and **Table 4-4** show propagation delay of inverter and tri-state buffer respectively from schematic simulations and their layout simulation data are presented in **Table 4-5**. Tables contain values for finger widths one to four and fanout ranging from 1 to 8. It can be seen that higher the 'finger-width' or the 'width', lesser is the delay. Thus, a decrease in the delay is one of the benefits of increasing the number of fingers.

Table 4-3: Propagation delay of wider and multi-finger inverters for fanout ranging from 1 to 8 (schematic simulations).

Fanout	**FW = 1**	**Wider Implementation**			**Multi-finger Implementation**		
↓	**W = 0.2**	**W = 0.4**	**W = 0.6**	**W = 0.8**	**FW = 2**	**FW = 3**	**FW = 4**
FO1	13	13	14	15	12	13	13
FO2	17	16	16	16	14	15	15
FO3	20	18	18	18	16	16	16
FO4	24	18	19	19	18	17	17
FO5	27	22	21	20	20	19	19
FO6	31	24	22	22	22	21	20
FO7	34	25	23	23	24	22	21
FO8	38	27	25	24	25	23	22

FW = Number of fingers (finger widths); W = Width in micrometers (um)

Table 4-4: Propagation delay of wider and multi-finger tri-state buffers for fanout ranging from 1 to 8 (schematic simulations).

Fanout ↓	**FW = 1 W = 0.2**	**Wider Implementation**			**Multi-finger Implementation**		
		W = 0.4	**W = 0.6**	**W = 0.8**	**FW = 2**	**FW = 3**	**FW = 4**
FO1	31	31	31	32	27	28	29
FO2	39	35	34	34	31	31	31
FO3	46	38	37	36	35	33	32
FO4	53	42	39	38	38	35	34
FO5	60	46	42	40	41	38	36
FO6	67	50	44	42	45	40	38
FO7	74	53	47	44	48	42	40
FO8	82	57	50	46	52	45	41

FW = Number of fingers (finger widths); W = Width in micrometers (um)

The area cost of larger ‘width’ is slightly less compared to higher ‘finger-width’ and as shown earlier in **Table 4-2** and **Figure 4-5**. **Table 4-3**, **Table 4-4**, **Table 4-5** and **Figure 4-6** present pieces of evidence, based on propagation delay data as to why multi-finger implementation is superior compared to wider implementation for SSIA e.g. delay of multi-finger is less compared to all the wider implementation for FO3, FO4 and FO5 as shown in the graph below. Higher finger-width transistors have the same height/length. This gives the benefit of modular design which isn’t present in wider transistor design. Also, reducing delay is more important to us then reducing area. Hence, investigation for the next parameters chose higher finger-width fundamental components. If there was more time investigation for wider transistors would have also been carried out.

Table 4-5: Propagation delay of inverter and tri-state buffer for finger widths 1 to 4 and fanout ranging from 1 to 8 (layout simulations).

Fanout ↓	Inverter				Tri-state buffer			
	FW= 1	**FW= 2**	**FW= 3**	**FW= 4**	**FW= 1**	**FW= 2**	**FW= 3**	**FW= 4**
FO1	23	19	22	24	41	38	42	47
FO2	30	22	24	26	54	41	44	49
FO3	37	25	26	28	68	45	46	50
FO4	44	27	28	31	81	48	48	52
FO5	51	30	30	31	94	52	50	53
FO6	57	32	31	33	108	55	52	55
FO7	64	34	33	34	122	58	53	56
FO8	71	36	34	36	136	62	55	57

FW = Number of fingers (finger widths);

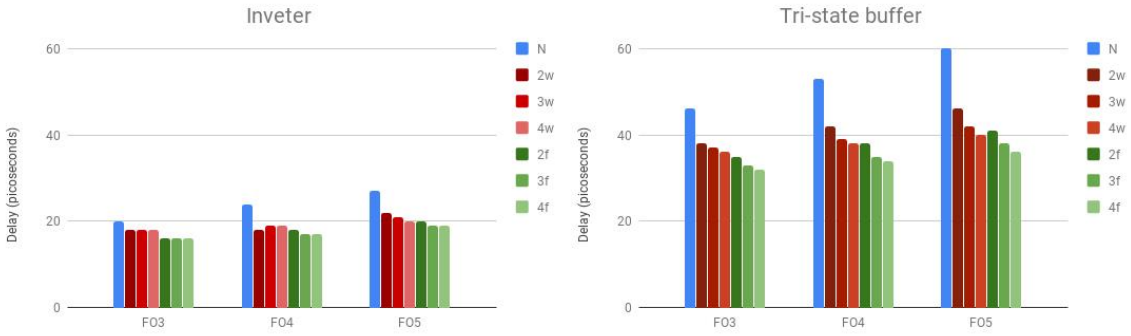


Figure 4-6: Graphs showing the comparison for wider and multi-finger implementations of inverter and tri-state buffers.

4.3.3 Capacitance

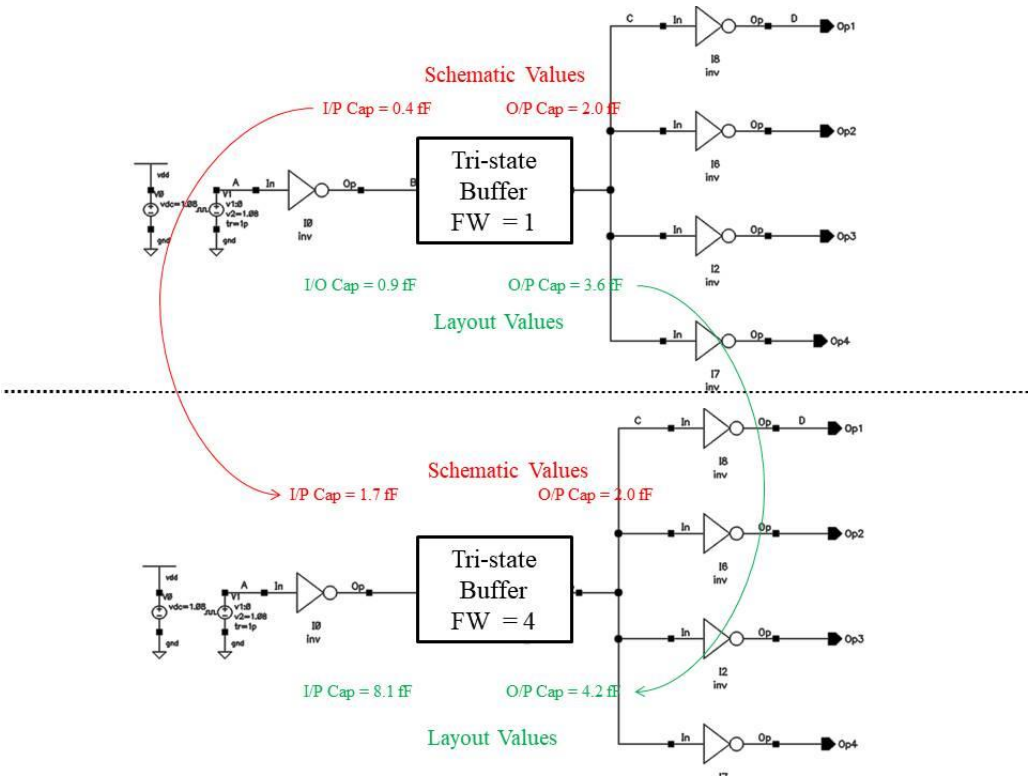


Figure 4-7: Diagram showing the change of capacitance on changing finger width for tri-state buffer with a fanout of four.

The third parameter of performance is Capacitance. A general trend of capacitance with respect to finger width can be seen in **Figure 4-7**. **Table 4-6** and **Table 4-7** present the capacitance data of inverter and tri-state buffer respectively. Tables contain values of both input and output node capacitances for finger widths one to four and fanout ranging from 1 to 8. For simplicity, only the total value of the capacitance has been presented (not individual fixed and variable components). It can be observed that output capacitance doesn't change much with a change in finger widths. This could be due to the fact that the output capacitance is mainly dependent on drain capacitance of UUT. Hence, it is the same for all finger widths (after schematic simulation) or changes

only marginally (after layout simulation). Thus, for capacitance, there isn't a significant cost or benefit of an increasing the number of fingers.

Inverter

Table 4-6: Total input and output capacitance of inverter of finger widths 1 to 4 for fanout ranging from 0 to 8 in femtofarad.

Fanout ↓	**FW = 1**		**FW = 2**		**FW = 3**		**FW = 4**	
	Sch.	**Lay.**	**Sch.**	**Lay.**	**Sch.**	**Lay.**	**Sch.**	**Lay.**
Total Input Capacitance (fF)								
All	0.4	1.0	0.9	2.4	1.3	4.8	1.7	8.1
Total Output Capacitance (fF)								
Fanout ↓	**FW = 1**		**FW = 2**		**FW = 3**		**FW = 4**	
	Sch.	**Lay.**	**Sch.**	**Lay.**	**Sch.**	**Lay.**	**Sch.**	**Lay.**
NOC	0	0.2	0	0.3	0	0.5	0	0.6
FO1	0.5	1.1	0.5	1.2	0.5	1.4	0.5	1.5
FO2	1.0	2	1.0	2.1	1.0	2.3	1.0	2.5
FO3	1.6	2.9	1.6	3.1	1.6	3.2	1.6	3.3
FO4	2.1	3.8	2.1	4.0	2.1	4.1	2.1	4.3
FO5	2.6	4.7	2.6	4.9	2.6	5.0	2.6	5.2
FO6	3.1	5.6	3.1	5.8	3.1	6.0	3.1	6.1
FO7	3.7	6.6	3.7	6.7	3.7	6.9	3.7	7.0
FO8	4.2	7.5	4.2	7.6	4.2	7.8	4.2	7.9

FW = Number of fingers (finger widths); Sch. = Schematic Data; Lay. = Layout

Data; femtoFarad (10^{-15})

Tri-state

Table 4-7: Total input and output capacitance of a tri-state buffer of finger widths 1 to 4 for fanout ranging from 0 to 8 in femtofarad.

Fanout ↓	**FW = 1**		**FW = 2**		**FW = 3**		**FW = 4**	
	Sch.	**Lay.**	**Sch.**	**Lay.**	**Sch.**	**Lay.**	**Sch.**	**Lay.**
Total Input Capacitance (fF)								
All	0.4	0.9	0.9	2.4	1.3	4.8	1.7	8.1
Total Output Capacitance (fF)								
Fanout ↓	**FW = 1**		**FW = 2**		**FW = 3**		**FW = 4**	
	Sch.	**Lay.**	**Sch.**	**Lay.**	**Sch.**	**Lay.**	**Sch.**	**Lay.**
NOC	0	0.2	0	0.4	0	0.6	0	0.7
FO1	0.5	1.0	0.5	1.3	0.5	1.5	0.5	1.6
FO2	1	1.9	1	2.2	1	2.4	1	2.5
FO3	1.5	2.7	1.5	3.0	1.5	3.2	1.5	3.3
FO4	2.0	3.6	2.0	3.9	2.0	4.1	2.0	4.2
FO5	2.5	4.5	2.5	4.7	2.5	5.0	2.5	5.1
FO6	3.0	5.3	3.0	5.6	3.0	5.8	3.0	5.9
FO7	36	6.2	3.6	6.5	3.6	6.7	3.6	6.8
FO8	4.1	7.0	4.1	7.3	4.1	7.8	4.1	7.7

FW = Number of fingers (finger widths); Sch. = Schematic Data; Lay. = Layout Data; femtoFarad (10^{-15})

4.3.4 Power

The fourth and last parameter of performance is power consumption. Tables contain values of both static and peak dynamic power for finger widths one to four and fanout ranging from 1 to 8.

Table 4-8: Peak dynamic and static power consumption of an inverter of finger width 1 to 4 for fanout ranging from 1 to 8 in nanowatts for low to high and high to low input transitions.

Peak Value of Dynamic Power (uW)								
Fanout	**FW = 1**		**FW = 2**		**FW = 3**		**FW = 4**	
↓	**L → H**	**H → L**	**L → H**	**H → L**	**L → H**	**H → L**	**L → H**	**H → L**
FO1	47	46	76	73	99	97	133	130
FO2	53	52	89	88	106	106	138	136
FO3	57	56	100	98	115	113	142	141
FO4	60	60	108	110	123	122	147	146
FO5	62	62	119	119	131	131	151	151
FO6	64	64	128	126	137	139	157	157
FO7	66	65	136	135	144	147	163	164
FO8	67	66	142	142	153	153	168	170
Static Power (nW)								
Fanout	**FW = 1**		**FW = 2**		**FW = 3**		**FW = 4**	
↓	**I/P= L**	**I/P= H**	**I/P= L**	**I/P= H**	**I/P= L**	**I/P= H**	**I/P= L**	**I/P= H**
ALL	1.2	1.0	5.5	5.1	12.2	13.6	25.3	22.4

FW = Number of fingers (finger widths); IP = Input State; L = Low state; H = High state; L → H = Transition of input state from Low to High and vice versa.

As we increase the number of fingers power consumption also increases. **Table 4-8** and **Table 4-9** present the post-layout power data of inverter and tri-state buffer respectively. Power (along with area) is the cost of an increasing number of fingers.

Table 4-9: Peak dynamic and static power consumption of a tri-state buffer of finger width 1 to 4 for fanout ranging from 1 to 8 in nanowatts for low to high and high to low input transitions..

Peak Value of Dynamic Power (uW)								
Fanout	**FW = 1**		**FW = 2**		**FW = 3**		**FW = 4**	
↓	**L → H**	**H → L**	**L → H**	**H → L**	**L → H**	**H → L**	**L → H**	**H → L**
FO1	36	32	109	95	161	145	202	185
FO2	37	33	113	98	167	152	206	188
FO3	38	34	116	103	173	157	210	193
FO4	38	35	119	107	178	162	215	199
FO5	39	35	121	109	182	167	220	205
FO6	39	36	123	112	186	170	225	210
FO7	40	36	125	114	190	174	229	214
FO8	40	36	127	115	193	179	233	218
Static Power (nW)								
Fanout	**FW = 1**		**FW = 2**		**FW = 3**		**FW = 4**	
↓	**L → H**	**H → L**	**L → H**	**H → L**	**L → H**	**H → L**	**L → H**	**H → L**

ALL	3.2	3.2	16.4	16.4	32.0	32.0	56.9	56.9

FW = Number of fingers (finger widths); IP = Input State; L = Low state; H = High state; L → H = Transition of input state from Low to High and vice versa.

4.4 Multiplexer

The evaluation of multiplexers in this section is similar to section 3.4 but extends for finger widths 1 to 4, and is using the Custom Cell library. Performance data for area and delay are presented in following subsections, its significance is discussed in the next section (4.5).

Table 4-10: Calculated area of 3X1, 4X1 and 5X1 of finger widths 1 to 4 in um^2.

Component (↓)	**FW = 1 (um^2)**	**FW = 2 (um^2)**	**FW = 3 (um^2)**	**FW = 4 (um^2)**
3X1	15.7	28.7	33.5	38.8
4X1	19.7	36.7	42.8	49.8
5X1	23.8	44.8	52.2	60.8

FW = Number of fingers (finger widths)

Table 4-11: Propagation delay of multiplexers (of finger widths 1 to 4) of SSIA for relevant fanouts in picoseconds in picoSeconds.

Component - Fanout	**FW = 1 (ps)**	**FW = 2 (ps)**	**FW = 3 (ps)**	**FW = 4 (ps)**
3X1 - FO3	110	77	79	80
4X1 - FO4	131	87	86	89
4X1 - FO5	137	89	88	91

5X1 - FO5	151	96	94	98

FW = Number of fingers (finger widths)

4.4.1 Area

The first parameter of performance is an area of multiplexers. The methodology used for calculation is the same as sub-section 3.4.1. Areas of 3X1, 4X1 and 5X1 multiplexers for various finger widths have been calculated and are presented in **Table 4-10**.

4.4.2 Delay

The second parameter of performance is the propagation delay. Propagation delay data of the multiplexers are shown in **Table 4-11**.

4.5 Cost-benefit analysis

In section 4.3 and 4.4 we looked at various parameters of performance for finger widths 1 to 4. In this section, we will present the cost-benefit analysis of changing finger widths. In general, it can be said that performance improves with higher finger widths. We will analyze how much improvement can be achieved through each increment in a number of fingers. Performance parameter that benefits from higher finger width is propagation delay, and hence we will be analyzing data presented in **Table 4-5** and **Table 4-11**. **Table 4-12** presents the data of **Table 4-5**, and similarly **Table 4-13** for **Table 4-11**, in terms of percentage of reduction in a delay relative to finger width 1. Green shading represents delay has improved compared to its predecessor. Blue shading represents delay has improved relative to finger width 1 but is less than the predecessor. Red shading represents delay has worsened. It can be observed that going from finger width 1 to 2 has been advantageous for all cases, and going from finger width 2 to 3 has

been advantageous for most but the improvement is far less (hence the diminishing return). In addition, going from finger 3 to finger width 4 isn't advantageous for any case. This behaviour can be attributed to self-loading. Increasing finger width or width increased the drive current and hence the delay starts to decrease but after a point capacitance also increases quite a lot and hence improvement starts to fade [66]. Thus, we found out that finger width 2 are the most beneficial, considering the diminishing return and the area cost. That's why 2g and 3g implementation (presented in the next section) has been investigated only for finger width 2.

Table 4-12: Delay comparison relative to finger width 1 for fundamental components.

Fanout	**Inverter**				**Tri-state buffer**			
↓	**FW= 1**	**FW= 2**	**FW= 3**	**FW= 4**	**FW= 1**	**FW= 2**	**FW= 3**	**FW= 4**
FO1	0	0.17	0.04	-0.04	0	0.07	-0.02	-0.15
FO2	0	0.27	0.20	0.13	0	0.24	0.19	0.09
FO3	0	0.32	0.30	0.24	0	0.34	0.32	0.26
FO4	0	0.39	0.36	0.30	0	0.41	0.41	0.36
FO5	0	0.41	0.41	0.39	0	0.45	0.47	0.44
FO6	0	0.44	0.46	0.42	0	0.49	0.52	0.49
FO7	0	0.47	0.48	0.47	0	0.52	0.57	0.54
FO8	0	0.49	0.52	0.49	0	0.54	0.60	0.58

Yellow – Starting Point; Green – Improvement; Blue – Diminishing Return; Red – Worsened;

Table 4-13: Delay comparison relative to finger width 1 for multiplexers.

Component – Fanout (↓)	FW = 1	FW = 2	FW = 3	FW = 4
3X1_FO3	0%	30%	28%	27%
4X1_FO4	0%	34%	34%	32%
4X1_FO5	0%	35%	36%	34%
5X1_FO5	0%	36%	38%	35%

Yellow – Starting Point; Green – Improvement; Blue – Diminishing Return; Red – Worsened;

4.6 2g and 3g Implementation for f2

2g and 3g components, which consider the impact of internal wiring and explained earlier in section 3.4.3 have been implemented using multi-finger MOSFETs of finger width 2 in this section, and their performance data are presented. They are labelled as $2g_{F2}$ and $3g_{F2}$ and multiplexers designed using them have been given subscripts ‘gF2’. **Table 4-14** presents the area and **Table 4-15** presents the propagation delay. **Table 4-15** is analogous to table in sub-section 3.4.3. There the difference of delay in percentage was less than 5% of the original value but here it’s much larger. Therefore it can be concluded that it’s important to consider the factor of internal wiring in the calculation to be more precise.

