DESIGN OF HIGH THROUGHPUT DIGITAL CIRCUITS USING NOVEL ASYNCHRONOUS PIPELINE METHODS

Thesis

Submitted in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by
K SRAVANI

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING,
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,
SURATHKAL, MANGALORE -575025

November, 2020

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL

DECLARATION

by the Ph.D. Research Scholar

I hereby *declare* that the Research Thesis entitled ***Design of High Throughput Digital Circuits using Novel Asynchronous Pipeline Methods*** Which is being submitted to the National Institute of Technology Karnataka, Surathkal in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy in **Electronics and Communication Engineering** is a *bonafide report of the research work carried out by me.* The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

155121EC15F03, K.SRAVANI (K.Sravani)

(Register Number, Name & Signature of the Research Scholar)

Department of Electronics and communication engineering

Place: NITK-Surathkal

Date: 19/11/2020

Note: Declaration to be signed by the Scholar and incorporated as part of the Ph.D. Research Thesis /Synopsis

CERTIFICATE

This is to certify that the Research Thesis entitled **Design of High Throughput Digital Circuits Using Novel Asynchronous Pipeline Methods** submitted by **K.SRAVANI** (Register Number:EC15F03) as the record of the research work carried out by her, is accepted as the *Research Thesis submission* in partial fulfillment of the requirements for the award of degree of **Doctor of Philosophy**.

Dr. Rathnamala Rao
Research Guide,
Assistant Professor,
Dept. of Electronics and Communication Engg.,
NITK Surathkal-575025.

Chairman-DRPC
(Signature with Date and Seal)

Acknowledgements

As I near the end of my Ph.D. journey, I would like to take a moment and acknowledge people who have selflessly helped me along the way. I am extremely grateful to my Ph.D. supervisor, Dr.Rathnamala Rao. This work would not have been possible without her guidance, support, and encouragement. I am grateful for the great advises, both technical and personal, that she gave me over these years. I have been amazingly fortunate to have her as my advisor, who showed much more faith in me than I deserved. I am particularly thankful for the freedom she gave me to explore on my own and at the same time, the guidance to recover when my steps faltered. She has always been generous throughout the entire period of research.

I am indebted to Prof. M.S. Bhat, for all his invaluable advice, constant moral support, and motivation. Along with him, I would like to thank my other Research Progress Assessment Committee member Dr.Geetha.V, Assistant professor in the Department of Information Technology, for her valuable suggestions to improve the content and quality of my research work. I want to thank Prof. Laxminidhi Tonse, the present Head, Department of Electronics and Communication Engineering, for his precious guidance and administrative support. I pay my sincere thanks to all the other faculty and staff members of the E & C department.

I thank to all my fellow research scholars at NITK for their continuous support. Special thanks to Karuna Kumari for her understanding, and the enormous help. I greatly value her wonderful company and deeply appreciate her constant support. I expand my special thanks to Shara Mathew for her suggestions and help during these years.

I am grateful to Ministry of Human Resource and Development, Government of India for financially supporting me to carry out this research work. I owe a deep debt of gratitude to our institute for giving the opportunity to complete this work.

Most importantly, none of this would have been possible without the unconditional love and prayers of my in-laws and parents . I pay high regards to my parents Lakshmi Devi and Thirumala Reddy for their sincere encouragement and inspiration throughout my research work and lifting me uphill this phase of life. I owe everything to them. My joy knows no bounds in expressing my cordial gratitude to my beloved husband, Dr.Vasudeva Reddy. His technical and moral support pushed me farther than I thought I could go. My Kids, Darahas, Tanush, and Tanvi, have been a great source of joy. Without their help and love, this work would not be completed. I am blessed to have them.

Finally, I would like to thank God for blessing me with good health, strength, and peace during my research work.

Dedicated to

Amma, Naanna, Vasudeva

&

Son of Dasaratha (Lord Sri Rama).

Abstract

Pipelining is a key technique that has enhanced the concurrency and throughput of all modern digital systems. Pipelining methods are broadly classified as synchronous and asynchronous based on the nature of synchronization present between pipeline stages. In the synchronous design style, a global clock signal provides synchronization among stages, and this design style has been predominating the digital world for several decades. However, the designers are switching their interest from synchronous to asynchronous design due to the problems associated with the clock distribution at lower technology nodes (ex: managing clock skew, wasteful clock power). As there is no global clock in the asynchronous design, it provides freedom from clock-related issues. In addition to this, the asynchronous design also has interesting properties like low power consumption, high performance, reduced electromagnetic emission, modularity, and the capacity to process variable data rate signals.

This research work introduces two novel high throughput asynchronous pipeline methods, suitable for gate-level pipelined systems. The proposed methods, named as Early Acknowledged Hybrid (EA-Hybrid) and high capacity hybrid pipeline with post detection (PD-Hybrid), use hybrid data path, that can combine the robustness of dual-rail encoding and simplicity of single-rail encoding schemes. The domino logic style has been adopted for constructing the logic gates in each pipeline stage, as it can provide the latch-less feature. The control path of EA-Hybrid is built based on high-speed early acknowledgment protocol, whereas in PD-Hybrid it is built based on simple and robust 4-phase protocol. Further, both the proposed pipeline styles allow their logic gates into a special state called isolate phase in addition to precharge and evaluation phases. The isolate phase leads to improvement in pipeline throughput as well as storage capacity.

Different digital circuits like FIFO, Ripple carry adder, array multiplier, and FIR filter are designed based on proposed pipeline styles and simulated using cadence tool suite.

Keywords: Asynchronous pipeline; Throughput; Hybrid logic; Handshaking; Domino logic; FIR filter.