Table 4-14: Length, width and area of ganged components and ganged multiplexer of finger width 2 in um^2.

Component (↓)	Length (um)	Width (um)	Area (um2)	GFW1 (um2)
$2g_{F2}$	2.8	7.4	20.6	9.5
$3g_{F2}$	2.8	9.7	27.2	14.8
$3X1_{gF2}$	2.8	11.2	31.3	18.2
$4X1_{gF2}$	2.8	16.2	45.3	22.4
$5X1_{gF2}$	2.8	18.5	51.9	27.7

GFW1 – Ganged design and finger width = 1;

Table 4-15: Propagation delay in picoseconds of finger width 2 ganged 3X1, 4X1 and 5X1 of SSIA for relevant fanouts in picoseconds with columns for normal and the difference.

Component – Fanout (↓)	Ganged Design (ps)	Normal Design (ps)	Difference in Percentage
$3X1_{gF2}$ - FO3	91.2	77.0	18%
$4X1_{gF2}$ - FO3	104.8	86.6	21%
$4X1_{gF2}$ - FO4	110.8	88.8	25%
$5X1_{gF2}$ - FO5	121.9	96.1	27%

4.7 Equal sized NMOS and PMOS - equalNP

The standard CMOS implementation of Inverters has an aspect ratio of 2:1 in order to get equal rise time and fall time. In this section, the impact of changing that

aspect ratio to 1:1 has been investigated. This will result in unequal rise time and fall time but as we consider the worst-case delays in calculation this won't pose a major challenge.

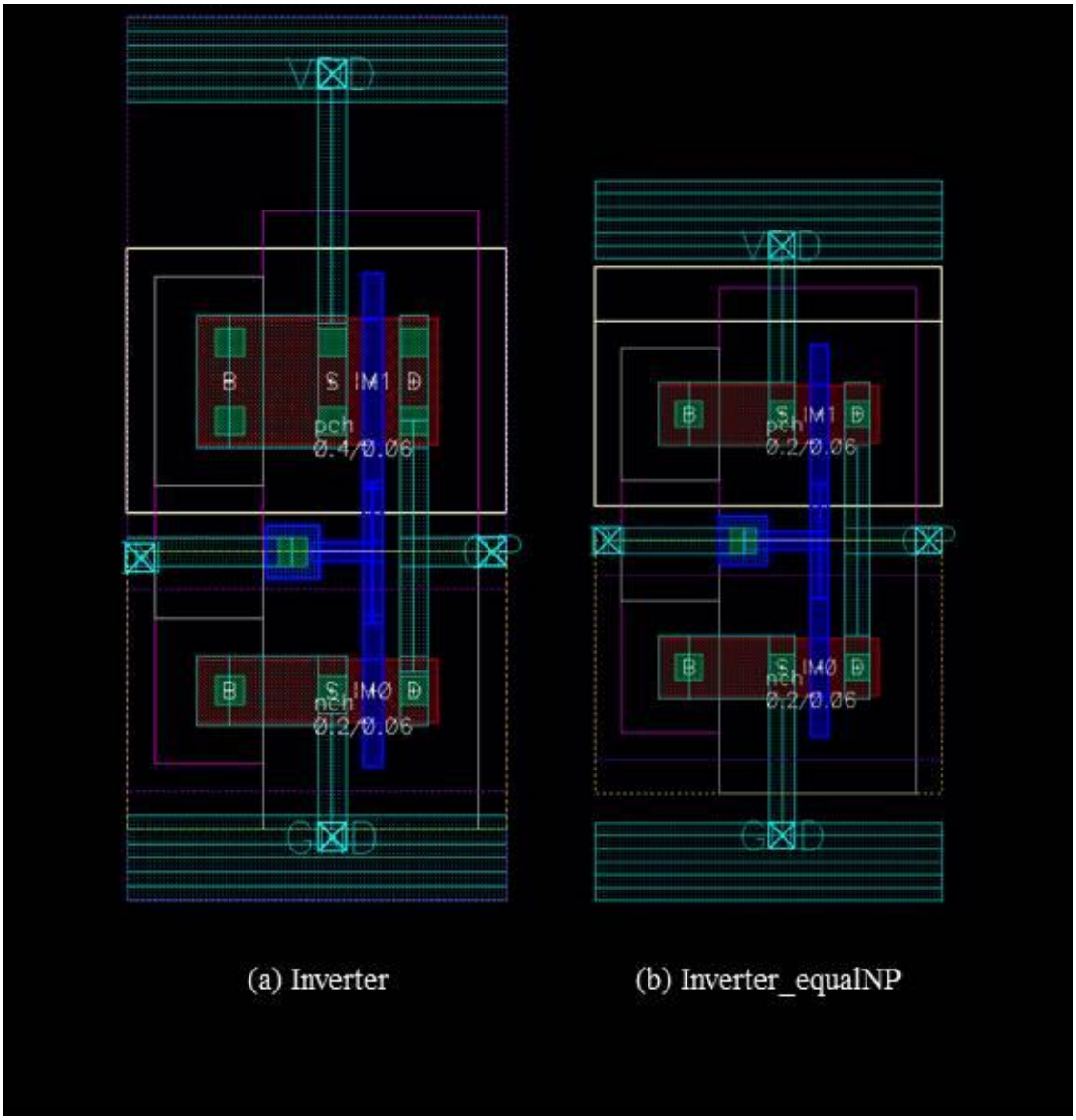


Figure 4-8: Layout of (a) inverter, and (b) inverter_equalNP.

These novel fundamental components, customised at the transistor level, have the suffix 'equalNP' to their names as they have equal size NMOS and PMOS. We designed inverter and tri-state buffer with NMOS and PMOS of equal width and

designed the 3X1 multiplexer using these fundamental components. Multiplexers made using these fundamental components have also been given the same suffix ‘equalNP’ e.g. 3X1_equalNP.

Table 4-16: Area of equalNP and normal components.

Components (↓)	**EqualNP area (um^2)**	**Normal area (um^2)**
Inverter	3.0	3.4
Tri-state buffer	3.7	4.1
Inverter (finger width = 2)	3.7	4.5
Tri-state buffer (finger width = 2)	7.2	8.1
3X1	14.0	15.7
$3X1_g$	16.8	18.2
$3X1_{F2}$	25.3	28.3
$3X1_{gF2}$	27.8	31.3

The measurements of 3X1 multiplexer have been given in These novel fundamental components, customised at the transistor level, have the suffix ‘equalNP’ to their names as they have equal size NMOS and PMOS. We designed inverter and tri-state buffer with NMOS and PMOS of equal width and designed the 3X1 multiplexer using these fundamental components. Multiplexers made using these fundamental components have also been given the same suffix ‘equalNP’ e.g. 3X1_equalNP.

Table 4-16 and **Table 4-17**, where it can be seen that area is lesser compared to normal approach and for finger width 2 delay is also lesser, that’s clearly a win-win situation. There is more investigation that can be done in this direction such as area,

delay, power and capacitance of ‘Inverter_equalNP’, and ‘Tri-state_buffer_equalNP’ can be measured, and 4X1_equalNP and 5X1_equalNP can be designed and their area and delay can also be calculated.

Table 4-17: Propagation delay data of 3X1 Multiplexer made using Custom Fundamental Components.

Components (↓)	**Normal (ps)**	**equalNP (ps)**
3X1	110.2	117.7
$3X1_{g}$	115.5	128.0
$3X1_{F2}$	77.0	76.9
$3X1_{gF2}$	91.2	80.1

4.8 Hybrid Multiplexers

In previous designs, all tri-state buffers of a multiplexer are uniform (i.e. all finger width 1, 2, 3 or 4). In our component level customization, the novel multiplexers have been designed. The customisation attempted here involves changing some of the tri-state buffers from finger width 1 to a finger width 2. This is done to speed-up those tri-state buffers with minimal area cost. We changed one of the three tri-state buffers in 3X1, two of the four in 4X1, and two of the five in 5X1. We believe these are the number of tri-state buffers which will be used frequently in the respective multiplexers. As multiplexers are hybrid of the transistor of finger width 1 and finger width 2 we have added the suffix ‘hybrid’. In this section, measurements of the performance parameters such as area and delay have been presented. **Table 4-18** presents the area and **Table 4-19** presents the delay. We observed that both kinds, finger width one and two, of tri-state buffer

experiences different delays. The delay measurements have been done for both kids and have been presented in **Table 4-19**.

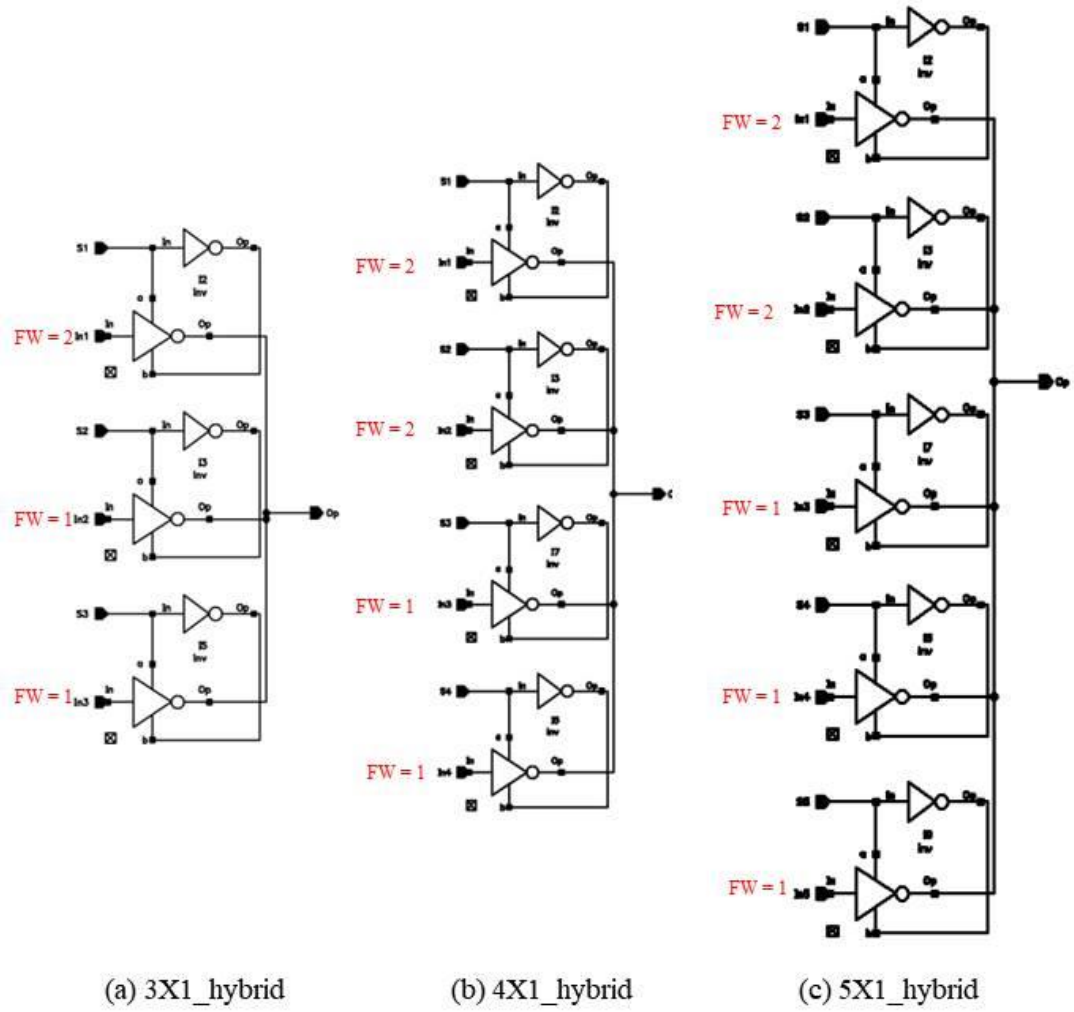


Figure 4-9: Schematic of hybrid multiplexers (a) 3X1_hybrid, (b)4X1_hybrid, and (c)5X1_hybrid.

Table 4-18: Area of normal and hybrid multiplexers.

Components (↓)	**Hybrid area (um^2)**	**Normal area (um^2)**

3X1	19.6	15.7
4X1	27.7	19.7
5X1	31.8	23.8

Table 4-19: Propagation delay data of hybrid multiplexers.

Components – Fanout - Remark (↓)	**FW1 (ps)**	**FW2 (ps)**	**Average (ps)**
3X1 – FO3 – 3 FW1	110.3	N/A	110.3
3X1_hybrid – FO3 – 2 FW1 and 1 FW2	128.3	77.8	111.8
4X1 – FO4 – 4 FW1	130.5	N/A	130.5
4X1_hybrid – FO4 – 2 FW1and 2 FW2	167.8	92.8	130.3
4X1_hybrid – FO5 – 4 FW1	137.4	N/A	137.4
4X1_hybrid – FO5 – 2 FW1 and 2 FW2	175.7	98.9	137.3
5X1 – FO5 – 5 FW1	150.6	N/A	150.6
5X1_hybrid – FO5 – 3 FW1 and 2 FW2	188.3	102.6	154.0

FW1 = Finger width = 1; FW2 = Finger width = 2;

4.9 Comparison of different multiplexer implementations

This section provides a comparison of various multiplexers presented so far, i.e. *Multi-finger Multiplexer* implementation, *equalNP Multiplexer* implementation, and *Hybrid Multiplexer* implementation. Multifinger - A cost-benefit analysis of Multi-finger implementation has been provided earlier in section 4.5. As the finger widths of MOSFETs increases, the area also increases with benefit in speed. One of the

improvements we thought of is to use higher finger widths for only some of the multiplexers. That way increase in the area won't be that much and if multiplexers are selected carefully then speed benefit can be really good but the disadvantage of this approach is that each multiplexer has to be customised for each application – that can be resource intensive.

The conventional approach in CMOS design is to use 'pch; transistor twice the size of 'nch'. This is to make sure rise time and fall time of the gate is equal. We decided to experiment with the unconventional approach of having 'pch' and 'nch' of equal size. As presented in section 4.7 we were able to reduce the area at the cost of higher propagation delay, which can be beneficial for components which are not in critical timing path. They can be made slower without affecting the overall propagation delay at the benefit of the lesser area.

4.10 System-level customisation

Table 4-20: Multiplexers and their possible versions.

Multiplexers (↓)	**Different possible versions**
5X1	$5X1_{F1}$, $5X1_{F2}$, $5X1_{F3}$, $5X1_{F4}$, 5X1_hybrid
4X1	$4X1_{F1}$, $4X1_{F2}$, $4X1_{F3}$, $4X1_{F4}$, 4X1_hybrid
3X1	$3X1_{F1}$, $3X1_{F2}$, $3X1_{F3}$, $3X1_{F4}$ $3X1_{F1}$_equalNP, $3X1_{F2}$_equalNP, 3X1_hybrid

The SSIA is made up 3X1, 4X1 and 5X1 multiplexers and there are many choices available for these multiplexers, as presented earlier in this, and the previous chapters, e.g there are 7 different versions for 3X1, and 5 different versions for 4X1 and 5X1. The

performance data of each version can be different. The users can customise the SSIA at system level based on their requirement. For example, if all multiplexers have a finger width one transistor then the area will be quite less but delay won't be as good as multi-finger versions. Area and delay can be further traded-off for finger widths 2, and 4. In SSIA of depth 5, there are eight 5X1, eight 4X1 and four 3X1 multiplexers. Thus, there can be 366 trillion – $7^4x5^8x5^8$ – different possible versions of SSIA. Considering this number it can be assumed, without a doubt, that the analysis of system level customisation can get quite lengthy.

4.11 Summary

It is demonstrated that the faster components are possible at the expense of larger chip area, although the trend is kind of diminishing return. Further, the impact of self-loading also has also come in sight, which implies a certain number of fingers (two in our case) offer the best performance gains. With more customisation, it was found that there is scope for improvement using system level customisation – this topic is further discussed in the next chapter (CHAPTER 5).

The chapter began with a discussion related to transistor property variations in section 4.2. In section 4.3 and 4.4, it was found that faster components are possible at the expense of a larger chip area and diminishing return trend has been observed which was specifically explained in section 4.5. In the same section, the impact of self-loading also came in sight, which implied a certain number of fingers (two in our case) offered the best performance gains. In section 4.9, it was found that there is huge scope for improvement using system level customisation and that the system level customisation

can get quite complex. Thus, their discussion has been presented separately in the next chapter (CHAPTER 5).

CHAPTER 5

INVESTIGATION OF SSIA

5.1 Introduction

In the previous chapter (in sub-section 4.9), it was demonstrated that there are trillions of different versions of SSIA. Different versions of SSIA offers different benefits at a different cost, and these cost-benefits have been discussed in this chapter. The chapter starts with an explanation of the methodology for calculating performance parameters of SSIA. This chapter also discusses a novel tool called SSIA Predictor, which can predict a suitable version of SSIA based on user requirements [67]. Thus, the chapter mainly discusses SSIA versions which have been chosen either manually or using an algorithm. At the end of the chapter comparison of SSIA with register files are also included.

Contributions made through this chapter are presented in the list below.

- Power/Energy consumption at SSIA level has been calculated and discussed
- Using manual analysis recommendations have been made for SSIA version that:
    - Occupies the least area on the chip.
    - Offers the smallest worst-case delay.
    - Consumes the least power.

- SSIA Predictor – an excel dashboard and DARWIN – an automated toolset have been proposed. These tools permit SSIA design space exploration, evaluation, and can ultimately make a recommendation using an algorithmic approach.
- Recommendations made by SSIA Predictor have been discussed and some are also compared with those, which were chosen manually.
- A new comparison of SSIA v/s register files, first done by Crispin-Bailey and Mullane, has been provided [1].

5.2 Methodology

Investigation of SSIA covers three performance parameters: Area, Delay and Power. A preliminary investigation of SSIA has already been presented in Section 3.5. This chapter extends that investigation and is more in-depth. Like preliminary investigation, here as well first order projection has been used. The SSIA that we consider for this chapter has an *Issue Width* of four and *Depth* of eight elements as shown in **Figure 5-1**. This section uses standard cell version of SSIA (SSIA_A) as an example, its configuration is shown in **Table 5-1**.

This section has sub-sections for each parameter which explains the methodologies used to calculate them. Some of the necessary backgrounds for this chapter has also been covered in this section.