Contents

Acknowledgements i

Abstract iv

List of figures ix

List of tables xi

Nomenclature xii

Abbreviations xii

1 INTRODUCTION **1**

1.1 Motivation for the present work 5

1.2 Research Objectives 5

1.3 Organization of the Thesis 5

2 LITERATURE REVIEW **7**

2.1 Introduction 7

2.2 Synchronous Pipelines 7

2.3 Asynchronous Pipelines 8

2.4 Background of Asynchronous Pipelines 9

2.4.1 Handshake Channels 9

2.4.1.1 Communication Protocols 10

2.4.1.2 Data Encoding Schemes 11

2.4.2 Performance Parameters of Asynchronous Pipelines 13

2.5 Classification of Asynchronous Pipelines 14

2.5.1 Static logic pipelines 14

2.5.2 Dynamic logic pipelines 16

2.5.2.1 Domino logic style 17

2.5.2.2 Asynchronous dynamic logic pipeline methods 19

2.6 PRML read channel 25

2.7 Summary 26

3 HIGH CAPACITY HYBRID PIPELINES 29

3.1 Introduction 29

3.2 Proposed Pipeline Styles 29

3.2.1 Data Path 32

3.2.1.1 Logic block 32

3.2.1.2 Stable critical data path construction 34

3.2.1.3 Robustness of critical path 35

3.2.1.4 Encoding converter 35

3.2.2 Control Path 36

3.2.2.1 Completion Detector 37

3.2.2.2 Stage Controller 38

3.2.3 Protocol of Early Acknowledged Hybrid Logic Pipeline 40

3.2.3.1 Timing Constraints 41

3.2.4 Protocol of Hybrid Logic Pipeline With Post Detection 42

3.2.5 Initialization 43

3.3 Simulation Results 44

3.3.1 FIFO 44

3.4 Summary 48

4 DESIGN OF HIGH SPEED COMPUTATIONAL UNITS 49

4.1 Introduction 49

4.2 Ripple Carry Adder 49

4.3 Array Multiplier 57

4.4 Summary 62

5 HIGH THROUGHPUT DIGITAL FIR FILTER FOR PRML READ CHANNEL APPLICATION 63

5.1 INTRODUCTION 63

5.2 Proposed Asynchronous Digital FIR filter 64

5.2.1 Specifications of FIR filter 64

5.2.2 Filter architecture 65

5.2.3 Simulation Results 69

5.3 Conclusion 71

6 CONCLUSIONS AND FUTURE WORK **73**

6.1 Summary and Conclusions 73

6.2 Scope of Future Work 74

Bibliography **77**

References 81

Publications based on the thesis **82**

List of Figures

1.1 Broadcast structure of 3-tap FIR filter (Parhi (2007)) 1
1.2 Fine-grain pipelined FIR filter (Parhi (2007)). 2

2.1 Synchronous pipeline(Mannakkara and Yoneda (2010)) 8
2.2 Asynchronous pipeline(Mannakkara and Yoneda (2010)) 9
2.3 Asynchronous communication protocols (Mannakkara and Yoneda (2010)). 10
2.4 SR encoding (Nowick and Singh (2015)) 12
2.5 (a) DR encoding (b) Completion detector (Nowick and Singh (2015)). . 12
2.6 Sutherland's micropipeline (Sutherland (1989)) 14
2.7 Mousetrap pipeline (Singh and Nowick (2007*c*)) 15
2.8 Domino logic AND gate (Rabaey *et al.* (2002)) 17
2.9 Dual-rail domino logic AND/NAND gate (Rabaey *et al.* (2002)) 18
2.10 DDCSV logic AND/NAND gate (Anis *et al.* (2002)) 18
2.11 Block diagram of PS0 (Williams (1990)) 19
2.12 Structure of PCHB pipeline (Lines (1998)) 20
2.13 Structures of LP3/1 and LP2/2 pipelines (Singh and Nowick (2007*b*)). . 22
2.14 Structure of HC pipeline (Singh and Nowick (2007*a*)) 22
2.15 Structure of SP pipeline (Midhun *et al.* (2014)) 23
2.16 Structure of Asynchronous pipeline based on constructed critical data paths (Xia *et al.* (2012*b*)) 23
2.17 Block diagram of disk drive read channelKi *et al.* (2000). 25
2.18 (a) Fork (b) Join 26

3.1 Structure of Early acknowledged hybrid logic pipeline (EA-Hybrid). . . 30
3.2 Structure of High capacity hybrid logic pipeline with post detection (PD-Hybrid). 30
3.3 Sequence of phases in a stage-N. 31

3.4 Domino logic AND gate. (a) Single rail gate. (b) DR-SLG. 32
3.5 An example data path of the proposed pipeline styles. 34
3.6 SLGL AND gate. 34
3.7 (a) Encoding converter. (b) Function table. 36
3.8 Completion Detector of EA-Hybrid logic pipeline style. 38
3.9 Completion Detector of PD-Hybrid logic pipeline style. 38
3.10 Stage Controller. 39
3.11 Block diagram for a single stage. 40
3.12 Sample input and output waveforms for stage-2 in FIFO. 45
3.13 Layout of 4-bit,10-stage FIFO. 46

4.1 4-bit ripple carry adder(Rabaey *et al.* (2002)) 50
4.2 Data path of fine grain pipelined ripple carry adder(Xia *et al.* (2015)) . 50
4.3 (a) FA-sum SR gate (b) FA-carry-SLG. 51
4.4 Screen-shot of 16-bit EA-Hybrid adder output. 52
4.5 Screen-shot of 16-bit PD-Hybrid adder output. 52
4.6 (a) Definition of workload Xia *et al.* (2015). (b) Power consumption at different workloads. 54
4.7 Throughput vs. operand size. 55
4.8 Layout of 4-bit adder. 56
4.9 (a) FA Carry circuit layout. (b) Completion detector layout. 56
4.10 (a) Data path of 3x3 array multiplier 57
4.11 Screen-shot of 8X8 EA-Hybrid array multiplier output. 58
4.12 Screen-shot of 8X8 PD-Hybrid array multiplier output. 58
4.13 Throughput VS Supply voltage. 60
4.14 Testing of array multiplier at different Work loads 61

5.1 Filter output waveforms and Error signal 65
5.2 Structure of FIR filter data path. 66
5.3 Join Structure for EA-Hybrid pipeline. 66
5.4 (a) completion detector of adder 1st stage (b) SLG of adder 1st stage. . 67
5.5 Filter input, output wave forms 69
5.6 Comparison of filter output waveforms. 70

List of Tables

2.1 Summary of asynchronous gate-level pipeline styles 24

3.1 SLG AND gate pull down paths state 33
3.2 Function table of the Stage Controller 39
3.3 Simulation results of FIFO in 180 nm technology. 46
3.4 Throughput of different pipeline designs for FIFO test case. 47

4.1 **Evaluation results of 16-bit adder (90 nm).** 53
4.2 Delay of individual components in 16-bit, EA-Hybrid adder 54
4.3 Delay of signals at nodes x and $\overline{x}$ 57
4.4 Evaluation results of 8x8 array multiplier (180 nm) 59
4.5 Simulation results of 8x8 array multiplier at lower technologies 60
4.6 Delay of individual components in 8x8, EA-Hybrid multiplier (65 nm) . 61

5.1 General data format of the FIR filterKi *et al.* (2000) 64
5.2 Simulation results of FIR filter 70
5.3 Comparison of HC and EA-Hybrid FIR filters (in 180 nm technology) . 71

Abbreviations

Abbreviation	Expansion
APCDP	Asynchronous pipeline based on constructed critical path
CD	Completion Detector
DR	Dual Rail
EA	Early Acknowledge
ED^2	Energy delay square
HC	High Capacity
LB	Logic Block
PD	Post Detection
SC	Stage Controller
SLG	Synchronizing logic gate
SLGL	Synchronizing logic gate with latch function
SR	Single rail

Chapter 1

INTRODUCTION

The digital circuits accept and process binary data (on/off) according to the rules of boolean algebra. Compared to analog circuits, digital circuits are less susceptible to noise. Further, it is easier to perform error detection and correction with digital signals. One of the driving forces behind the tremendous improvement in functionality, performance, and power in digital circuits is technology scaling. However, the leakage currents increase as a result of scaling (De and Borkar (1999)). A significant thrust in digital circuit design is to minimize power dissipation while still maintaining high speed. Pipelining is a fundamental method to improve the speed of digital circuits. It divides a complex circuit into small pipeline stages and allows numerous operations to happen in parallel. However, pipelining increases the number of latches and circuit latency. If the circuits are pipelined such that the amount of logic per stage is minimum, such kind of partitioning is called fine-grain pipelining. Fine-grain pipelining allows a circuit to operate at maximum speed. This can be understood by considering an example. Consider a data broadcast structure of the 3-tap FIR filter shown in Figure 1.1.

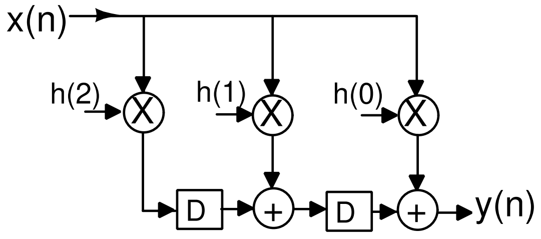


Figure 1.1: Broadcast structure of 3-tap FIR filter (Parhi (2007))

Here, the critical path is limited by the delay of one multiplier (T_M) and one adder (T_A) units. Thus, the minimum 'clock period', T_{clk} (or the maximum clock frequency f_{clk}) for this structure can be calculated as in equation (1.1).

$$T_{clk} = (T_M + T_A) \quad or \quad f_{clk} = 1/(T_M + T_A) \tag{1.1}$$

If $T_M = 10$ units and $T_A = 2$ units, then the delay of the critical path for the structure in Figure 1.1 is 12 units. Suppose if the multiplier unit in this structure is fine-grain pipelined, as shown in Figure 1.2 , then the critical path delay reduces to $(T_{m3} + T_A)$, i.e., 5 units. Thus the speed of the filter is increased Parhi (2007).

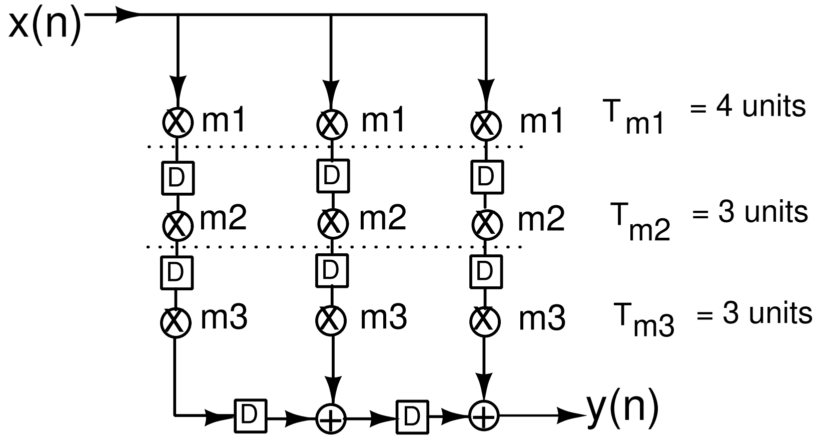


Figure 1.2: Fine-grain pipelined FIR filter (Parhi (2007)).