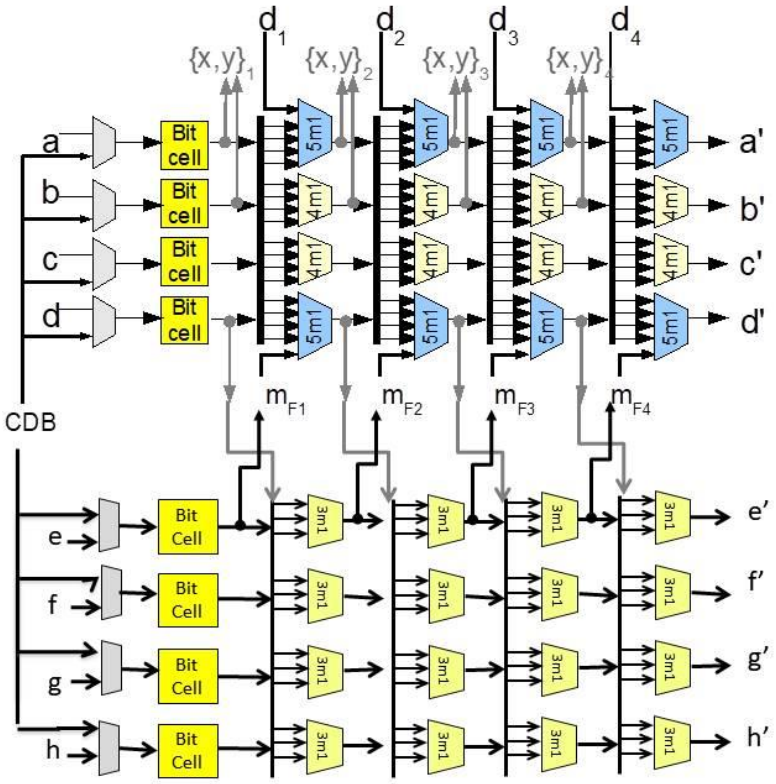


Figure 5-1: SSIA_A version of SSIA (depth = 8).

Table 5-1: Configuration of SSIA_A (standard cell version).

	IW_1	**IW_2**	**IW_3**	**IW_4**
S_0	5X1_F1_FO5	5X1_F1_FO5	5X1_F1_FO5	5X1_F1_FO5
S_1	4X1_F1_FO5	4X1_F1_FO5	4X1_F1_FO5	4X1_F1_FO5
S_2	4X1_F1_FO4	4X1_F1_FO4	4X1_F1_FO4	4X1_F1_FO4
S_3	5X1_F1_FO5	5X1_F1_FO5	5X1_F1_FO5	5X1_F1_FO5

S_4	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3
S_5	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3
S_6	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3
S_7	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3

5.2.1 Area

The area calculation has been explained previously during preliminary investigation in sections 3.5. The area of SSIA is simply the sum total of area of all components which make up the SSIA structure. As we have fixed the issue width to four and depth to eight for this chapter, the new simplified equation for this chapter is **Eq. 5-1**. Based on the **Eq. 5-1** and **Table 3-14**, the area of the example – standard cell version of SSIA (SSIA_A) – is 598 um^2.

$$A = 4 * [2 * 5X1_{area} + 2 * 4X1_{area} + 4 * 3X1_{area}]$$
$$= 8 * [5X1_{area} + 4X1_{area} + 2 * 3X1_{area}] \quad \textbf{Eq. 5-1}$$

5.2.2 Delay

The delay calculation has also been explained previously during preliminary investigation in subsection 3.5.2. Each input signal of stack array will experience a different amount of delay based on its journey in the SSIA. Hence the worst-case delay values are of the interest and have been presented in the chapter. The cumulative worst-case delay of each component is dependent on elements through which signal travels before arriving there. Each component through which signal passes adds some delay. A chart to explain this delay dependency of each component of an issue width is presented in **Figure 5-2** below.

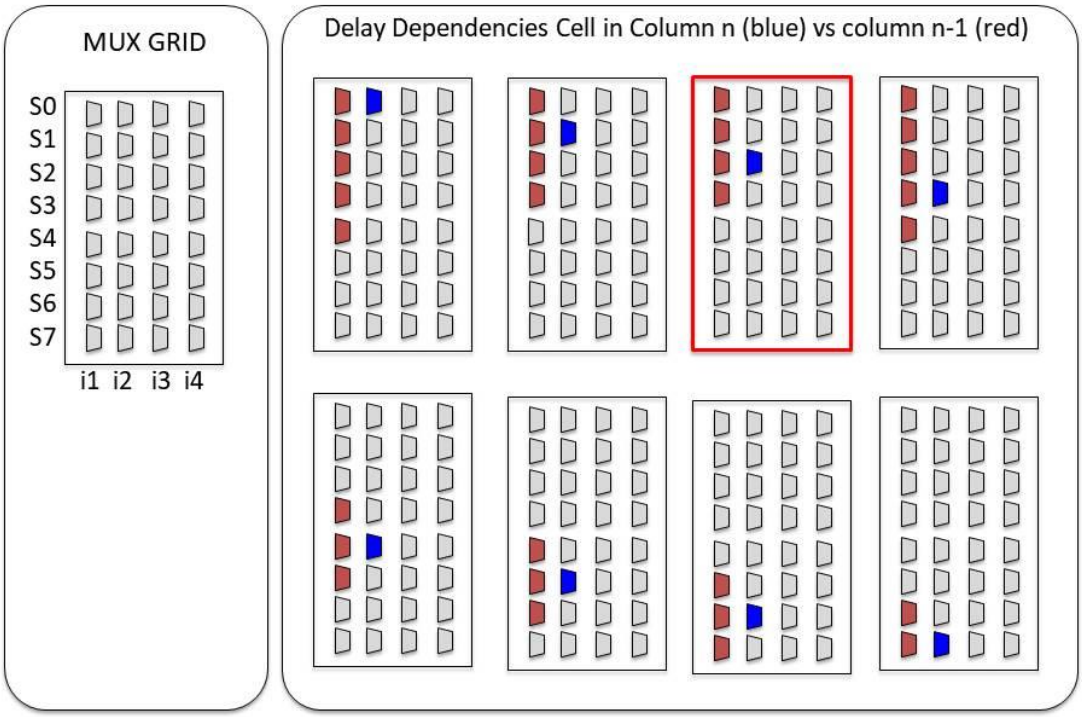


Figure 5-2: Delay dependency chart.

The worst-case cumulative delay of the component is the sum total of the delay of the component and worst-case delay of components on which it is dependent. Consider S2 element of issue slot 2 (i2) as an example (red border block of **Figure 5-2**). It's the 4X1 multiplexer with a fanout of 4 (4X1_FO4) and highlighted in blue and its individual delay is approximately 131ps according to **Table 3-15** on page 72. Its cumulative worst-case delay is the sum of individual delay and worst-case individual delay of components highlighted in red in i1 – which is approximately 151ps – and hence S2's cumulative worst-case delay is approximately 281ps (it's not 282ps because values have been

rounded off to nearest integer). This cumulative worst-case delay of all components is presented in **Table 5-2**.

Table 5-2: Cumulative Delay of SSIA_A Components.

	IW_1 (ps)	**IW_2 (ps)**	**IW_3 (ps)**	**IW_4 (ps)**
S_0	151	301	452	602
S_1	137	288	439	589
S_2	131	281	432	582
S_3	151	301	452	602
S_4	110	261	411	562
S_5	110	220	371	522
S_6	110	220	331	481
S_7	110	220	331	441
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
S_{D-1}	110	220	331	441

The table has 4 columns like SSIA and the respective cell position correspond to elements position in SSIA. This is similar to **Table 3-21** which showed the delay of an individual component. An equation, like **Eq. 5-1** for the area, and for the delay, may be hard to write but it can be said that **the worst-case delay for the SSIA will be the highest delay in the final issue slot and that will be used as a performance**

parameter. In our example of the standard cell design model, this value is approximately 602ps.

5.2.3 Power/Energy Consumption

Unlike area and delay, the power/energy consumption at SSIA level wasn't discussed in CHAPTER 3, only power consumption at component level was investigated. Based on certain assumption and the data of fundamental components, data for SSIA power consumption can be estimated. This sub-section explains this approach and assumptions underlying the approach. Though this estimation will only be close approximation, because to estimate real power/energy consumption of SSIA a model for stack behaviours during its execution would be required, which represents the average switching behaviour of every bit of every stack element, based on an accurate model of real-world workloads, but unfortunately this is beyond the scope of this thesis. Instead, we use a worst-case mode where every multiplexer is assumed to change its input selection and its output state for every clock cycle. This provides a worst-case performance ceiling for each design, in reality, power will almost always be lower than this estimate. The data may be approximate but as the approach is the same for all the versions and hence this can certainly provide an accurate comparison of various versions. In addition to the methodology, this sub-section also presents the results of power/energy consumption experiments of fundamental components. The power/energy consumption of SSIA has been estimated under the following assumptions.

- The static component is presented as power consumption.
- The dynamic component is presented in terms of energy consumed during a single switching event.

- Static power consumption calculation assumes no switching.
- In one clock cycle
    - Only two switching events occur for any multiplexer – one in the tri-state buffer and the other in the inverter.
    - Each multiplexer has only one active tri-state buffer.
- Dynamic energy consumed by a tri-state buffer of the multiplexer in high impedance (high-Z) state is very low and can be considered null (As verified in measurements from simulation and can be seen in **Figure 5-4**).
- A multiplexer is assumed to swap the input channel, to make sure we have the worst-case scenario.
- Active tri-state has a 50/50 Hi/Low switching behaviour – again for the worst-case calculations.

In this chapter, power/energy consumption of SSIA is presented in two components - the static component (power consumption when there is no switching) and dynamic component (energy consumed due to switching events). Each of them is explained in their own sub-sub-sections.

5.2.3.1 *Static Component – Power Consumption of SSIA*

Static power consumption of inverter and tristate buffer has been presented earlier in section 4.3.4 in **Table 4-8** and **Table 4-9** on page 97. The same measurements have been done again and resulting data has been presented in **Table 5-3**. Using our assumptions, static power consumption of multiplexer has been calculated and presented below in **Table 5-4**. Similarly, the power consumption of SSIA can be calculated using

Table 5-4 and **Eq. 5-2**. For our example, standard cell version of SSIA (SSIA_A) this value is approximately 420 nW.

Table 5-3: Static power consumption data of fundamental components.

Components (↓)	**FW = 1**	**FW = 2**	**FW = 3**	**FW = 4**	**equalNP**	
					FW = 1	**FW = 2**
Inverter	1.1nW	5.3nW	12.9nW	23.9nW	0.8nW	3.9nW
Tri-state buffer	3.2nW	16.5nW	32.0nW	56.9nW	2.1nW	10.9nW

FW = Finger width;

Table 5-4: Static power consumption data of multiplexers.

Components (↓)	**FW = 1**	**FW = 2**	**FW = 3**	**FW = 4**
3X1	10.7nW	54.5nW	108.9nW	194.6nW
3X1_equalNP	6.9nW	36.6nW	-	-
4X1	13.9nW	70.9nW	140.9nW	251.5nW
5X1	17.1nW	87.3nW	172.9nW	308.4nW

FW = Finger width;

An equation to calculate the value for SSIA

$$P_s = 8 * \left[5X1_{power} + 4X1_{power} + 2 * 3X1_{power}\right] \qquad \textbf{Eq. 5-2}$$

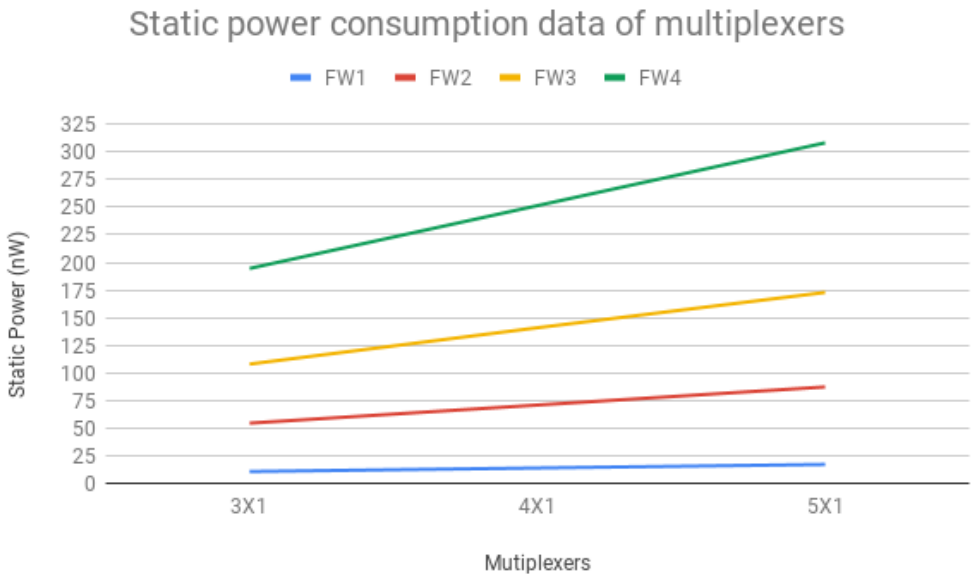


Figure 5-3: Static Power Consumption of Multiplexers.

It is interesting to note that the power data reinforces the idea that the ganged tri-state is a very effective design model. It can be estimated that 2X1 multiplexer will consume 7.5nw power given the linear relationship is shown in **Figure 2-3**. Four 2X1 multiplexers will be required to build a 5X1 multiplexer and thus its power consumption will be around 30nw, whereas 5X1 multiplexer that is designed using tri-state buffers will consume only 17.1 nW.

5.2.3.2 *Dynamic Component – Energy consumption of SSIA*

Previously in the section 4.3.4 peak value of dynamic power has been presented. A better estimation of power/energy due to transition would be the total energy consumed rather than the peak values; those values have been calculated and presented in this section. **Figure 5-4** is a screenshot of one of the graphs that are used to measure these values. In the figure, the red line is a graph for power consumption and the yellow is for energy consumption. The data obtained have been presented in **Table 5-5** and **Table 5-6**.

Using our assumptions and data of fundamental components, the energy consumption of multiplexer can be calculated and are presented in the **Table 5-7**. Similarly, energy consumption of SSIA can be calculated using **Table 5-7** and **Eq. 5-3**. For our example – standard cell version of SSIA (SSIA_A) – this value is approximately 192 fJ.

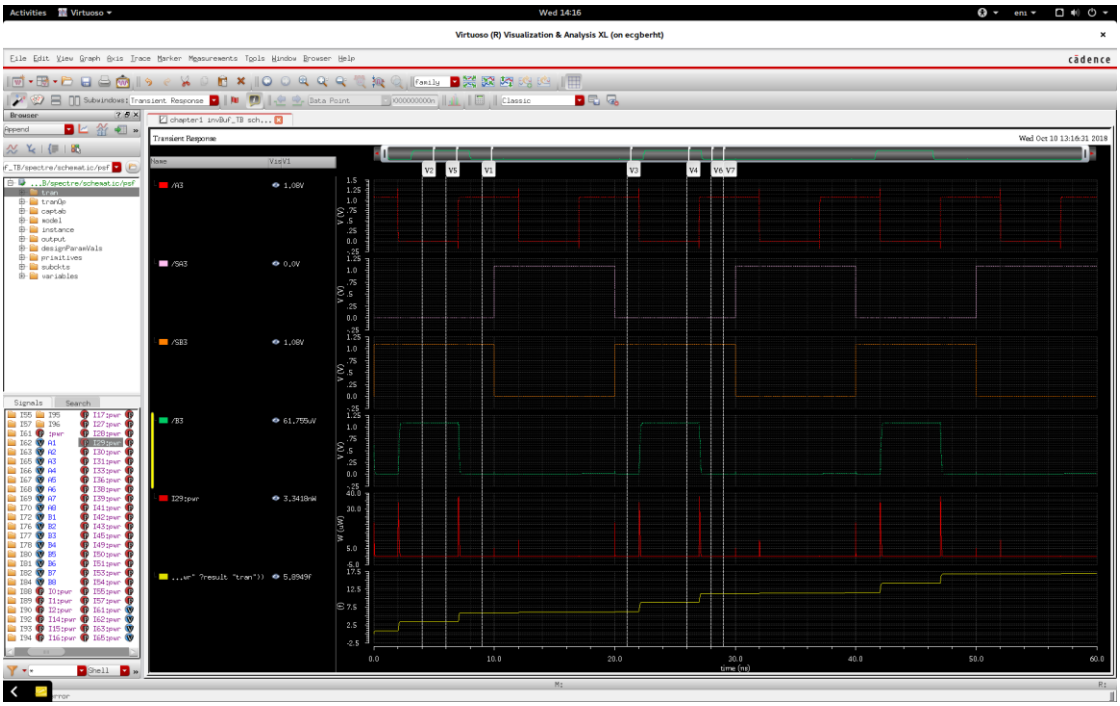


Figure 5-4: Cadence screenshot of power/energy calculations of the tri-state buffer.

Table 5-5: Energy consumption of inverter for various fanouts in femtojoules.

Fanout ↓	**FW = 1 (fJ)**	**FW = 2 (fJ)**	**FW = 3 (fJ)**	**FW = 4 (fJ)**	**equalNP (fJ)**	
					FW = 1	**FW = 2**
FO1	1.11	1.81	3.62	8.13	0.96	1.45
FO2	1.67	2.37	4.09	8.45	1.525	2.015

FO3	2.24	2.94	4.57	8.76	2.09	2.58
FO4	2.82	3.51	5.09	9.14	2.67	3.145
FO5	3.40	4.08	5.61	9.51	3.25	3.71

Table 5-6: Energy consumption of tri-state buffer for various fanouts in femtojoules.

Fanouts (↓)	**FW = 1 (fJ)**	**FW = 2 (fJ)**	**FW = 3 (fJ)**	**FW = 4 (fJ)**	**equalNP (fJ)**	
					FW = 1	**FW = 2**
FO3	2.82	6.03	10.01	15.66	2.48	4.71
FO4	3.40	6.58	10.55	16.18	3.06	5.27
FO5	3.99	7.14	11.09	16.70	3.65	5.83

Table 5-7: Energy consumption of various multiplexers.

	FW = 1	**FW = 2**	**FW = 3**	**FW = 4**
3X1_FO3	5.06	8.97	14.58	24.42
3X1_equalNP	4.56	7.29	-	-
4X1_FO4	6.22	10.09	15.64	25.32
4X1_FO5	6.80	10.66	16.15	25.69
5X1_FO5	7.38	11.21	16.69	26.21

An equation to calculate SSIA value is shown below.

$$P_D = 4 * \left[2 * 5X1\_FO5_{power} + 4X1\_FO5_{power} + 4X1\_FO4_{power} + 4 * 3X1\_FO3_{power}\right] \quad \textbf{Eq. 5-3}$$

5.2.4 Performance Parameters of an SSIA Version

Table 5-8: Performance parameters of SSIA_A (the standard cell version).

Area	598 um^2
Delay (worst-case)	602 ps
Static - Power Consumption	420 nW
Dynamic – Energy Consumption	192 fJ

5.3 Manual Selection for SSIA

The performance of SSIA can be quantified in terms of three parameters (area, delay and power), as explained in the previous section (5.2). Through this section, the version of SSIA is recommended which are best for each of the three parameters i.e. version of SSIA that will:

- Occupy the lowest area.
- Have the smallest worst-case delay.
- Have the lowest power consumption.