Pipeline stages have to be synchronized for the proper operation of digital circuits. In general, pipelining methods are broadly classified as synchronous and asynchronous based on the nature of synchronization present between pipeline stages. Most of the traditional pipelined circuits use synchronous pipeline paradigm, in which a global clock synchronizes various modules present in the circuit. Though the design complexity of the synchronous approach is quite less, handling the clock distribution becomes a challenging task in the era of the nano scale(Calhoun *et al.* (2008)). The modularity and managing the variable data rates are a few more issues in synchronous design. Moreover, synchronous pipelining is not reasonable for deeper pipelines. Reducing the pipeline depth beyond eight logic levels, reduces the amount of useful work per cycle, and increases the overheads associated with latches, clock skew, and jitter(Smirnov *et al.* (2004)).

The asynchronous design style is a viable option to provide reduced-cost solutions to several of synchronous design difficulties. Since asynchronous design replaces the centralized control with distributed local communication of stages, it has the potential to offer significant improvements in energy, performance, reliability, and scalability. In particular, asynchronous digital circuits can provide low power (components are activated only on-demand), high performance (some asynchronous systems have significantly lower latency and increased average throughput, rather than be bound to a worst-case clock rate) alternatives to their synchronous counterparts with excellent robustness to timing variability and modularity. Further, the unique capability of these clock less systems is that their behavior is independent of the implementation granularity. Hence, this work concentrates on asynchronous pipeline methods.

There are several asynchronous gate-level pipeline styles proposed in literature. In general, the overhead due to latches is more when a circuit is fine grain pipelined. Most of these asynchronous gate-level pipeline styles adopt dynamic gates to minimize this overhead. The output nodes of the dynamic gates can store the data and eliminate the need for explicit latches. The Pipeline Stages concatenated with Zero latches (PS0)(Williams (1990)), Precharge half buffer (PCHB) (Lines (1998)), Look ahead Pipelines (LP) (Singh and Nowick (2007*b*)), High Capacity (HC) (Singh and Nowick (2007*a*)), Self Precharge (SP) (Midhun *et al.* (2014)), and Asynchronous Pipeline based on Constructed critical Data Path (APCDP) (Xia *et al.* (2015)) are some of the popular gate-level asynchronous pipeline styles. The performance of these asynchronous pipelines depend upon the choice of data encoding and communication protocols used to construct the pipeline. The {Single-Rail (SR), Dual-Rail (DR)} and {4-phase, 2-phase} are commonly used data encoding schemes and communication protocols, respectively. Any combination from these sets can be picked to construct a pipeline design. The 2-phase protocol is transition-based, and the 4-phase protocol is level based. The pipelines based on 2-phase protocol are faster at the cost of design complexity, and the pipelines based on 4-phase protocol are simple to construct at the cost of performance. The Early Acknowledge (EA) protocol proposed by Yoneda *et al.* (2005), has the speed similar to 2-phase protocol and yet maintains the simplicity similar to 4-phase protocol. However, when the pipeline circuits are designed using this EA protocol, a stage should absorb the data before its preceding stage release the request signal. This imposes stringent timing constraints on the pipeline stages.

In data encoding schemes, the Single-Rail encoding offers high performing and area-efficient designs since it uses one wire to carry one data bit. The pipeline methods HC and Look ahead-SR are based on SR encoding scheme. This encoding needs a separate request wire and matched delays to ensure the validity of data present in the data bus. Hence in the presence of arbitrary delays, the SR design may fail to function correctly. On the contrary, the Dual-Rail encoding embeds the request signal within the data bus by using a pair of wires to carry a single data bit. The complemented data on these wires represents both the value of data and the request signal. The spacer or null data (0,0) on these wires indicate the release of the request signal. So every valid data transmission should be followed by a spacer. The DR design uses a completion detector at the receiver. The job of completion detector is to detect the presence of valid data on all data paths and produce acknowledgment to the sender. This encoding can give more robust designs even in the presence of random delays but at the cost of detection and encoding overheads. The PS0, PCHB, SP, Look ahead-DR are the few pipelines based on DR encoding.

In 2015, Xia.et.al (Xia *et al.* (2015)) have proposed a new pipeline method APCDP, which has adopted hybrid data encoding. The Hybrid encoding scheme combines the merits of both single-rail and dual-rail encoding schemes. In hybrid encoding, the critical path in each stage is made stable by building it with DR synchronizing logic gate (SLG) (Xia *et al.* (2010)). As the critical paths are made stable, it is sufficient if the completion detector checks only the critical path rather than detecting all the data paths. This, in turn, leads to the reduction of overhead in the completion detectors. The overhead in logic blocks is minimized by building non-critical paths with SR gates. Thus the APCDP pipeline style has combined the robustness of DR and simplicity of SR. However, throughput is the bottleneck of the APCDP pipeline method due to the return to zero nature of 4-phase protocol.

Asynchronous gate level pipelines can be used to design FIR filter for Partial Response Maximum Likelihood (PRML) read channel. These PRML read channels are used in optical and magnetic disk drives. High throughput FIR filters have become key requirement in the design of PRML read channels to support the ever-increasing data rates from magnetic and optical media (Singh *et al.* (2009)). These filters will also have to take care of the varying data rates during the disk read operations. Asynchronous filters can very well suit these requirements.

1.1 Motivation for the present work

Asynchronous pipelines have several benefits over their synchronous counterparts. Most of the existing asynchronous gate-level pipelines have adopted either single-rail or dual-rail encoding schemes to construct their data paths. The single-rail encoding is delay-sensitive, and the overhead of dual-rail encoding is very high. Though the hybrid encoding can combine the merits of SR and DR, there exists only one pipeline style named APCDP, based on this encoding scheme. APCDP pipeline can produce robust and simple circuits but it has poor performance due to the use of 4-phase protocol. From the perspective of this, there is a scope to improve the throughput of the hybrid data path by designing it with a high-speed communication protocols. In addition to this there are several variations of dynamic logic gates available in literature, which can be adopted in dual rail circuits to minimize the cycle time and hence maximize the throughput. Further, in literature, to the best of our knowledge there is no report on the design of fully asynchronous gate-level pipelined FIR filter. Hence it would be useful if the high throughput asynchronous pipelines are extended to design an FIR filter. Based on these, the following research objectives are derived.

1.2 Research Objectives

- To propose new/improved asynchronous pipeline architectures for high performance applications and compare them with the existing asynchronous pipeline architectures.
- To design and evaluate the performance of multiplier and adder circuits using proposed pipeline techniques.
- To design a high throughput asynchronous digital FIR filter suitable for read channel application.

1.3 Organization of the Thesis

The focus of this thesis is to realize new/alternative asynchronous pipeline methods for high throughput applications. Based on the background and motivation, the main interest lies in developing gate-level pipeline styles that adopt the hybrid encoding

scheme for the data paths.
Chapter 2, reports the background of asynchronous pipeline styles. Further, a brief discussion on performance parameters and classification of asynchronous gate-level pipeline styles is presented. The state of the art pipeline architectures reported in the literature are discussed.

Chapter 3, details the proposed asynchronous pipeline architectures. First, a brief introduction about the structure of proposed pipeline styles is given, and then the timing constraints present in the proposed methods are discussed. Further, the proposed pipeline styles are validated by designing a 4-bit,10-stage FIFO.

Chapter 4, describes the design of high throughput computational units based on the proposed pipeline methods and compares the simulation results with state of the art asynchronous pipeline based designs available in literature.

Chapter 5, extends the proposed EA-Hybrid pipeline method to complex circuits by designing a high throughput FIR filter suitable for PRML read channel applications. The adder and multiplier units of the filter are deeply pipelined using EA-Hybrid pipeline style. The simulation results show that the EA-Hybrid FIR filter concretely suits for the PRML read channel application.

Finally, Chapter 6, concludes this thesis by describing the contribution of this thesis and throwing highlights on possible future works.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

One of the dominant objectives of digital circuit designers is to discover ways of increasing the system speed. Pipelining is an important strategy for speed improvement. The pipelining technique divides a complex system into small functional blocks or stages and connects them in series. The state registers are embedded between adjacent stages for data buffering. The registers store the output of a particular stage and retain it as the next stage input. This kind of partitioning (pipelining) allows the systems to operate at high throughput by executing all the pipeline stages independently instead of waiting for the entire function to complete before the start of the next cycle(Parhi (2007), Beerel *et al.* (2010)). The synchronization between pipeline stages can be synchronous or asynchronous. This chapter briefly describes the synchronous and asynchronous pipelines. Further, the background of the asynchronous pipelines, their performance parameters are discussed in detail. Different asynchronous pipeline designs available for high throughput applications are then reviewed.

2.2 Synchronous Pipelines

It is a well-known fact that the synchronous paradigm has predominated the semiconductor industry for several decades. Most of the conventional processors use synchronous pipelines as the design complexity of the synchronous approach is quite less. Further, there are ample resources available in design tools for synchronous techniques. Figure 2.1 shows the synchronous pipeline method. In synchronous pipelines,

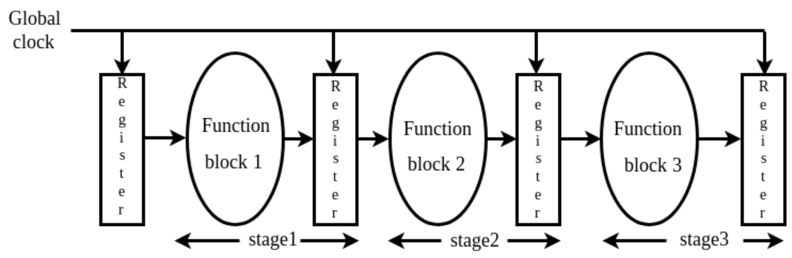


Figure 2.1: Synchronous pipeline(Mannakkara and Yoneda (2010))

a global clock is used to synchronize the communication among pipeline stages. The centralized clock synchronously controls all the pipeline registers and decides when they should exactly sample the input data. To ensure that the input data is stable and valid when registers are clocked, the clock period should accommodate the longest stage delay along with clock skew and a safety factor for all process, temperature, and supply voltage variations(Mannakkara and Yoneda (2010)). Hence in synchronous pipelines, the worst-case determines the critical path delay and thus the maximum clock frequency at which the entire pipeline should operate. In addition to this, handling the clock distribution is a challenging task in the era of a nanoscale (Calhoun *et al.* (2008)). Further, modularity and managing the variable data rates are some more issues in synchronous design. Though the synchronous elastic circuits(Carmona *et al.* (2009), Rezaei *et al.* (2017)), can manage the variable data rates, the problems associated with the global clock distribution still need to be solved. Hence the designers are directing their focus towards asynchronous methods.