5.3.1 Area

Standard cell design has a low area compared to all the custom design approach, except equalNP version – an example of this for the 3X1 multiplexer is shown in **Table**

5-9. So, standard cell implementation of SSIA in which 3X1 multiplexers are replaced with 3X1_equalNP multiplexer will offer the lowest area. The configuration for this version (SSIA_B) is shown in the table below. The hybrid versions also can't provide an SSIA version better than this because some of the components of the hybrid have higher finger width which means larger than the normal area.

Table 5-9: Different versions of 3X1 multiplexers and their area.

Type of 3X1 Multiplexer	**Area (um^2)**
Finger width = 1	15.7
Finger width = 2	28.7
Finger width = 3	33.5
Finger width = 4	38.8
Finger width = 1 (equalNP)	14.0
Finger width = 2 (equalNP)	25.3

Table 5-10: Configuration of SSIA_B (lowest area version).

	IW_1	**IW_2**	**IW_3**	**IW_4**
S_0	5X1_F1_FO5	5X1_F1_FO5	5X1_F1_FO5	5X1_F1_FO5
S_1	4X1_F1_FO5	4X1_F1_FO5	4X1_F1_FO5	4X1_F1_FO5
S_2	4X1_F1_FO4	4X1_F1_FO4	4X1_F1_FO4	4X1_F1_FO4
S_3	5X1_F1_FO5	5X1_F1_FO5	5X1_F1_FO5	5X1_F1_FO5
S_4	3X1_F1_equalN	3X1_F1_equalN	3X1_F1_equalN	3X1_F1_equalN

	P_FO3	P_FO3	P_FO3	P_FO3
S_5	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3
S_6	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3
S_7	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3	3X1_F1_equalN P_FO3

Table 5-11: Performance parameters of SSIA_B (lowest area version).

Parameters	**SSIA_A**	**SSIA_B**	**% difference**
Area	598 um^2	572 um^2	4.34 % (↓)
Delay (worst-case)	602 ps	602 ps	N/A
Static - Power Consumption	420 nW	360 nW	14 % (↓)
Dynamic – Energy Consumption	192 fJ	184 fJ	4.16 % (↓)

5.3.2 Delay

As explained in the methodology section, finding the SSIA version that has the lowest worst-case delay is slightly complicated compared to other parameters. **Table 5-2** earlier on page number 117 presented the delay data for SSIA_A, which is baseline data for SSIA. It was observed in section 4.5 and **Table 4-13** that it is advantageous to go from finger width 2 to 3 for most components, so implementing SSIA with MOSFETs of finger width 3 should be good. The resulting performance of this configuration is

presented in **Table 5-12** below. It can be seen that the worst-case delay has gone down to 376ps from 602ps (bold and underlined in **Table 5-12**), a reduction of 37.5% in the delay. In section 4.5 and **Table 4-13**, it was also observed that there wasn't any improvement going from finger width 2 to 3 for 3X1 multiplexer and 4X1 multiplexer (which has a fanout of four). Thus replacing those components with finger width 2 should also offer the same performance (or better). The new configuration due to this change, SSIA_C, is shown in **Table 5-14**, and its resulting performance is presented in **Table 5-13** (bold and underlined text represents the improvement). Although the area isn't a priority, that too improves, because we have moved back from finger width three to two. This is because we have downgraded non-critical components to a slower speed and smaller area without affecting the critical path.

Table 5-12: Cumulative delay of SSIA version made up of finger width 3 MOSFETs.

	IW_1	**IW_2**	**IW_3**	**IW_4**
S_0	94	188	282	**<u>376</u>**
S_1	88	182	275	369
S_2	86	180	274	368
S_3	94	188	282	**<u>376</u>**
S_4	79	173	267	360
S_5	79	157	251	345

S_6	79	157	236	330
S_7	79	157	236	315

Table 5-13: Cumulative delay of SSIA_C components.

	IW_1	**IW_2**	**IW_3**	**IW_4**
S_0	94	188	282	**376**
S_1	88	182	275	369
S_2	87	181	274	368
S_3	94	188	282	**376**
S_4	77	171	265	**359**
S_5	77	154	248	**342**
S_6	77	154	231	**325**
S_7	77	154	231	**308**

Table 5-14: Configuration of SSIA_C (lowest delay version).

	IW_1	**IW_2**	**IW_3**	**IW_4**
S_0	5X1_F3_FO5	5X1_F3_FO5	5X1_F3_FO5	5X1_F3_FO5
S_1	4X1_F3_FO5	4X1_F3_FO5	4X1_F3_FO5	4X1_F3_FO5
S_2	4X1_F2_FO4	4X1_F2_FO4	4X1_F2_FO4	4X1_F2_FO4
S_3	5X1_F3_FO5	5X1_F3_FO5	5X1_F3_FO5	5X1_F3_FO5
S_4	3X1_F2_FO3	3X1_F2_FO3	3X1_F2_FO3	3X1_F2_FO3
S_5	3X1_F2_FO3	3X1_F2_FO3	3X1_F2_FO3	3X1_F2_FO3

S_6	3X1_F2_FO3	3X1_F2_FO3	3X1_F2_FO3	3X1_F2_FO3
S_7	3X1_F2_FO3	3X1_F2_FO3	3X1_F2_FO3	3X1_F2_FO3

Table 5-15: Performance parameters of SSIA_C (lowest delay version).

Parameters	**SSIA_A**	**SSIA_C**	**% difference**
Area	598 um^2	1181 um^2	197.5 % (↑)
Delay (worst-case)	602 ps	376 ps	37.5 % (↓)
Static - Power Consumption	420 nW	3104 nW	739 % (↑)
Dynamic – Energy Consumption	192 fJ	382 fJ	199 % (↑)

The hybrid versions and equalNP can't reduce the worst-case delay any further. But it's possible to obtain a better version which will have lesser area and power consumption then this version keeping the delay parameter same. It can be obtained by deliberately slowing the faster components to an extent that they don't worsen the worst-case delay parameter. This is discussed in the next section (5.4) of this chapter.

5.3.3 Power

Power/energy consumed increases with increase in the finger width and it's also relatively proportional to area, so SSIA_B, the version of SSIA which occupies the least area, also consume the lowest power.

5.3.4 SSIA Predictor

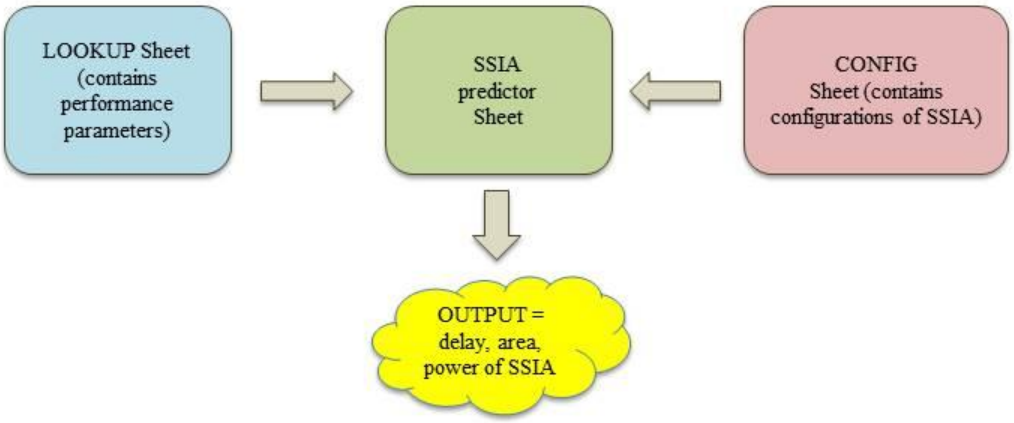


Figure 5-5: Simple flow chart of SSIA Predictor.

As already explained, there are so many different versions of SSIA, based on various designed components that it will be useful to have a tool to explore this design space. The SSIA Predictor is a tool that can be used for this exploration. It is a Microsoft Excel spreadsheet/dashboard consisting of several sheets working together:

- The CONFIG sheet contains a list of possible SSIA configurations – each configuration is simply a set of component numbers for SSIA.
- The LOOKUP sheet contains data for the area, delay, and power/energy consumption for components that make up the SSIA. The current version of SSIA Predictor has a total of 18 components. This lookup sheet is shown in **Table 5-16**.
- The SSIA predictor sheet looks up the data for each component specified in the configuration chosen from the CONFIG sheet and calculates area, delay, and

power/energy consumption according to the methodology detailed earlier in section 5.2.

Using the SSIA predictor, it is possible to quickly calculate the data for any valid SSIA configuration, the data such as area, delay, and power/energy consumption of SSIA. A user can also interactively modify the configuration to get closer to the desired set of parameters (e.g. improve area, power or delay). This was also used to verify our manual calculation of SSIA.

The SSIA predictor can be loosely contrasted to register file generators, such as [67]. SSIA will be part of Stack Processor and will be used to store data within the processor. In the FPGAs, the equivalent is on-chip block-RAM(BRAM) – BRAMs in FPGA is used to store data which can have various usage. Though there is a subtle difference between both. In the SSIA Predictor, user can provide their priority/requirement for the area, delay and power. Whereas, in the MPRFGEN of [67] user can provide parameters of register files such as Read mode, number of write ports, number of read ports, Bitwidth of each memory entry and Memory size. They are different because the user can provide their priority/requirement for the area, delay and power of register files.

Table 5-16: Lookup Table

Components (↓)	Area (um^2)	Delay (pS)	Static Power (nW)	Dynamic Energy (fJ)
5X1_F4_FO5	60.4	98.4	308.36	26.21
5X1_F3_FO5	51.7	93.9	172.87	16.69
5X1_F2_FO5	44.4	96.1	87.31	11.21
5X1_F1_FO5	23.8	150.6	17.08	7.38
4X1_F4_FO4	49.5	89.5	251.46	25.32
4X1_F3_FO4	42.3	86.4	140.87	15.64
4X1_F2_FO4	36.4	86.6	70.91	10.09
4X1_F1_FO4	19.7	130.5	13.88	6.22
4X1_F4_FO5	49.5	91.5	251.46	25.69
4X1_F3_FO5	42.3	87.6	140.87	16.15
4X1_F2_FO5	36.4	88.8	70.91	10.66
4X1_F1_FO5	19.7	137.4	13.88	6.80
3X1_F4_FO3	38.5	80.2	194.56	24.42
3X1_F3_FO3	32.9	78.7	108.87	14.58
3X1_F2_FO3_equalNP	25.3	76.9	36.60	7.29
3X1_F2_FO3	28.3	77	54.62	4.56
3X1_F1_FO3_equalNP	14	117.7	6.85	5.06
3X1_F1_FO3	15.6	110.2	10.68	5.06

5.4 Automated selection for SSIA using DARWIN

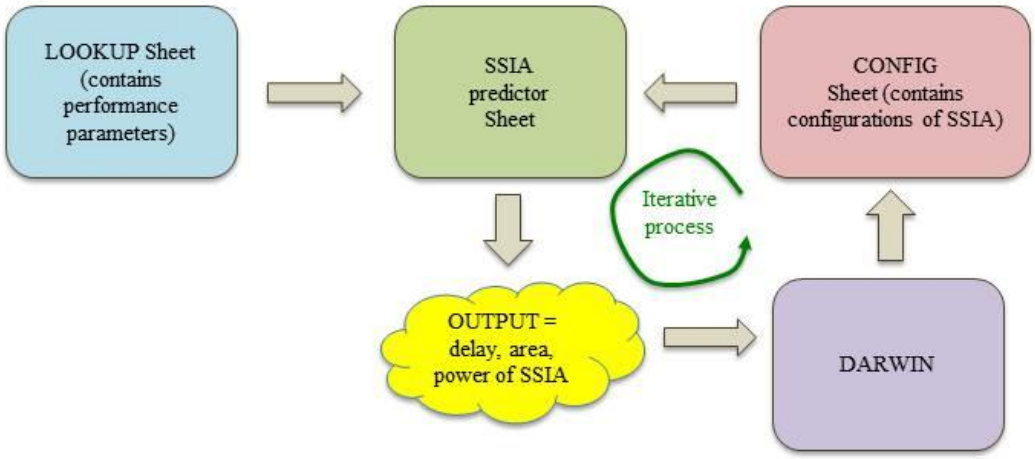


Figure 5-6: Simple flow chart of SSIA Predictor and DARWIN.

This section starts with an explanation of a component called DARWIN - **D**igital **A**rchitecture **R**efinement **W**ith **IN**heritance. The component can make a recommendation similar to what was made in the previous section but using an algorithm rather than manual experimentation. The new recommendations have been generated using DARWIN and have been analyzed as well as compared with the old recommendation from the previous section.

5.4.1 Section DARWIN

The SSIA predictor can be extended by an additional component – DARWIN. The single performance factor is rarely a priority, most of the times all three factors are important with different priorities. In our tool we can specify the weight for multiple factors (area and delay OR area and power OR delay and power OR all three of them)

and the tool will make the recommendation accordingly based on the user's requirement. DARWIN starts with a base model (SSIA_A) and calculates the *cost and score* and comes to the recommendation using a *search algorithm* – explanation of both are provided below.

- **Cost and Score**: The default configuration of the SSIA is based upon the standard cell components before any optimisation work is carried out. That is the baseline. The area, power and delay parameter of this configuration is used as a normalisation function. Such that for a given parameter, the cost for a given configuration is:

$$C = \frac{Mc}{Ms} \qquad \textbf{Eq. 5-4}$$

  Where C is the cost, Mc is the parameter metric measured for the current configuration and Ms is the same parameter metric for the standard configuration. This ensures that all three parameter metrics are expressed as costs in terms of a number of the same range (typically <=1.0). By normalizing the ranges, it is then possible to combine costs using a weight applied to each cost. For example, where only speed is important, the weight for delay would be 100% and 0% for the other components, if speed and area are of equal importance they might be weighted 50%/50%, and so on. The sum of the weighted, normalized, costs is the score of the function.

- **Search Algorithm**: DARWIN generates various SSIA configurations, in a closed loop, such that each new variation results in a new set of outputs from the SSIA Predictor. If the score of the new output is better than the previous ones, then the

new configuration is kept. If the new configuration is worse than the previous one, it is discarded. DARWIN implements a very simple hill-climbing algorithm, effectively a primitive evolutionary algorithm, by randomly 'mutating' small numbers of SSIA components. At each iteration, variations in performance can be discovered, some better, some worse. By only keeping the better mutations, the configuration gradually improves, according to the specified criteria [68].

Importantly, automation means that solutions to complex cases can be discovered easily. For example, it would be very difficult to manually find a solution that offers good delay but also finds moderate area and power. DARWIN is good at exploring the design space to identify particular trade-off solutions.

The previous section used SSIA of depth 8, whereas this section considers SSIA of depth 16 to be closer to the realistic scenario. Also, the calculations here are for full 32 bits unlike previous section (**Table 5-8**) where the calculations were only for a single bit. 32-bit data corresponding to **Table 5-8** is shown in **Table 5-17** below. DARWIN is also capable of minimizing the spread of worst-case and best-case delay but these explanations have been omitted for the purpose of simplicity.

Table 5-17: Performance parameters of SSIA_A (the standard cell versions).

Performance Parameters	**Depth = 8; 1 bit**	**Depth = 16; 32 bits**
Area	598 um^2	35098 um^2
Delay (worst-case)	602 ps	602 ps
Static - Power Consumption	420 nW	24386 nW
Dynamic – Energy Consumption	192 fJ	11326 fJ

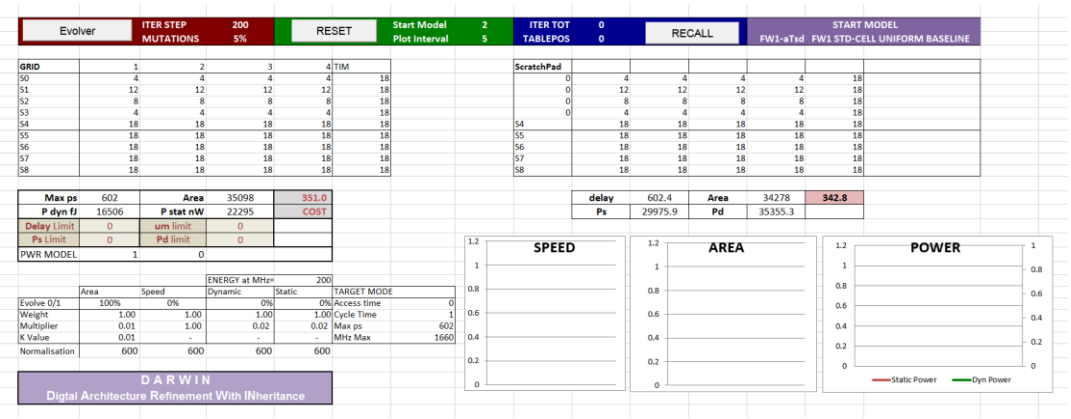


Figure 5-7: Screenshot of DARWIN (State - Reset).

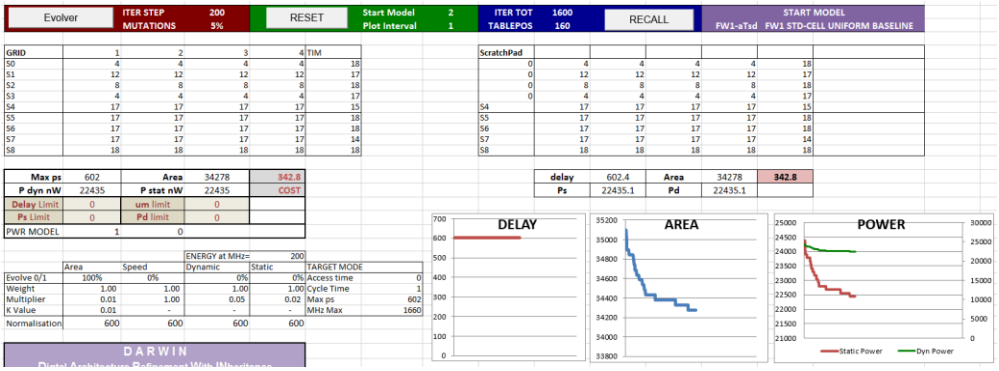


Figure 5-8: Screenshot of DARWIN (State – Evolved for Area).

5.4.2 Comparison with manual selections

Through this sub-section, versions of SSIA are recommended which are best for each performance parameter, but this time using DARWIN, i.e. version of SSIA that will:

- Occupy the least area.
- Have the smallest worst-case delay.
- Have the least power consumption.

The following sub-sub-sections present the results of DARWIN (and also give a glimpse of the tool).

5.4.2.1 *Area*

This sub-sub-section presents the result of an experiment when the SSIA predictor is set to evolve SSIA_A so that a new version that has the least area. We get SSIA_B as the output from the SSIA Predictor, the same version we get manually. It is expected that we get the same results as the area is fairly easy to optimize. The graph for this evolution is shown in **Figure 5-9** below and performance parameters are presented in **Table 5-18** below.

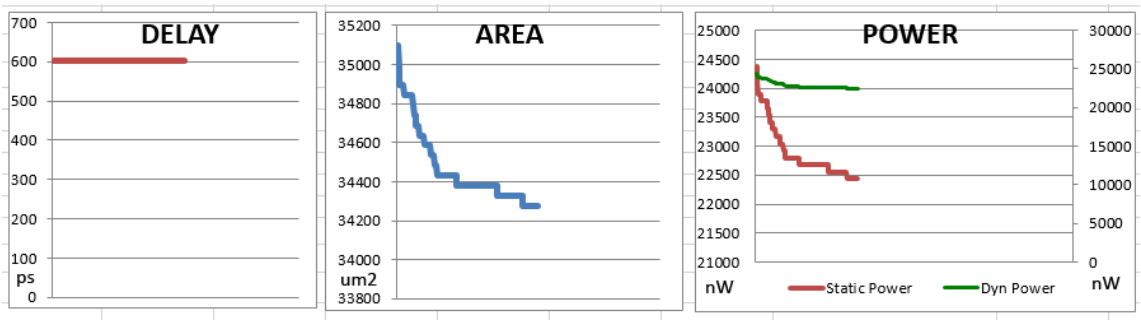


Figure 5-9: Graph of SSIA_A evolution for area optimization.

Table 5-18: Performance parameters of SSIA_B (lowest area version).

Parameters	**SSIA_A**	**SSIA_B**	**% difference**
Area	35,098 um^2	34,278 um^2	2.33 % (↓)
Delay (worst-case)	602 ps	602 ps	-
Static - Power Consumption	24,386 nW	22,435 nW	8.00 % (↓)
Dynamic – Energy Consumption	11,326 fJ	11,071 fJ	2.25 % (↓)

5.4.2.2 *Delay*

This sub-sub-section presents the result of an experiment when the SSIA predictor is set to evolve SSIA_A so that a new version that has the least delay. We got SSIA_D as the output. It's different from the one we came up manually and is also not better in performance compared to SSIA_C – the version that we derived manually for the optimized delay. Though it may possible to obtain the same or better result if we run DARWIN long enough. This is one of the limitations of DARWIN and can be improved with a more sophisticated algorithm – something to investigate for future work. The graph for this evolution is shown in **Figure 5-10** below, the configuration in **Table 5-19** and the performance parameter is presented in **Table 5-20**.

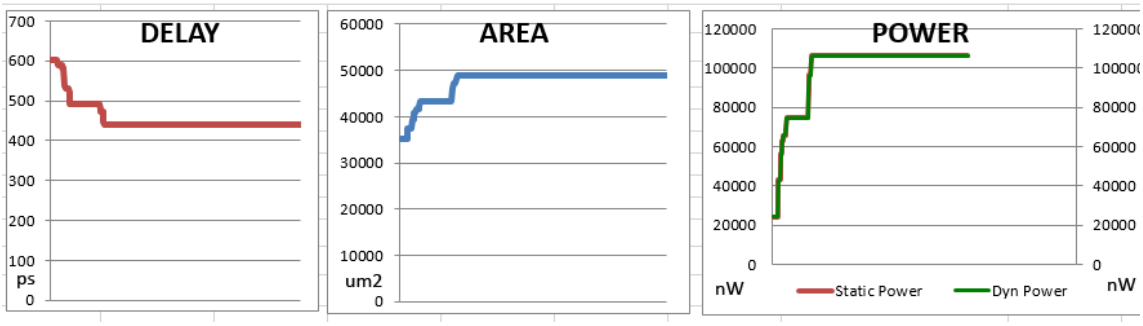


Figure 5-10: Graph of SSIA_A evolution for delay/speed optimization.

Table 5-19: Configuration of SSIA_D (A DARWIN Recommendation).

	IW_1	**IW_2**	**IW_3**	**IW_4**
S_0	5X1_F3_FO5	5X1_F4_FO5	5X1_F2_FO5	5X1_F1_FO5
S_1	4X1_F4_FO5	4X1_F4_FO5	4X1_F4_FO5	4X1_F1_FO5
S_2	4X1_F3_FO4	4X1_F2_FO4	4X1_F2_FO4	4X1_F1_FO4

S_3	5X1_F3_FO5	5X1_F4_FO5	5X1_F2_FO5	5X1_F3_FO5
S_4	3X1_F2_FO3	3X1_F3_FO3	3X1_F2_FO3_equalNP	3X1_F2_FO3
S_5	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3
S_6	3X1_F1_FO3	3X1_F2_FO3_equalNP	3X1_F1_FO3	3X1_F1_FO3
S_7	3X1_F1_FO3	3X1_F1_FO3	3X1_F1_FO3	3X1_F4_FO3

Table 5-20: Performance parameters of SSIA_D.

Parameters	**SSIA_A**	**SSIA_C**	**SSIA_D**	**A & D % difference**
Area (in um^2)	35,098	66,778	**48,810**	139 % (↑)
Delay (worst-case) (in ps)	602	376	**441**	27% (↓)
Static - Power Consumption (nW)	24,386	155,253	**106,458**	436 % (↑)
Dynamic – Energy Consumption	11326 fJ	21,404 fJ	**17,372 fJ**	153 % (↑)

5.4.2.3 *Power/Energy*

This sub-sub-section presents the result of an experiment when the SSIA predictor is set to evolve SSIA_A so that a new version that has the least power/energy consumption. We got SSIA_B as the output for the same reason as what was mentioned in sub-section 5.3.3 of Manual selection of SSIA. Power/energy consumed increases with

increase in the finger width and it's also relatively proportional to area, so SSIA_B, the version of SSIA which occupies the least area, also consume the lowest power.

The SSIA predictor, which uses DARWIN, suggested the same version of SSIA for area parameter and power parameter, but for delay parameter, the tool couldn't suggest the same or better solution than what was derived manually. But DARWIN can be helpful in reducing the area of the version which has the least worst-case delay – experiments of this kind have been presented in the next sub-section (5.4.3).

5.4.3 Analysis of DARWIN Recommendations

5.4.3.1 *Best delay, better area*

This sub-sub-section presents the result of an experiment when the SSIA predictor is set to evolve SSIA_C, the version that has the least propagation delay. In the DARWIN, we have a *limit* function. The function can limit the value of a performance parameter. In our case, we limit the delay at 376 ps. After evolving, DARWIN recommends SSIA_F. The graph for this evolution is shown in **Figure 5-11** below, the configuration is in **Table 5-21**, and performance parameters are presented in **Table 5-22**.

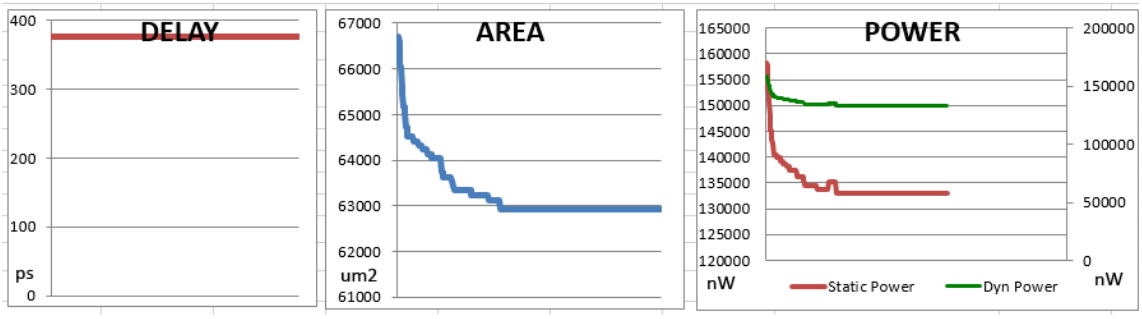


Figure 5-11: Graph of SSIA_C evolution for area optimization.

Table 5-21: Configuration of SSIA_F (A DARWIN Recommendation).

	IW_1	**IW_2**	**IW_3**	**IW_4**
S_0	5X1_F3_FO5	5X1_F3_FO5	5X1_F3_FO5	5X1_F3_FO5
S_1	4X1_F2_FO5	4X1_F2_FO5	4X1_F2_FO5	4X1_F2_FO5
S_2	4X1_F2_FO4	4X1_F2_FO4	4X1_F2_FO4	4X1_F2_FO4
S_3	5X1_F3_FO5	5X1_F3_FO5	5X1_F3_FO5	5X1_F3_FO5
S_4	3X1_F2_FO3_equalNP	3X1_F2_FO3_equalNP	3X1_F2_FO3_equalNP	3X1_F2_FO3_equalNP
S_5	3X1_F2_FO3_equalNP	3X1_F2_FO3_equalNP	3X1_F2_FO3_equalNP	3X1_F1_FO3
S_6	3X1_F2_FO3_equalNP	3X1_F2_FO3_equalNP	3X1_F1_FO3	3X1_F1_FO3
S_7	3X1_F2_FO3_equalNP	3X1_F2_FO3_equalNP	3X1_F1_FO3	3X1_F4_FO3

Table 5-22: Performance parameters of SSIA_F and SSIA_C.

Parameters	**SSIA_C**	**SSIA_F**	**% difference**
Area	66,778 um^2	62,934 um^2	7 % (↓)
Delay (worst-case)	376 ps	376 ps	-
Static - Power Consumption	155,253 nW	132,937 nW	14% (↓)

Dynamic – Energy Consumption	21,404 fJ	19,486 fJ	9% (↓)

5.4.3.2 *Better delay, better area, better power*

This sub-sub-section presents the result of an experiment when the SSIA predictor is set to evolve SSIA_F, the version that has the best delay & better area. This time the limit is a little bit relaxed, 400 ps, to reduce the area and power. After evolving, DARWIN recommends SSIA_G. The graph for this evolution is shown in **Figure 2-3** below, the configuration is in **Table 5-23** and performance parameters are presented in **Table 5-24**

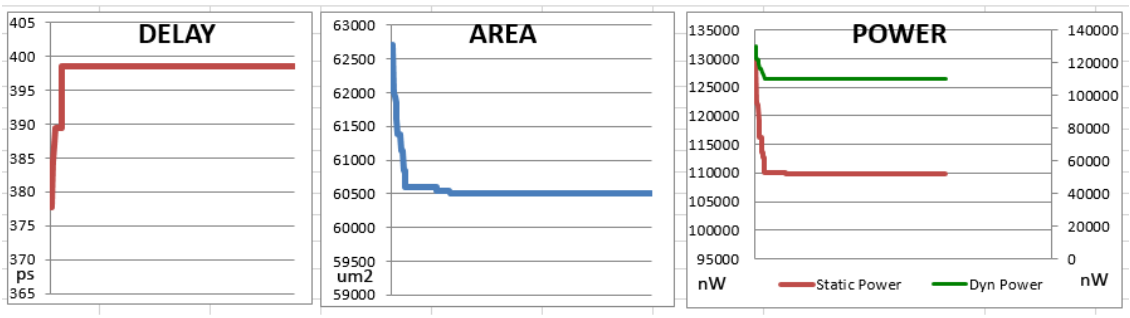


Figure 5-12: Graph of SSIA_F evolution for better area and speed.

Table 5-23: Configuration of SSIA_G (A DARWIN Recommendation).

	IW_1	**IW_2**	**IW_3**	**IW_4**
S_0	5X1_F2_FO5	5X1_F2_FO5	5X1_F2_FO5	5X1_F2_FO5
S_1	4X1_F2_FO5	4X1_F2_FO5	4X1_F2_FO5	4X1_F2_FO5
S_2	4X1_F2_FO4	4X1_F2_FO4	4X1_F2_FO4	4X1_F2_FO4
S_3	5X1_F2_FO5	5X1_F2_FO5	5X1_F2_FO5	5X1_F2_FO5

S_4	3X1_F2_FO3_eq ualNP	3X1_F2_FO3_eq ualNP	3X1_F2_FO3_eq ualNP	3X1_F1_FO3
S_5	3X1_F2_FO3_eq ualNP	3X1_F2_FO3_eq ualNP	3X1_F2_FO3_eq ualNP	3X1_F1_FO3_eq ualNP
S_6	3X1_F2_FO3_eq ualNP	3X1_F2_FO3_eq ualNP	3X1_F1_FO3_eq ualNP	3X1_F1_FO3_eq ualNP
S_7	3X1_F2_FO3_eq ualNP	3X1_F2_FO3_eq ualNP	3X1_F1_FO3_eq ualNP	3X1_F1_FO3_eq ualNP

Table 5-24: Performance parameters of SSIA_G.

Parameters	**SSIA_C**	**SSIA_F**	**SSIA_G**	**C & F % difference**
Area (in um2)	66778	62934	**60499**	9 % (↓)
Delay (worst-case) (in ps)	376 ps	376 ps	**399 ps**	6 % (↑)
Static - Power Consumption (nW)	155253	132937	**109807**	29 % (↓)
Dynamic – Energy Consumption	21,404 fJ	19,486 fJ	**17,932 fJ**	16 % (↓)

5.4.4 Summary

Sections 5.3 and 5.4 presented 6 different versions of SSIA each offers different benefits at different cost. All of these are optimized for a single parameter. This investigation can be extended by optimizing SSIA for multiple performance parameters (i.e. different permutation and combinations of the area, delay, and power). Thus, by using DARWIN, even better version of SSIA could be obtained. The next section ()

revisits the Crispin-Bailey and Mullane comparison with register file using this newly optimized versions of SSIA.

5.5 Comparison with Previous SSIA implementations

Bailey and Mullane have compared their SSIA implementations with register files. As the vast majority of mainstream processors use register files, the comparison is a necessity for the investigation of SSIA. As the implementations of SSIA and register files are different, the comparison is difficult and required them to make an assumption to establish the common ground, e.g. *if we have a total of eight stack elements, the question is raised: is this SSIA equivalent to a register file of eight registers?*

Answer to the above question, as well as the rationale for comparison has been established before [1], the investigation in this thesis builds upon that. It was established that SSIA with issue width 1, 2, 3 and 4 are equivalent to 3, 6, 9 and 12 superscalar register ports respectively. In a typical superscalar register machine, there is one write port for every two read ports. In SSIA, we have two read ports for each issue width ($\{x, y\}_1$, $\{x, y\}_2$, $\{x, y\}_3$, $\{x, y\}_4$ in **Figure 5-1**) same as register files.

The investigation involves a comparison of *Cycle Time* and *Access Time* of register files and SSIA. For SSIA *cycle time* is the time it requires to complete the full cycle and *access time* is the time needed to access the data.

In our comparison, we use FO4 delay metrics as they are independent of technologies [69]. Delay fluctuates a lot, and to counter this problem, a standardized delay metric (FO4 delay) has been developed and its fluctuation is rather less [70]. *One FO4 delay* unit is equivalent in the delay of an inverter that is driving other 4 inverters of the same size [71]. This value in our case is **44** ps. Cycle time and access time for SSIA

can be obtained from SSIA predictor. These values in terms of FO4 delay metrics are provided in **Table 5-26** and **Table 5-27**.

The original experiment by Crispin-Bailey and Mullane showed that *Cycle Time* using UMC 90nm technology was comparable to a population of register files under the comparable read/write port equivalence explained above. The same finding was observed for *Access Time* as a function of superscalar register read ports. Data, provided by Crispin-Bailey has been presented in **Table 5-25** below, and the corresponding graph is shown in **Figure 5-13** below.

Table 5-25: Cycle time and Access time for SSIA designed in UMC 90nm.

Superscalar Register Ports (Issue Width)	**Access Time/ Read Time (in number of FO4s)**	**Cycle Time (in number of FO4s)**
3 (1)	0	8.8
6 (2)	4.3	13.1
9 (3)	8.7	17.3
12 (4)	13	21.8

Data have been provided by Crispin-Bailey.

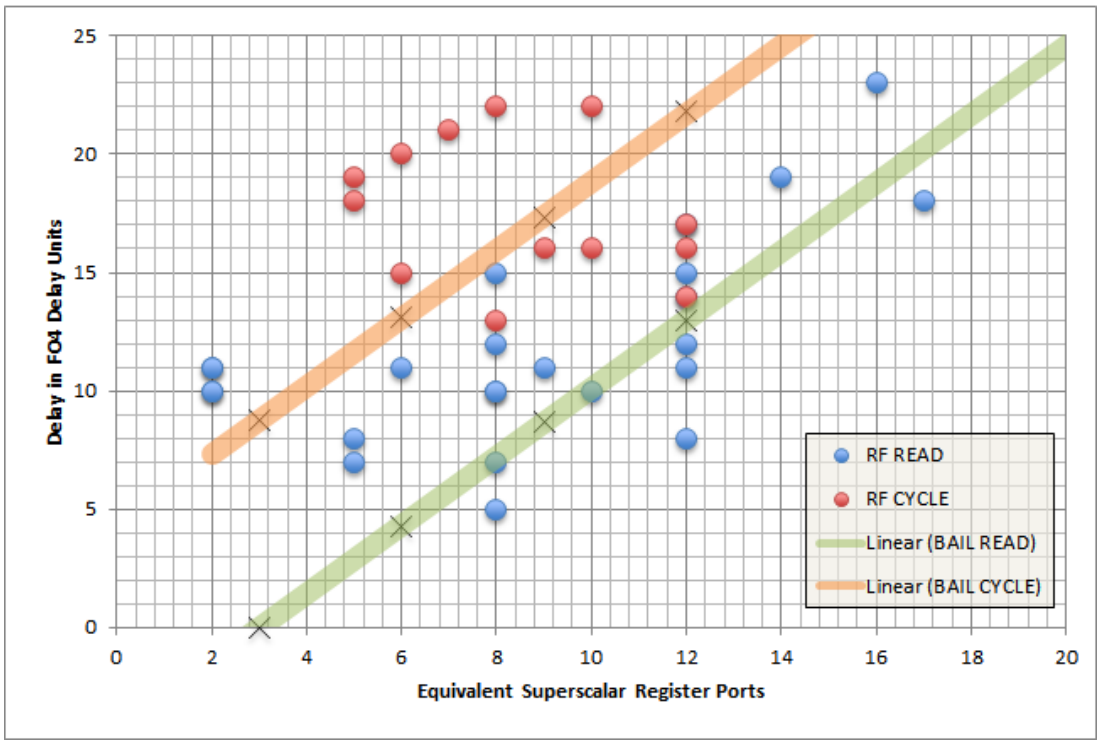


Figure 5-13: SSIA and register file comparison by Bailey & Mullane (2014).

In **Figure 5-13**,

- *RF Read* stands for Register File Access Time/Read Time
- *RF Cycle* stands for Register File Cycle Time
- *Linear (BAIL READ)* stands for Access Time/Read time of SSIA designed in UMC 90 nm by Crispin-Bailey and Mullane.
- *Linear (BAIL CYCLE)* stands for Cycle time of SSIA designed in UMC 90 nm by Crispin-Bailey and Mullane.

The corresponding data for Standard Cell SSIA (SSIA_A) version, obtained from SSIA Predictor, are presented in **Table 5-26** and **Figure 5-14** below. *Cycle Time* by

Crispin-Bailey and Mullane is very optimistic, potentially due to temperature and fanouts condition was assumed ideal.

Table 5-26: Cycle time and Access time for SSIA_A.