2.3 Asynchronous Pipelines

There is no global clock in asynchronous pipelines. Figure 2.2 shows the structure of asynchronous pipeline. The synchronization between the pipeline stages is achieved locally from the handshake channels. These pipelines function based on the principle of initiating a data computation ('Req' signal indicates a computation initiation), waiting for its completion('ack' signal indicates corresponding computation completion), and then initiating the next one. If a computation completes early, then the next computation can start early. Hence, the speed of asynchronous pipelines depends on the computation time of data being processed, not on the worst-case timing. As a result, these pipelines exhibit an average-case performance.

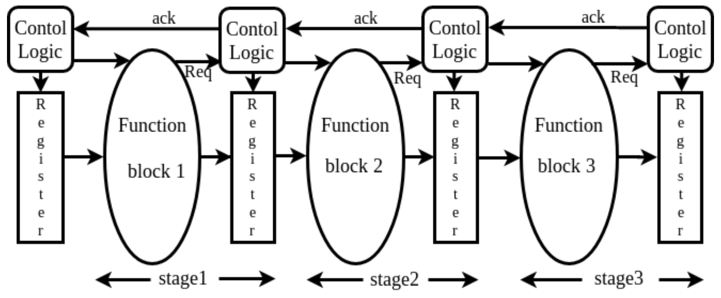


Figure 2.2: Asynchronous pipeline(Mannakkara and Yoneda (2010))

Further, these pipelines are free from all clock-related issues due to the absence of the global clock. In addition to this, the asynchronous design also has interesting properties like low power consumption, high performance, reduced electromagnetic emission, modularity, and the capacity to process variable data rate signals. Further more, asynchronous designs, in general, can adapt to timing variations caused by process, voltage, and temperature (PVT) extremes better than there synchronous counterparts Kuentzer *et al.* (2020*a*). In view of these benefits, Section-2.4 discusses the asynchronous pipelines in detail.

2.4 Background of Asynchronous Pipelines

The asynchronous pipeline method uses handshake control signals to synchronize the pipeline stages. These handshake signals are generated locally from the states of the pipeline stages and the external control signals. All the pipeline stages are driven by their corresponding handshake signals rather than by a single centralized clock. The delay due to complicated handshake channels often surpass all the advantages of asynchronous paradigm. Hence the selection of handshake channel is a key concern in the design of asynchronous pipelined systems, to fully enjoy their benefits. The Section-2.4.1, discusses in detail about the handshake channels.

2.4.1 Handshake Channels

In general the handshake channels are characterized via two parameters: communication protocols and data encoding schemes. {4-phase or Return-to-Zero (RZ), and 2-phase or Non-Return-to-Zero (NRZ)} are the prominent communication protocols.

{Single-rail/bundled data (SR), dual-rail (DR)} are popular data encoding schemes. Any combination from these two sets can be picked to construct the handshake channels.

2.4.1.1 Communication Protocols

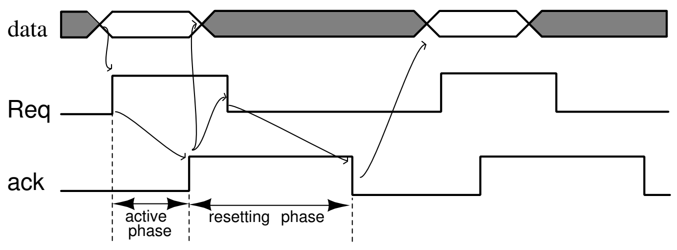
The 4-phase protocol is indicated in Figure 2.3(a). In this protocol, the request and acknowledge signals begin from zero level and return to zero level at the termination of a handshake cycle. It has 4 communication actions in a single cycle:

1. The sender makes a request by rising the request signal to high.

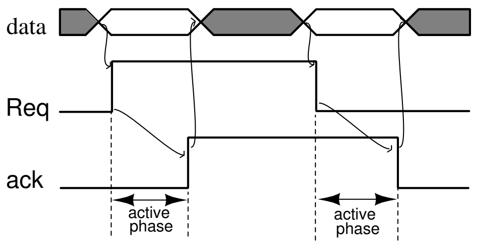
2. The receiver generates the corresponding acknowledgement by rising the acknowledge signal to high at the end of active phase. Here active phase is the time taken by the receiver to absorb the data after the rise of request.

3. The sender responds for the acknowledge by resetting the request signal

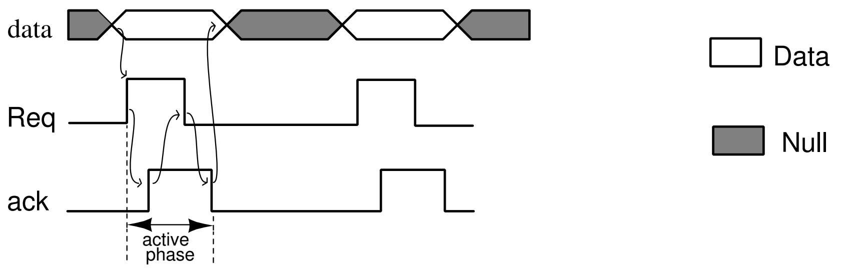
4. Receiver releases acknowledge to logic low at the end of the cycle.



(a) 4-phase protocol



(b) 2-phase protocol



(c) EA protocol

Figure 2.3: Asynchronous communication protocols (Mannakkara and Yoneda (2010)).