Superscalar Register Ports (Issue Width)	Access Time/ Read Time (in number of FO4s)	Cycle Time (in number of FO4s)
3 (1)	0.0	7.3
6 (2)	3.4	10.7
9 (3)	6.8	14.1
12 (4)	10.3	17.5

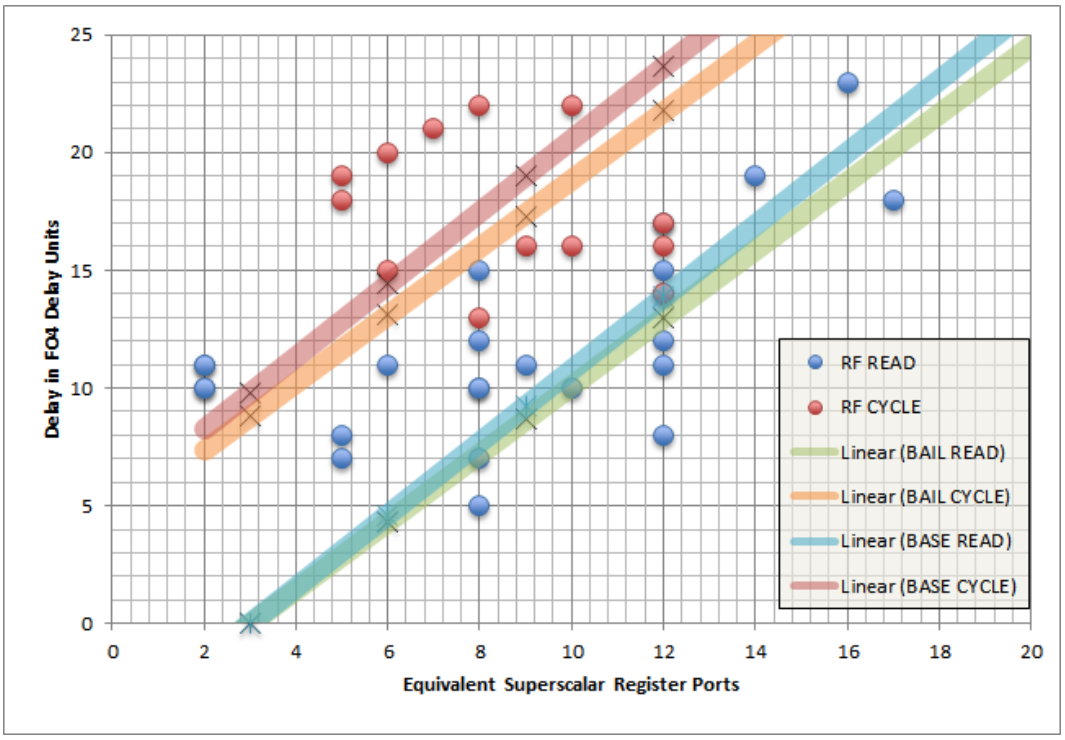


Figure 5-14: SSIA_A superimposed over Bailey & Mullane comparison graph.

In **Figure 5-14**,

- *LINEAR(BASE READ)* is Read Time for SSIA_A
- *LINEAR(BASE CYCLE)* is Cycle Time for SSIA_A

It was found out in the previous section that finger width 3 offers very good delay performance, and hence the SSIA version made up of finger width 3 transistors (let's call it SSIA_H) should offer better performance than SSIA_A. We have presented cycle time and access time data for SSIA_H **Table 5-27** below, and the graph is shown in **Figure 5-15** below. **Figure 5-16** shows only SSIA_H data and register files data. It can be seen that SSIA_H data are far lower in the graph signifying that this is a very competitive circuit in comparison to register files and also gives a direct proof for the hypothesis that SSIA can be optimized by use of advanced VLSI design techniques.

In **Figure 5-15** and **Figure 5-16**

- *LINEAR(FW3 READ)* is Read Time for SSIA_H
- *LINEAR(FW3 CYCLE)* is Cycle Time for SSIA_H

Table 5-27: Cycle time and Access time for SSIA_H.

Superscalar Register Ports (Issue Width)	**Access Time/ Read Time (in number of FO4s)**	**Cycle Time (in number of FO4s)**
3 (1)	0.0	6.0
6 (2)	2.1	8.1
9 (3)	4.3	10.2
12 (4)	6.4	12.4

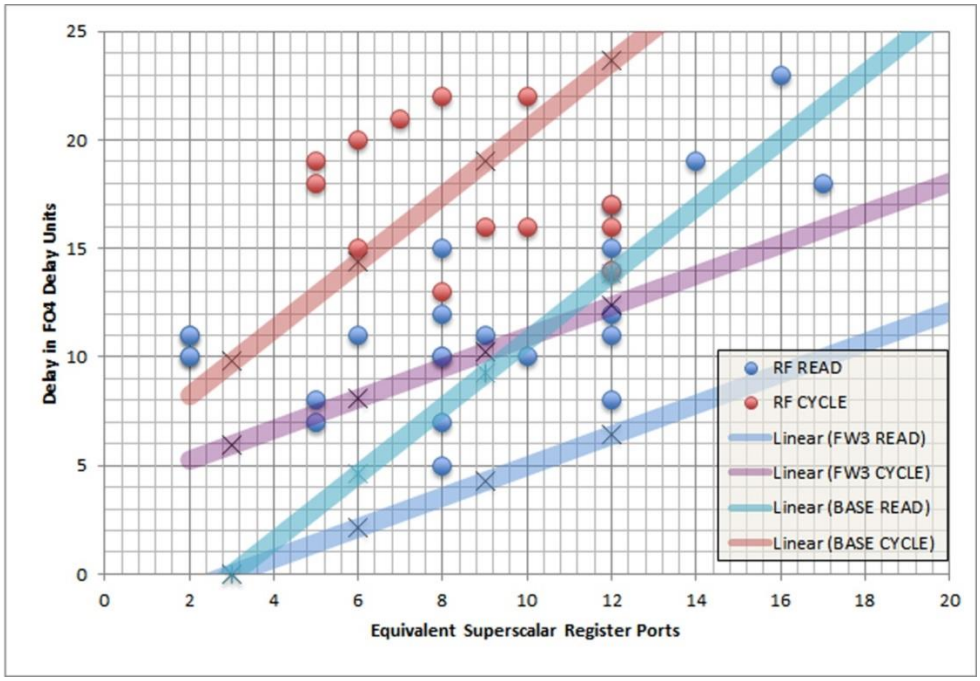


Figure 5-15: Comparison graph of SSIA_A, SSIA_H and register file.

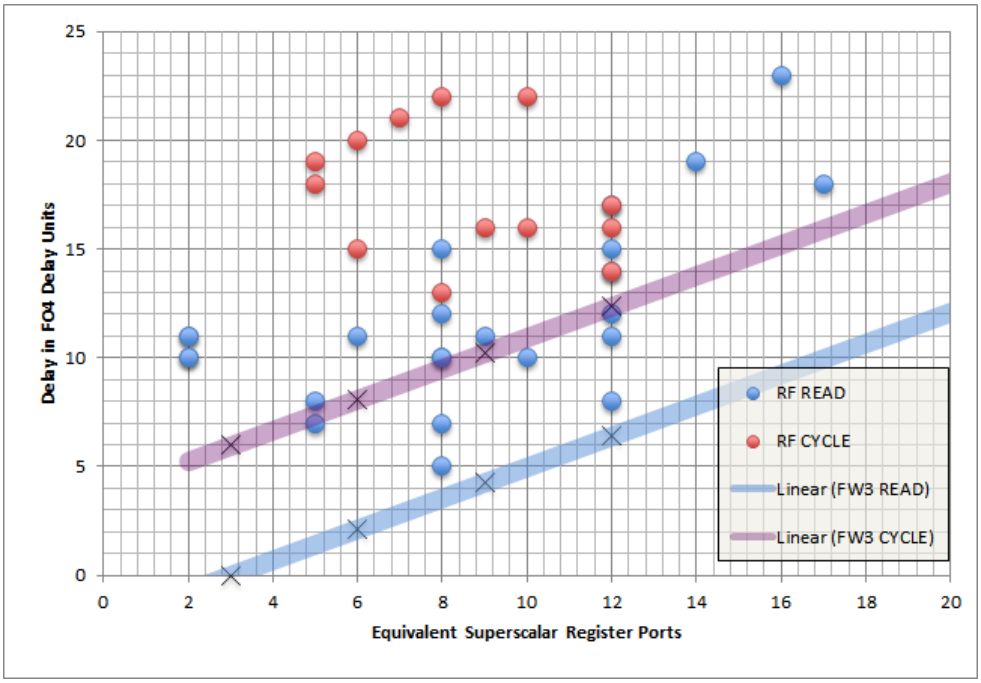


Figure 5-16: Comparison graph of SSIA_H and register files.

5.6 Summary

It was observed in the previous chapter that suitable component combinations can achieve high speed at the expense of the area. This chapter shows that the same high speeds can also be achieved with a little lesser area than the 'naive' solutions, by exploiting the individual members of the component library. Power consumption has been investigated which is an important performance parameter. The use of an algorithmic approach (and SSIA Predictor) also yields solutions that balance specific design priorities (combinations of the area, delay, and power) in a way that is difficult to figure out through a manual selection process. Register files are used widely in mainstream processor design hence it's important to provide the comparison of our SSIA and register files. Plus, the comparison suggests that using custom design previous limitation of SSIA, highlighted by Crispin-Bailey and Mullane, can be overcome.

The chapter started with an explanation of the methodology for calculating the performance parameters (Area, Delay and Power/Energy) of SSIA using SSIA_A version as an example. The power consumption at SSIA level wasn't explored before but it's an important parameter of performance measurement hence it is a novel and important part of the investigation of section 5.2. Section 5.3 recommended version of SSIA which are suitable for each performance parameter and those recommendations were improved further by using the SSIA Predictor spreadsheet and DARWIN. Overall this chapter shows that the high speeds can also be achieved with a little lesser area than the 'naive' solutions, by exploiting the individual members of the component library. Lastly, the comparison with register files helps establish the credibility of SSIA in relation to mainstream architecture and provided direct evidence that proves the hypothesis.

CHAPTER 6

EVALUATING FUTURE OPPORTUNITIES FOR ADVANCED VLSI DESIGN TECHNIQUES

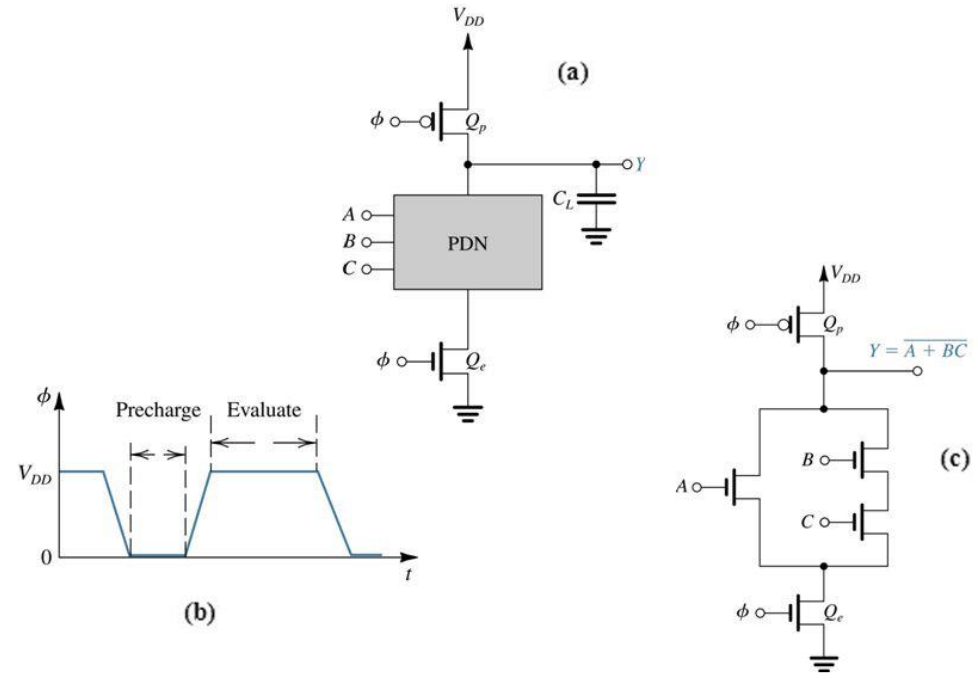
A review of advanced VLSI optimization techniques has been provided earlier in section 2.2.2 of the literature review chapter of the thesis. The preliminary experiments of some of these techniques are presented in this chapter. This preliminary experiment paves the path for more optimization of SSIA in future. The techniques include dynamic-logic, power gating and current mode implementations.

Contributions made through this chapter are presented in the list below.

- A VLSI Optimization technique - dynamic-logic - is examined for tri-state buffer and its benefits are presented.
- Another VLSI Optimization technique – power gating - is examined for multiplexers and its impact on propagation delay is investigated. Power gating causes ground bounce and their measurement has been done in our experiments. Though it has been minimized using suitable design techniques.
- The improvements in performance that can be achieved by the use of current-mode techniques are predicted.

6.1 Dynamic-logic

Dynamic-logic and pre-charging were briefly explained earlier in section 2.2.2.



(a) basic structure of dynamic-MOS logic circuits; **(b)** waveform of the clock needed to operate the dynamic-logic circuit; and **(c)** an example circuit.

Figure 6-1: Dynamic-logic schematic, waveform and an example circuit [72].

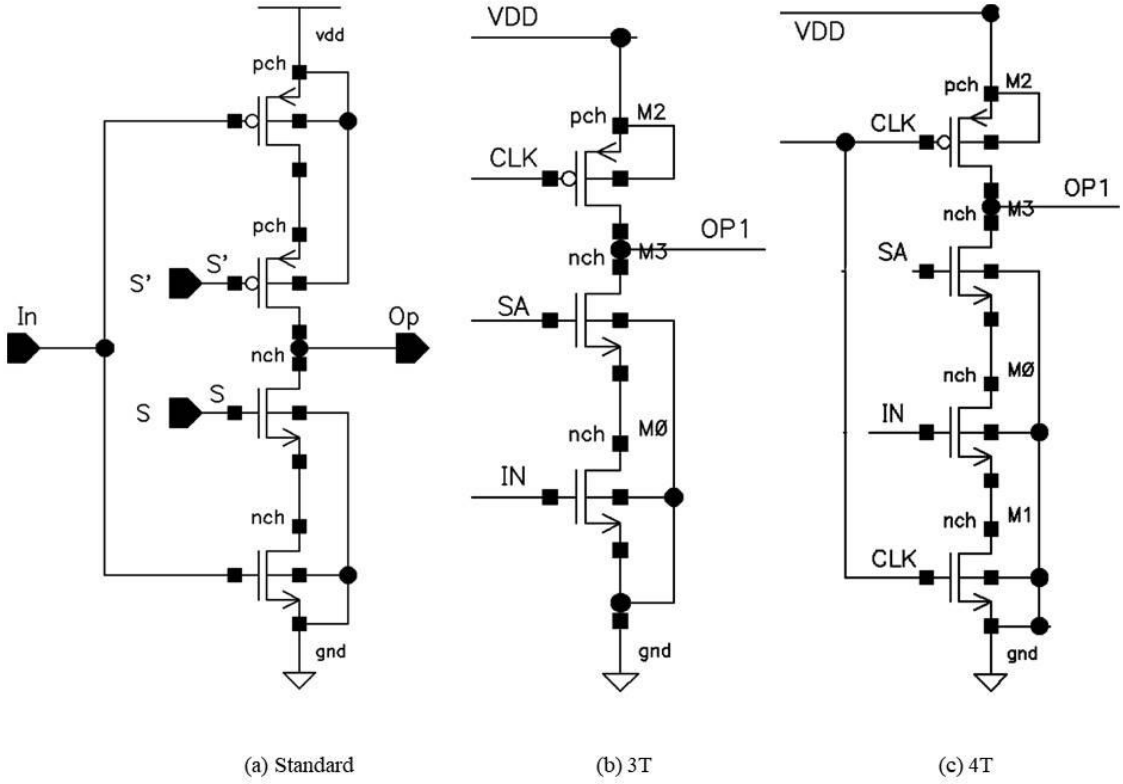


Figure 6-2: Schematics of tristate buffers (a) Standard tristate buffer, (b) 3T tristate buffer and (c) 4T tristate buffer.

In our previous experiments static CMOS implementation of inverter and tri-state buffer have been used. In theory, use of dynamic-logic – instead of static logic – should offer better speed and reduced power consumption [72]. One of the main things that differ in dynamic-logic is the storage of signal voltages in parasitic capacitance. A typical dynamic-MOS logic circuit is shown in **Figure 6-1**. Dynamic-logic circuit also requires a clock (ø) which is fed to the p-channel transistor at the top, and to the n-channel transistor at the bottom. When the clock input is low it's a pre-charge phase of the circuit and otherwise, it's an evaluation phase. In this chapter, two different dynamic-logic implementations of tri-state buffer have been proposed which are shown in **Figure 6-2**.

These tri-state buffers have been simulated to measure propagation delay data and data have been presented in **Table 6-1**. It can be seen that the worst-case delay increases, but overall performance of delay should improve as the delay due to rise time is nil, dynamic logic must have reduced the area and power consumption as well because pull up network is replaced with single transistor.

Table 6-1: Rise time and fall time propagation delay of various kinds of tri-state buffers.

Tristate	Rise Time	Fall Time
Standard	49 ps	47 ps
4T Tristate	0 ps (Pre-charged)	58 ps
3T Tristate	0 ps (Pre-charged)	38 ps

From the above data, it can be said that worst-case delay in dynamic-logic is significantly lower compared to standard tri-state made up of finger width 1. Given more time and investigation data of multiplexer and SSIA designed using this tri-state buffer can be obtained. A slightly less or similar improvement can be safely assumed for the multiplexers and SSIA.

6.2 Power gating

Power gating is a technique of turning off the inactive parts of the circuit, in order to reduce the leakage current and unnecessary power consumption of the circuit. The benefits come at expense of complexity, increase in area, increase in delay of the circuit and extra power gating controller circuitry.

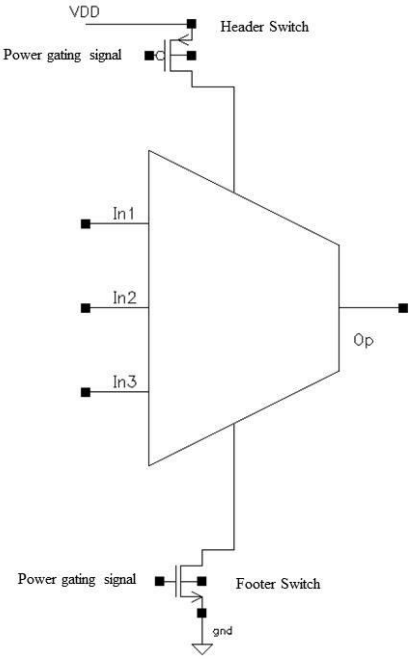


Figure 6-3: Power gating diagram for a 3X1 multiplexer showing header switch and footer switch.

Our internal power gating technique is fairly simple and involves using power gating transistors (as shown in the figure above) which will act as switches (PMOS as a header switch and NMOS as a footer switch) to turn on and off the component. Size of transistors used as a switch (Power gate size) is an important parameter and as a rule of thumb, this is three times the normal transistor [73]. In our experiments, power gating techniques and the power gate size parameter have evaluated for 3X1 and 5X1 multiplexers. In particular, we have measured the cost of this technique on the propagation delay of a component. Data have been presented in **Figure 6-4** and **Table 6-2**.

Table 6-2: Propagation delay of multiplexers in picoseconds for powergated transistor of finger widths 1, 3, 6 and 9.

FW of Power Gate Transistor	3X1 (in ps)	4X1 (in ps) (estimated value)	5X1 (in ps)
1	168	194	221
3	133	155	176
6	124	144	164
9	121	140	159
No Power Gating	110	130	150

FW = Finger width

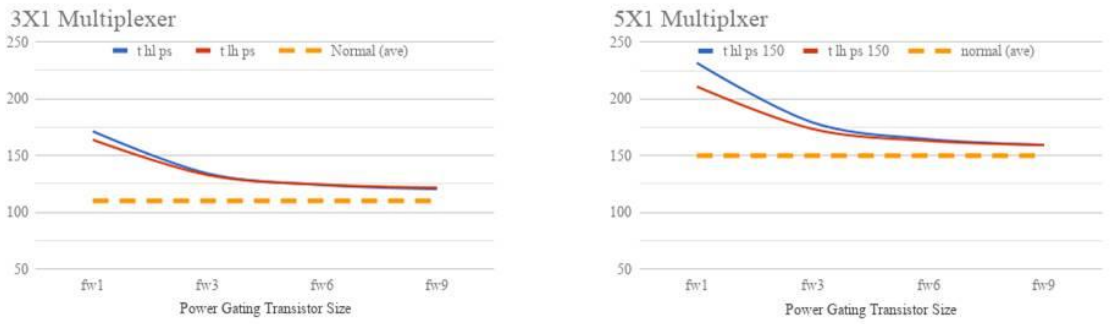


Figure 6-4: Graphs showing propagation delay getting closer to normal with an increase in the size of a powergated transistor for multiplexers.

It can be seen in the measurements that delay increases when power gating is applied, as might be expected. However, for reasonable sized gating transistor delay increase is only around 10% for the multiplexer of interest. Therefore designers have an opportunity here to accept slower multiplexer in exchange for potential reduction in

power consumption. We have seen in the previous chapter that slower components can sometimes be used in non-critical paths without affecting overall SSIA delay.

Power gating causes VDD and GND bouncing [74] and our observation of that phenomenon has also been presented below. Experiments (using 5X1 multiplexer as an example) have also been conducted to investigate the impact when one of the switches (either header or footer) is removed. All of the experimental results are presented in tables and figures below.

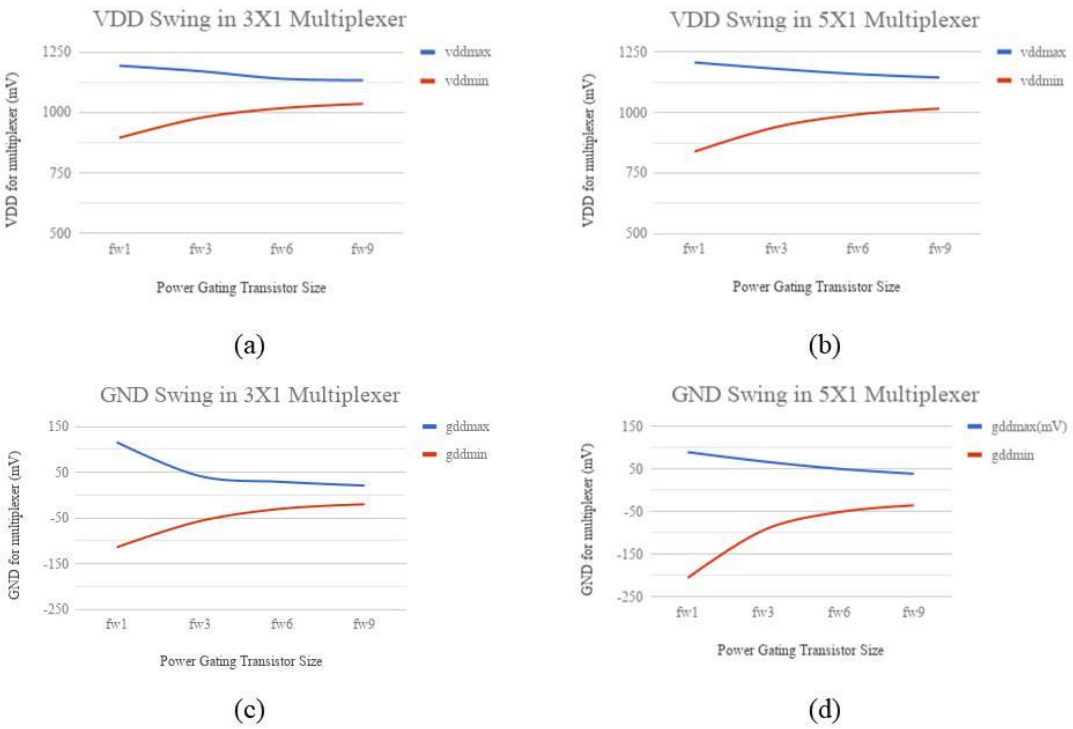


Figure 6-5: Graphs showing GND bounce and VDD bounce getting closer to normal with an increase in the size of a powergated transistor for multiplexers.

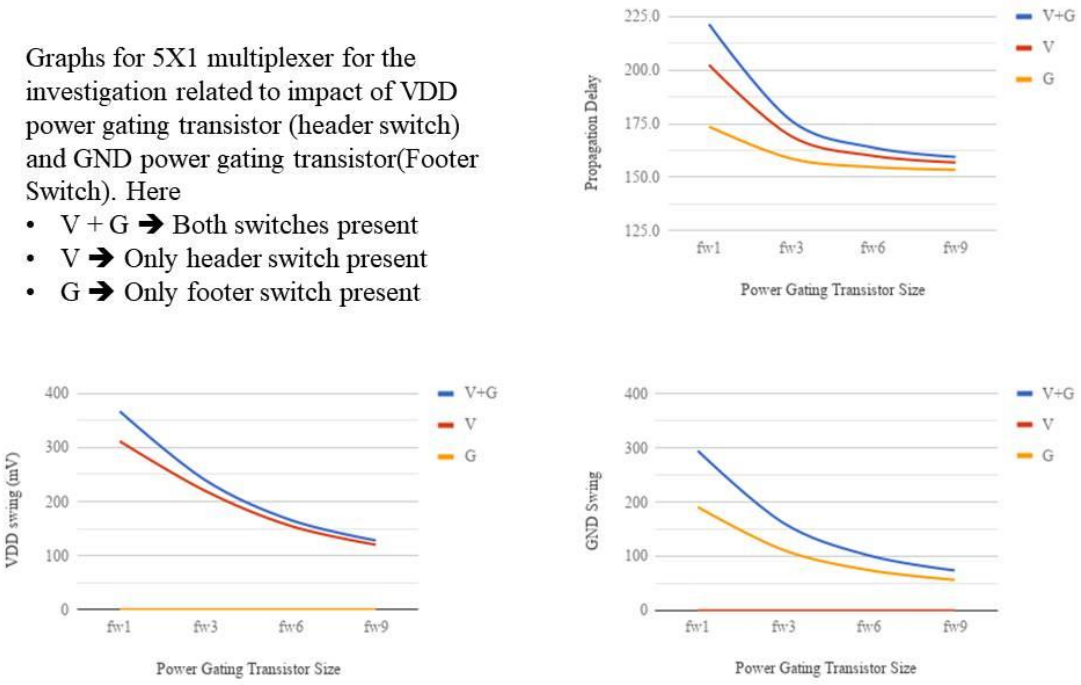


Figure 6-6: Graphs for 5X1 multiplexers for the investigation related to the impact of header switch and footer switch (Quadrant 1)propagation delay, (Quadrant 3)VDD swing and (Quadrant 4)GND swing.

6.3 MOS Current Mode Logic (MCML) Implementation

MOS Current Mode Logic (MCML) Implementation (or simply *current mode implementation*) has been explained briefly in the section 2.2.2 of literature review chapter.

It has been proven by N. Pandey et al. that the propagation delay of tri-state buffer can be reduced by 11% to 60% by using PFSCL methodology [53], K. Gupta et al. are able to reduce the delay of inverter by nearly 20% using MCML [49], and Khule et al. are able to reduce the delay of inverter by nearly 75% using MCML [54]. Based on their

research very conservative speculation of 10% to 20% reduction in delay can be made. With that assumption, new data for SSIA has been calculated and has been presented in **Table 6-3** and **Table 6-4** below. It can be seen that it may be possible to even halve the worst-case delay using current mode implementation.

Table 6-3: Speculated propagation delay data for the MCML multiplexer.

	Original	**10%**	**15%**	**20%**
5X1_F3_FO5	94	85	80	75
4X1_F3_FO5	88	79	75	70
4X1_F2_FO4	86	77	73	69
3X1_F2_FO3	79	71	67	63

Table 6-4: Prediction for SSIA based on the speculation shown in table 6-3.

Parameters	**Current Mode**	**SSIA_C**	**SSIA_A**	**% difference**
Delay (worst-case)	300 ps	376 ps	602ps	50% (↓)

6.4 Summary

The review suggested that there is scope for optimization of SSIA. It is observed here that speed can be improved (or in other words, propagation delay can be reduced) by use of dynamic-logic techniques. It is also observed that power-gating is a useful technique in the SSIA design to save power at cost of reduced speed. Finally, SSIA

improvements that could be attained using current mode have been speculated. As mentioned earlier, three optimization techniques have been explored.

The first section (6.1) about dynamic-logic presented the propagation delay data of dynamic Tri-state buffer - two novel implementations. As explained previously in section 2.2.2 that Dynamic Logic replaces the pull-up network with a single pull-up transistor. Replacing the pull-up network with a pull-up transistor will offer a reduction in area. In general, dynamic implementation is fasters too – those are the advantages of choosing dynamic implementation at the expense of complexity and less robust design. SSIA_A (on page number 114) is static CMOS implementation of SSIA. Comparison of SSIA_A with the dynamic implementation of SSIA will be an insightful work which couldn't be done due to lack of time and is a candidate for future work.

The second section on power gating (6.2) explained how the technique impacts the propagation delay of the multiplexer, and also mentioned about the ground bounce. Power gating is relevant for SSIA because during each instruction only one of the multiplexers of SSIA is in use. Though more research is required to make a case it is feasible to design power gated SSIA.

The third section (6.3) about current mode implementation speculated the improvement using the optimization techniques. Comparison of MCML implementation of SSIA with the dynamic implementation of SSIA will also be an insightful work but this as well couldn't be completed and is also a candidate for future work.

This is the last contribution chapter of the thesis. The next chapter will present consolidated conclusions of all the contribution chapters (CHAPTER 3, CHAPTER 4, CHAPTER 5, and CHAPTER 6) and future direction for the thesis.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Previous thesis chapters presented evidence to answer the main research question. In this chapter, the evidence has been summarized, and their significance has been discussed – in relation to the main research question. A future direction for the research has also been provided in this chapter.

7.1 A Reminder of the Research Question

This section restates the research question stated previously in the sub-section 1.4.4 of CHAPTER 1. In order to answer the very broad research question specific tasks were set and they are also mentioned below in a numbered list.

"Can the performance of superscalar stack structure be significantly improved by the use of advanced design techniques, and will this make them more competitive with register file?"

1. **Develop a baseline performance model**, the *Standard Cell* library, which builds upon existing work, and which can form a point of comparison against any improvements achieved.
2. **Evaluate the custom design approach**, create a new *Custom Cell* library, and evaluate their performance relative to the above point (1).

3. Given standard Cell library, and custom cell library, **compare the SSIA structures** which are possible and evaluate how much improvement can be obtained over the baseline implementation.
4. With suitable optimal or highly optimized designs for SSIA, repeat the comparison of Bailey and Mullane for **SSIA versus register file models**, and determine if competitiveness has improved significantly.
5. Document and test more **advanced VLSI design techniques** that can be used in the future to inform future research directions, and highlight possibilities for continued work in future research.

7.2 Summary of Contributions

This section presents all the contribution made using previous chapters in relation to the research question and its tasks, restated in the previous section. The contributions in this section have been categorised based on the tasks.

7.2.1 Develop a Baseline Performance Model

- Standard Cell library has been created in 65nm CMOS process technology.
- The fundamental components of the Standard Cell library have been quantified in terms of performance parameters such as area, propagation delay, power, and capacitance. The impact of fanout on performance has also been evaluated.
- Multiplexers for Standard Cell library have been designed using the fundamental components and they have also been quantified in terms of performance parameters such as area and delay.

- Multiplexers have been simulated using an alternative methodology (which takes the impact of internal wiring), and the comparison with original methodology has also been provided. The comparison helps gauge the impact of internal wiring.
- First order performance projection of SSIA, which is designed using the Standard Cell library, has also been presented.

7.2.2 Evaluate the custom design approach

- Custom Cell library has been created in 65nm CMOS process technology.
- Two custom design approaches *wider MOSFET implementation* and *multi-finger MOSFET implementation* have been compared.
- An evaluation for performance has been carried out for fundamental components made up of multi-finger MOSFETs (As the above-mentioned comparison found multi-finger MOSFET implementation advantageous).
- Like Standard Cell library, multiplexers have been designed for Custom Cell library and have also been quantified in terms of performance parameters such as area and delay.
- Novel components have been created – employing custom design approach at various design-levels.

7.2.3 Compare the SSIA Structures

- SSIA versions have been compared based on their performance parameters – such as area, delay and power.
- SSIA versions – such as SSIA_A, SSIA_B, SSIA_C, etc. – have been selected manually that:
    - occupies the least area on the chip.

- offers the smallest worst-case delay.
- consumes the least power.

- SSIA Predictor – an excel dashboard, and DARWIN – a toolset have been proposed. These tools permit SSIA design space exploration, evaluation, and can ultimately make a recommendation using an algorithmic approach.
- Recommendations made by SSIA Predictor have been discussed and some are also compared with those, which were chosen manually.

7.2.4 SSIA versus Register File Models

- Result of baseline performance model (SSIA_A) is found to agree with work done by Crispin-Bailey and Mullane in 2014 [1] (**Figure 5-14**).
- Standard cell design (SSIA_A) could be improved significantly with custom cell design (SSIA_H) as noted in **Figure 5-15**.
- Performance of SSIA_H is found to be on par with the register file as noted in **Figure 5-16**.

7.2.5 Advanced VLSI Design Techniques

- A review of advanced VLSI optimization techniques has been presented.
- A VLSI Optimization technique, pre-charging, has been examined for tri-state buffer and its benefits have been presented.
- Another VLSI Optimization technique, power-gating, has been examined for multiplexers and their impact on propagation delay has been investigated.
- Improvements in a performance that can be achieved by the use of current-mode techniques have been predicted.

This section presented a sub-section for each task of the research question, and each sub-section presented a list of the contribution relevant to the task made in previous chapters.

7.3 Discussion of Contributions

The structure of this section is the same as the previous section, i.e. a sub-section for each task of the research question. In this sub-section, we discuss the significance of the contributions listed above section and how they help answer the research question.

7.3.1 Develop a Baseline Performance Model

One of the aims of this research is to advance the preliminary investigation and find out if the performance of SSIA can be improved. In order to answer this baseline performance data is a necessity. The baseline performance data obtained in CHAPTER 3 can be used to measure any enhancement of performance of SSIA that will be achieved using sophisticated design techniques. Additionally, the data included many performance parameters (such as Area, Delay, Power, Capacitance and Fanouts), these are useful in painting a realistic picture of performance. Components (such as inverter, tri-state buffer and multiplexers) of SSIA are used in varying fanout conditions hence the detailed fanout analysis has been performed to find out the trend of their performance. Also, results were found to agree well with Crispin-Bailey and Mullane work as seen in **Figure 5-14** of section 5.5.

7.3.2 Evaluate Custom Design Approach

It has been identified during the literature review that custom cell design technique can be used to improve the SSIA, but these benefits will come at a cost. The measurements have revealed that faster components are possible at the expense of larger

chip area although the trend is of diminishing return. It has also been observed that it is advantageous to use multi-finger MOSFETs to achieve higher drive strengths, but also that there is a tradeoff against the penalty of the effect of 'self-loading', generally, this implies a certain number of fingers offer the best performance gains. Further, it can be observed that, contrary to the naive view, deliberately reducing the speed of some components may offer area benefits, without reducing overall system performance.

7.3.3 Compare the SSIA Structures

As mentioned earlier the development of two libraries resulted in various versions of components and hence the sheer number of possible versions of SSIA and accompanying trade-off. Therefore a tool (SSIA Predictor) can be useful in the exploration and evaluation. Through the tool, it could be discovered that high speed of best level can also be achieved with a little lesser area than the 'naive' solutions, by exploiting the individual members of the component library. The use of an algorithmic approach (in SSIA Predictor) also yielded solutions that balance specific design priorities (combinations of the area, delay, and power) in a way that is difficult to figure out through a manual selection process.

7.3.4 SSIA versus Register File Models

Register files are used widely hence it's important to provide the comparison of our SSIA and register files. Comparison of SSIA and register files have been provided in section 5.5. The first graph (**Figure 5-13**) in the section has been taken from work done previously by Crispin-Bailey and Mullane, it can be seen there that at higher issue width SSIA performance wasn't as good as register files. This limitation has been overcome by SSIA_H (custom design) and has been shown in **Figure 5-16**.

7.3.5 Advanced VLSI Design Techniques

The review signifies that there is more scope for optimization of SSIA. As it has been observed that speed and power can be improved, this suggests an even better version of SSIA is possible and it can be deduced that SSIA better than register files can be designed because our comparison in the previous task found SSIA and register files equivalent in performance.

This section (7.3) presented a sub-section for each task of the research question, and each sub-section presented the significance of the contribution relevant to the tasks.

7.4 Opportunities for Future Work and Refinements

For this thesis, the only a preliminary investigation could be conducted for advanced optimization in the time we had. In future, it would be useful to find out what more could be achieved for SSIA using Dynamic Logic, Domino Logic and Current Mode Implementation. An investigation was limited to TSMC 65nm technology but it's possible to obtain FO4 data and extrapolate the results for advanced technology using other research [75]. It would also be useful to simulate the full SSIA circuit, see SSIA in action, and provide more realistic input data – this will provide new results for SSIA and can even be compared to the first order projection. DARWIN – a toolset for SSIA recommendations – can be made more advanced to, as already mentioned in the relevant chapter.

Another direction that can be taken is towards Java Virtual Machine. JVM is modern and quite successful platform and uses stack architecture. Hennessy and Patterson expressed that their hardware implementation is yet to see commercial success.

A recent attempt in that direction has been made by some [19]. It is a topic that can be a dimension of stack processor research at York.

7.5 Concluding Remarks

From the discussions in previous sections, it is safe to say that there many evidence that advanced VLSI design techniques can improve the performance of SSIA, and IT IS advantageous to use custom design techniques for SSIA – which answers the first part of the research question. Further, test benches and their simulation are done keeping realistic conditions in mind and worst-case scenario has been assumed (e.g. in the delay calculation and temperature of the simulation). In the thesis, a baseline model performance has been presented that can be used to measure the enhancement of SSIA, custom design approach (such as wider MOSFET v/s multi-finger MOSFET) has been investigated, a tool that can assist in the component selection of SSIA has been proposed, SSIA designs have been compared with register files and it has been found out that SSIA can be as competent as register files – this answers the second part of the research question. Preliminary investigations of advanced VLSI techniques have also been presented.