In resetting phase (i.e., time duration between the rise of acknowledge and reset of acknowledge), the 4-phase protocol does not allow the sender stage to perform any useful task. It simply waits for the reset of request and acknowledge to start the next cycle. Though the presence of resetting phase limits the speed of circuits designed using this protocol, the return to zero nature of signals make the hardware design simple. Figure 2.3(b) shows the 2-phase protocol. It is a transition-based communication protocol. In this, every transition of the request and acknowledge signals (i.e., $0 \rightarrow 1$ and $1 \rightarrow 0$) indicate an event. A transition on the Req signal makes a request. The receiver also acknowledges the completion of work with the same transition on the ack signal. The next cycle can start immediately after this. As this 2-phase protocol has only two communication actions in a handshake cycle, it can produce faster circuits. However, it is often difficult to construct the circuits that respond to transitions Spars and Furber (2002).

Yoneda *et al.* (2005) have introduced Early Acknowledgement (EA) protocol, which is an improved version of the 4-phase. This can produce faster circuits similar to 2-phase, by hiding the resetting phase of 4-phase protocol. Figure 2.3(c) shows the EA protocol. Unlike 4-phase, in this the receiver sets the acknowledge signal to high before it absorbs the data and both the request and acknowledge signals are released to low at the end of active phase. The next cycle can start immediately after this. Hence, this EA protocol removes the resetting phase implicit in the 4-phase protocol and yet maintains its simplicity by preserving the return-to-zero control signals. However, when the circuits are designed using this protocol, the receiver should absorb the data before the sender releases the request signal to low. This impose stringent timing constraints on the pipeline stages.

2.4.1.2 Data Encoding Schemes

In asynchronous circuit design, Single rail (SR) and Dual rail (DR) are the popular data encoding schemes. Figure 2.4 represents the SR encoding scheme. Similar to the synchronous method, the SR channels encode the information in the data path with a single wire per bit. The control path of these channels contain two wires, request and acknowledge, to indicate the timing instances of data initiation by the sender and data absorption by the receiver, respectively. These channels commonly lead to the circuits, which are efficient in terms of both the area and the power. However, the bundled data channels have a timing constraint, i.e., valid information should be

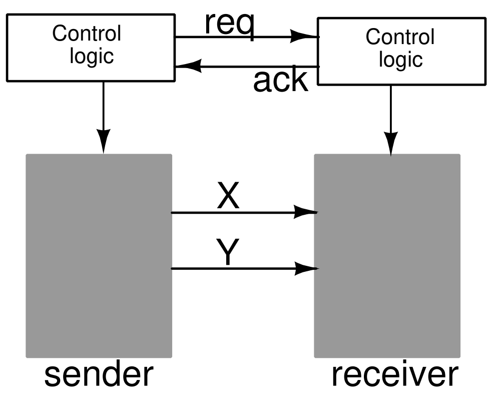
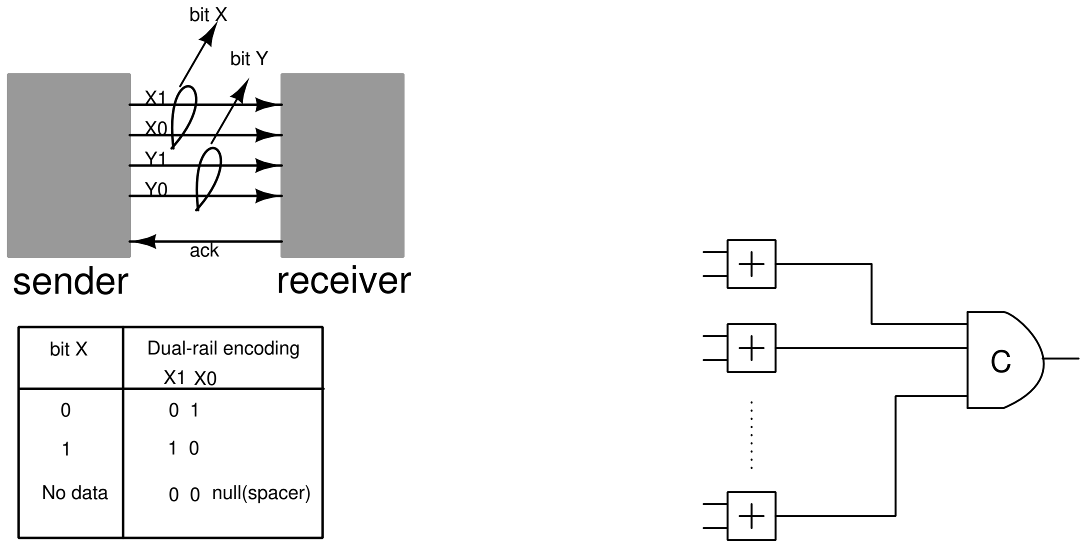


Figure 2.4: SR encoding (Nowick and Singh (2015))

present on the data path before the request signal reaches the receiver to ensure the correct operation. To meet this constraint, the request signal should be delayed by placing significant delay margins in its path, which might be a difficult task in the variable delay environments. In some of the recent works Kuentzer *et al.* (2020*b*), these timing margin issues are addressed.



bit X	Dual-rail encoding X1 X0
0	0 1
1	1 0
No data	0 0 null(spacer)

(a) **(b)**

Figure 2.5: (a) DR encoding (b) Completion detector (Nowick and Singh (2015)).

Figure 2.5(a) shows the DR encoding scheme. In the DR channels, the request signal is embedded within the data path by using two wires to carry a single data bit (for example to represent data X in the Figure 2.5(a) two wires X1 and X0 are used). The complimented data on these two wires are identified as the request made by the sender. The null data or spacer, i.e.,(0,0) on these wires are identified as the reset of the request signal. This kind of handshake channels lead to more robust designs. In this encoding scheme, the receiver uses a completion detector to generate an acknowledge signal. Figure 2.5(b) shows the completion detector circuit. This circuit checks all the data paths and produces acknowledgment only if valid data is present on all data paths. As the completion detector has to detect all the paths, the overhead in this circuit is very high. The overhead in the function blocks is also high as they need two wires to generate a single data bit. This overhead in function blocks and completion detectors increase with the increase in data size.