APPENDIX A
DATASET

Table A-1: Propagation delay data of old and new implementation of SSIA.

Components (↓)	**90nm UMC**	**65nm TSMC**		**Reduction (in percentage)**
	Standard Cell		**Custom Cell**	
Tristate	89.0 ps	70.2 ps	57.7 ps	35.2 %
2X1	98.0 ps	83.2 ps	65.2 ps	33.5 %
3X1	107.0 ps	95.5 ps	72.1 ps	32.6 %
4X1	116.0 ps	107.3 ps	79.0 ps	31.9 %
5X1	125.0 ps	118.9 ps	85.6 ps	31.5 %
6X1	134.0 ps	130.3 ps	92.1 ps	31.3 %
7X1	143.0 ps	141.3 ps	98.5 ps	31.1 %
8X1	152.0 ps	151.2 ps	104.7 ps	31.1 %

ABBREVIATIONS

ACAG	Advanced Computer Architecture Group
ACE	Automatic Computing Engine
ADE	Analog Design Environment
ALU	Arithmetic Logic Unit
AMADEUS	Architectures Machines And Devices for Efficient Ubiquitous Systems
AMD	Advanced Micro Devices (Company name
ASIC	Application Specific Integrated Circuit
CAD	Computer-Aided Design
C_{dn}	NMOS Drain/Source Capacitance
C_{dp}	PMOS Drain/Source Capacitance
C_{gn}	NMOS Gate Capacitance
C_{gp}	PMOS Gate Capacitance
CISC	Complex Instruction Set Computer
C_{load}	Load Capacitance (Equivalent Capacitance at Node)
CPU	Central Processing Unit
D	Depth of a stack
DARWIN	Digital Architecture Refinement With INheritance
DEC	Digital Equipment Corporation

DLX	RISC processor architecture pronounced 'deluxe'
DRC	Design Rule Check
DTI	Department of Trade and Industry
equalNP	The model name for an inverter with same size NMOS and PMOS
FOX	Fanout of X (e.g FO4; X ranges from 1 to 8 in this thesis)
FPGA	Field Programmable Gate Array
FW	Finger Width
GND	Ground
GPU	Graphics Processing Unit
HP	Hewlett Packard
I/P = H and I/P = L	Input value high(1) and low (0)
IC	Integrated Circuits
IE	Internet Explorer
ILP	Instruction Level Parallelism
IPC	Instruction Per Clock-cycle
ISA	Instruction Set Architecture
Iw_x	Issue Width (X can be from 1 to 4)
JVM	Java Virtual Machine
L → H	Low to High transition
LVS	Layout Versus Schematic
MCML	MOS Current-Mode Logic
MISC	Minimal Instruction Set Computer

MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MUX	Multiplexer
NMOS	n-channel MOSFET
NOC	No Connection (and hence zero fanout)
NOMAD	Non-standard Operand Mechanism Architecture Demonstrator
NORMA	Normal Order Reduction Machine
NX1	N-input MUX (e.g 3X1)
PFSCL	Positive Feedback Source-Coupled Logic
PMOS	P-channel MOSFET
RC	Resistor-Capacitor
RF	Register Files
RISC	Reduced Instruction Set Computer
RTL	Register-Transfer Level
SAFA	Stack and Frame Architecture
SCL	Source-Coupled Logic
S_j	the j^{th} element of the stack
SRAM	Static Random-Access-Memory (RAM)
SSIA	Superscalar Stack Issue Array
SSIA_X	Model X of SSIA (‘X’ ranges from A to H in the thesis)
TNN	Tristate multiplexer model with Nonencoded select inputs, and Noninverting output per stage
TSMC	Taiwan Semiconductor Manufacturing Company
UFO	Ubiquitous Forth Objects

UMC	United Microelectronics Corporation
UUT	Unit Under Test
VDD	Positive supply voltage
VLSI	Very-Large-Scale Integration

BIBLIOGRAPHY

[1] C. Bailey and B. Mullane, "Investigation of a Superscalar Operand Stack Using FO4 and ASIC Wire Delay Metric," VLSI Design, vol. 2014, p. 13, 2014.

[2] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, Fourth ed., San Fransisco: Morgan Kauffman, 2007, pp. *K9-K10*.

[3] H. Sutter, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software," [Online]. Available: *http://www*.gotw.ca/publications/concurrency-ddj.htm.

[4] J. Handy, "The End of Semiconductor Scaling," 13 November 2011. [Online]. Available: *https://www.forbes*.com/sites/jimhandy/2011/11/13/the-end-of-semiconductor-scaling/. [Accessed 2018].

[5] C. Bailey and M. Weeks, "An experimental Investigation of Single and Multiple Issue," in EuroForth, Cheshire, 2000.

[6] H. Shi, Investigating opportunities for instruction-level parallelism for stack machine code, University of York, 2006.

[7] M. Shannon, A C Compiler for Stack Machines, University of York, 2006.

[8] A. Arnone, "Feasibility of Accelerator Generation to Alleviate Dark Silicon in a Novel Architecture," University of York, 2017.

[9] G. Moore, "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.," *IEEE Solid-State Circuits Newsletter*, 2006.

[10] B. Chacos, "Breaking Moore's Law: How chipmakers *are* pushing PCs to blistering new levels," PCWorld, April 2013. [Online]. Available: https://www.pcworld.com/article/2033671/breaking-moores-law-how-chipmakers-are-pushing-pcs-to-blistering-new-levels.html.

[11] Applied Materials, "3D Chip Technology for Dummies - *Video*," appliedschannel, YouTube, 2012.

[12] J. Knechtel, O. Sinanoglu, I. M. *Elfadel*, J. Lienig and C. C. N. Sze, "Large-Scale 3D Chips: Challenges and Solutions for Design Automation, Testing, and Trustworthy Integration," *IPSJ Transactions on System LSI Design Methodology,* vol. 10, pp. 45-62, 2017.

[13] G. A. Subbarao, "Low-power Microprocessor based on *Stack-Architecture*," 2015.

[14] C. Crispin-Bailey, "NOMAD Funding Success," [Online]. Available: https://www.cs.york.ac.uk/arch/news/nomad-funding-success.html.

[15] J. L. Henessey and D. A. Patterson, Computer Architecture, Elsevier/Morgan Kauffman, 2007.

[16] E. Valderrama and J.-P. Deschamps, "Digital *Systems*: From Logic Gates to Processors," Coursera, [Online]. Available: https://www.coursera.org/learn/digital-systems.

[17] P. Koopman, Stack Computers, E. Horwood, 1989.

[18] GreenArrays, Inc., "Homepage," [Online]. Available*:* http://www.greenarraychips.com/index.html.

[19] M. Schoeberl, "Design and Implementation of an Efficient Stack Machine," in *12th IEEE Reconfigurable Architecture Workshop*, Denver, Colorado, 2005.

[20] C. E. LaForest, "Second Generation Stack Computer,*"* 2008.

[21] J. Łukasiewicz, Elements of Mathematical Logic, New York*:* Macmillan, 1963.

[22] C. L. Hamblin, "An Addressable Coding Scheme *Based* on Mathematical Notation," in Proceeding of the First Australian Conference on Computing *and Data Processing*, 1957.

[23] F. L. Bauer, "The Formula-Controlled Logical Computer *"*Stanislaus"," Mathematics of Computation*, vol. 14, 1960.*

[24] K. Samelson and F. Bauer, "Sequential Formula *Translation*," Comm. of the ACM, vol. *3*, pp. 76-83, 1960.

[25] D. E. Newton, Alan Turing: A Study *in* Light and Shadow, USA: Xilbris Corporation, 2003.

[26] G. Gray, "Unisys History Newsletter - Burroughs Third-Generation Computers," October 1999. [Online*]*. *Available:* https://archive.li/Eepw. [Accessed 2018].

[27] I. Martin, "Too far ahead of its time*:* Barclays, burroughs, *and real-time banking," IEEE* Annals of the History of Computing, 2012.

[28] B. Findlay, "The English Electric KDF9," [Online*]*. Available: http://www.findlayw.plus.com/KDF9/The%20English%20Electric%20KDF9.pdf. [Accessed 2018].

[29] J. C. Dvorak, "IBM and the Seven *Dwarfs* — Dwarf One: Burroughs,*"* [Online]. Available: http://www.dvorak.org/blog/ibm-and-the-seven-dwarfs-dwarf-one-burroughs/.

[30] T. Napier and E. Krieg, "Lost at *C*? Forth May *Be the Answer," [Online*].

[31] J. Fox, "Moore Forth - Chuck Moore's Comments on Forth," [Online]. Available: http://www.ultratechnology.com/moore4th.htm.

[32] C.-h. Ting and C. H. Moore, *"*MuP21 - A *High Performance MISC Processor,"* [Online]. Available: http://www.ultratechnology.com/mup21.html.

[33] Micro Live, "Computer architectures 1986 (Transputers)".

[34] M. Scheevel, "NORMA: a graph reduction processor,*"* in ACM Conference *on LISP and Functional Programmin*, Cambridge, Massachussetts, USA, 1986.

[35] B. Paysan, "Implementation of the 4stack Processor Using Verilog," Technical University of Munich, 1996.

[36] S. Y. Jien, "SAFA: Stack and frame architecture," 2006.

[37] C. Bailey, "Optimization Techniques for Stack Based Architecture," University of Teeside, 1996.

[38] C. Bailey and R. Sotudeh, "HLL Enahancement *for* Stack Based Processor," Microprocessing and MIcroprogramming, vol. 40, pp. 685-688, 1994.

[39] C. Bailey and R. Sotudeh, "Quantitative Assessment *of* Machine-Stack Behaviour for Better Computer *Performance"*.

[40] C. Crispin-Bailey, "AMADEUS," [Online]. Available: https://www.*cs.york.ac.uk*/amadeus.

[41] C. Crispin-Bailey, H. Shi and M. Shannon, "Towards Scalable Parallelism *with Stack Machines," Paris*, 2012.

[42] P. Corsonello, S. Perri and V. Kantabutra, "Design of 3:1 multiplexer standard cell," *Electronics Letters*, vol. 36, no. 24, pp. 1994 - 1995, 2000.

[43] IIT Guwahati, "Optimization techniques for Digital VLSI Design," National Programme on Technology Enhanced Learning, 2019.

[44] W.-K. Chen, VLSI handbook, CRC Press, *1999.*

[45] H. Eriksson, P. Larsson-Edefors, T. Henriksson and C. Svensson, "Full-custom vs. standard-cell design flow: an *adder* case study," in Asia and South Pacific Design *Automation Conference, 2003.*

[46] P. Gautam, D. Kaushik and R. P. G. P. Sharma, "Design of CMOS Inverter Using Different *Aspect Ratios," International Journal of Scientific Research Engineering &Technology (IJSRET),* no. EATHD-2015 Special Issue, pp. 132-137, 2015.

[47] L. E. Han, V. B. Perez, *M.* L. Cayanes and M. G. Salaber, "CMOS Transistor Layout Kung *Fu,"* 2005.

[48] A. P. Martınez, "Design of MOS Current-Mode *Logic* Standard Cells," Microelectronics Systems Laboratory, EPFL, Lausanne, 2007.

[49] K. Gupta, R. Sridhar, J. Chaudhary, *N.* Pandey and M. Gupta, "Performance comparison of MCML and PFSCL gates in 0.18 *μm CMOS technology," in International Conference on Computer and Communication Technology - ICCCT*, Allahabad, 2011.

[50] K. Gupta, N. Pandey and M. Gupta, "Performance improvement of PFSCL gates through capacitive coupling," in IMPACT-2013, Aligarh, *2013.*

[51] M. Sumathi and Y. C. Kartheek, "Performance and analysis of CML Logic gates and latches," in 2007 International *Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, Hangzhou, 2007.

[52] K. Gupta, R. Sridhar, J. Chaudhary, *N.* Pandey and M. Gupta, "New Low-Power Tristate Circuits in Positive Feedback Source-Coupled Logic," *Journal of Electrical and Computer Engineering*, vol. 2011, 2011.

[53] N. Pandey, B. Choudhary, K. Gupta and A. Mittal, "New Sleep-Based PFSCL Tri-State Inverter/Buffer Topologies," *Journal of Circuits, Systems and Computers,* vol. 26, no. 12, 2007.

[54] R. Khule, M. Kajale, S. Kshatriya and S. Sardar, "ANALYSIS OF MCML INVERTER," International Journal of Advance Research and Innovative *Ideas in*

Education, vol. 3, no. 3, pp. 1808 - 1816, 2017.

[55] K. Gupta, N. Pandey and M. Gupta, "Multithreshold MOS Current Mode Logic Based Asynchronous Pipeline Circuits," ISRN Electronics, vol. 2012, 2012.

[56] S. Badel and Y. Leblebici, "Tri-state buffer/*bus* driver circuits in MOS current-mode logic," in 2007 Ph.D Research in *Microelectronics and* Electronics *Conference (PRIME), Bordeaux, France, 2007.*

[57] R. Singh, G.-M. Hong, M. Kim, J. Park, W.-Y. Shin and S. Kim, "Static-switching pulse domino*: A switching-aware design technique for wide fan-in dynamic multiplexers,*" Special Issue of GLSVLSI 2011: Current Trends on *VLSI and Ultra Low-Power Design,* vol. 45, no. 3, pp. 253-262, 2012.

[58] F. Maloberti, Analog Design for CMOS VLSI Systems, Boston, MA: Springer US, 2003.

[59] M. Tohidi, J. K. Madsen, M. J. R. Heck and F. Moradi, "Low-power comparator in 65-nm CMOS with *reduced delay time," in 2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS),* Monaco, 2016.

[60] P. Meher and K. K. Mahapatra, "A Low-Power Circuit Technique for Domino CMOS Logic," in 2013 International Conference on *Intelligent Systems and* Signal Processing (ISSP), 2013.

[61] A. K. Pandey, R. A. Mishra and R. K. Nagaria, "Leakage Power Analysis of Domino XOR *Gate,"* ISRN Electronics, vol. 2013, p. 7 page, *2013.*

[62] R. Thakur, A. K. Dadoria and T. K. Gupta, "Comparative analysis of various Domino logic circuits for better performance," in *2014 International Conference on Advances in Electronics, Computers and Communications (ICAECC),* 2014.

[63] Cadence, "Spectre Circuit Simulator User Guide," Cadence, 2018.

[64] M. Shams, "Advanced Digital Electronics (ELEC-4708)," 2014. [Online]. Available: http://www.doe.carleton.ca/~shams/ELEC4708/Lab1SchematicTut2014.pdf.

[65] H. E. Graeb, Analog layout synthesis, New York: Springer, 2011.

[66] J. M. Rabaey, A. Chandrakasan and B. Nikolic, Digital Integrated Circuit, 2003.

[67] N. Kavvadias, "mprfgen manual," [Online]. Available: http://www.nkavvadias.com/doc/mprfgen/mprfgen-README.html.

[68] S. Skiena, The Algorithm Design Manual, 2nd ed., Springer Science+Business Media, 2010, p. 253.

[69] M. Horowitz, D. Harris, R. Ho and G.-Y. Wei, "The Fanout-of-4 Inverter Delay Metric," [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.831&rep=rep1&type=pdf. [Accessed 2018].

[70] D. Harris, R. Ho, G.-Y. Wei and M. Horowitz, "The Fanout-of-4 Inverter Delay Metric," Unpublished.

[71] D. Harris and I. Sutherland, "Logical effort of carry propagate adders," in 37 Asilomar Conference *on Signals, Systems & Computer*, Pacific Grove, CA, USA, 2003.

[72] A. S. Sedra and K. C. Smith, Microelectronic Circuits, Oxford University Press, Inc..

[73] Murali, "ASIC-System on Chip-VLSI Design," 2008. [Online]. Available: http://asic-soc.blogspot.com/2008/04/power-gating.html.

[74] UltraCAD Design, Inc, "Ground Bounce," Printed Circuit Design, a Miller Freeman publication, August and September 1997.

[75] R. Kumar and V. Kursun, "Impact of temperature fluctuations on circuit characteristics in 180nm and 65nm CMOS technologies," in 2006 *IEEE International Symposium on Circuits and Systems, Island* of Kos, Greece, 2006.

[76] R. Shioya, K. Horio, M. Goshima and S. Sakai, "Register Cache System Not for Latency Reduction Purpose," in 43rd IEEE/ACM *International Symposium on Microarchitecture (MICRO)*, 2010.

[77] H. Shen-Fu and W. Pu-Cheng, "Design of low-leakage multi-port SRAM for," in IEEE International Symposium on *Circuits and Systems (ISCAS)*, 2014.

[78] T. Shakir, "Low-Power, Low-Voltage SRAM Circuit Designs For Nanometric CMOS Technologies," University of Waterloo, 2011.

[79] J. Shah, "Low-Power Soft-Error-Robust Embedded SRAM," University of Waterloo, 2012.

[80] J. M. Shah, "Modeling and Mitigation of Soft Errors in Nanoscale SRAMs," University of *Waterloo*, 2008.

[81] A. Neale, "Digital Timing Control in SRAMs for Yield Enhancement and Graceful Aging Degradation," University of Waterloo, 2010.

[82] B. Merchant, "IBM Patented a One Atom-Thick Graphene Transistor That Works 1,000 Times Faster Than *Silicon - Motherboard,"* February 2013. [Online]. Available: https://motherboard.vice.com/en_us*/article/kbb9wx/a-single-atom-thick-graphene-transistors-transmits-electricity-1000-times-faster-than-silicon-chips.*

[83] Y. Li, L. Zhang, Q. Zhang, Z. Wang and L. Mao, "A Low Power Area Efficient Full Custom 3-Read 3-Write General Purpose Register in 65nm Technology," in *International Conference on Computer Sciences and Applications (CSA)*, 2013.

[84] K. Kwong, "Design of a Robust, Low-Leakage Register File for Sub-130nm Technologies," University of Waterloo, 2004.

[85] T. Karim, "On-Chip Power Supply Noise: Scaling, Suppression and Detection," UWSpace, 2012.

[86] S. Galal and M. Horowitz, "Energy-Efficient Floating-Point Unit Design," IEEE Transaction on Computers, *vol. 60, pp.* 913-922, 2011.

[87] Y. L. P. A. D. Farmer, "Graphene field-effect transistors with self-aligned gates," *Applied Physics Letter,* 2010.

[88] Chuang, "High-Performance, Energy-Efficient CMOS Arithmetic Circuits," University of Waterloo, 2014.

[89] A. Alvandpou, R. Krishnamurthy, K. Soumyanath and S. Borkar, "A lowleakage dynamic multi-ported register file in 0.13um CMOS Technology," in *International Symposium on Low Power Electronics and Design*, 2001.

[90] Stanford University, "21st Century Computer Architecture," 2012.

[91] O. Z. YONG, "LOW-POWER RF DESIGN: SELECTIVE POWER-GATED DOMINO MULTIPLEXER," Universiti Tunku Abdul Rahman, perak, Malaysia, 2014.