In 2012, Xia et al. have proposed a hybrid data encoding scheme. This scheme combines the merits of SR and DR encoding schemes by adopting DR encoding for the critical paths and SR encoding for the non-critical paths(Xia *et al.* (2012*a*)). The completion detectors in this scheme detect only the critical paths. Hence, the hybrid encoding scheme reduces the overhead in completion detectors as well as in function blocks. However, when using this encoding scheme, the critical paths should be made stable.

As the handshake channels provide localized communication among the stages, the performance of the asynchronous pipelines depends on the timing of individual stages as well as the timing of their neighbors. The following section introduces quantitative parameters used for measuring the performance of asynchronous pipelines in detail.

2.4.2 Performance Parameters of Asynchronous Pipelines

Latency, and Cycle time (inverse of Throughput) are the two parameters used to quantify a pipeline's performance(Spars and Furber (2002)).

- Latency (L): The latency is the time taken by a data item to propagate through the initial empty pipeline.

- Cycle time (T_{cycle}): The cycle time is defined as the time required to complete a handshake cycle. Considering 4-phase, Dual-rail pipeline with the forward propagation delay of a valid data item (t_V), reverse propagation rise delay of acknowledge ($R\uparrow$), forward propagation delay of the null data (t_E), and reverse propagation fall delay of acknowledge ($R\downarrow$), the cycle time can be computed as in equation(2.1).

$$T_{cycle} = t_V + R\uparrow + t_E + R\downarrow \tag{2.1}$$

The throughput is the inverse of the cycle time($1/T_{cycle}$) and is defined as the number of valid data items produced at the output in unit time.

2.5 Classification of Asynchronous Pipelines

Asynchronous pipelines are classified based on the logic style used for the data path. A designer must consider various criteria to adopt an approach for the intended application. The available design tools and design effort typically determines the choice of static versus dynamic data paths.

2.5.1 Static logic pipelines

Asynchronous pipelines which use static logic for the data paths are called static logic pipelines. All the existing static pipelines are constructed based on the single rail encoding scheme and use explicit latches to isolate the adjacent stages.

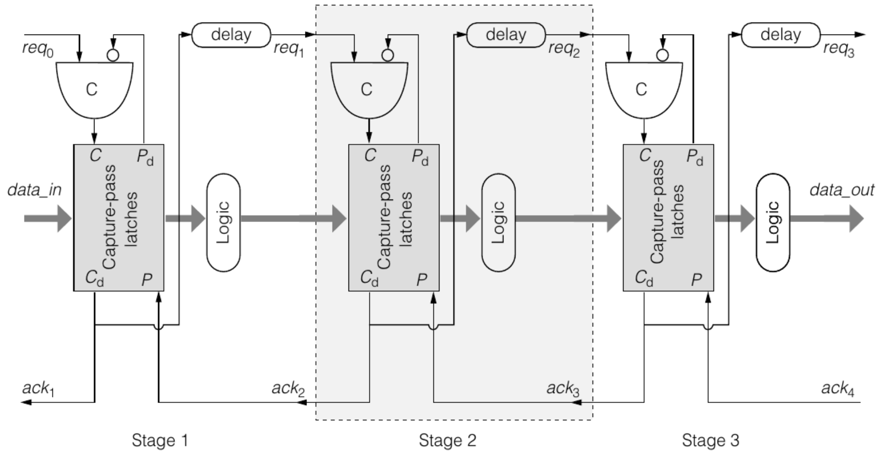


Figure 2.6: Sutherland's micropipeline (Sutherland (1989))

Sutherland (1989) proposed a static pipeline named as Sutherland's micro pipeline. The Figure 2.6 shows the pipeline structure. It uses a 2-phase protocol. The data path of the pipeline contains logic blocks and capture-pass latches. The logic blocks process the data and the latches act like storage elements and isolate the neighboring stages. The control path contains request (Req), acknowledge (ack) signals, and Muller

C-elements. Sufficient delay margins are added in all the request signal paths, to maintain the bundling constraint. The Muller C-element is a common asynchronous sequential component that generates control signals to the latches. If all the inputs of C-element are 1, then the output is 1, and if all the inputs are 0, then the output is 0. For other cases, the output preserves its previous state. The pipeline works based on capture-pass protocol. Initially, all the latches are in a transparent state (i.e., data can directly flow through them). As the data progresses through each individual stage latches, a transition occurs on the C input, which drives the corresponding latches into the opaque state. In this state, the latches store and protect the data from any further changes on the input channel of the stage. Once the data progresses through the next stage latch, it makes a transition on the P control input, which drives the current stage latch into the transparent state and allows the next data item to enter.

The Mousetrap pipeline, developed by Singh et al., Singh and Nowick (2007*c*), is a high-performance static pipeline and supports the use of standard cell methodology. Though its protocol is based on capture-pass logic similar to micro pipelines, it has lower overhead, less complex signaling, and simple control and data latches. Figure 2.7 shows the structure of the Mousetrap pipeline. A single 2-input exclusive-NOR(XNOR) gate generates the control signal for each stage. A single bank of level-sensitive D-latches that are available in standard cell libraries replaces the capture-pass latches of micro pipelines. This results in a simple control and storage structure.

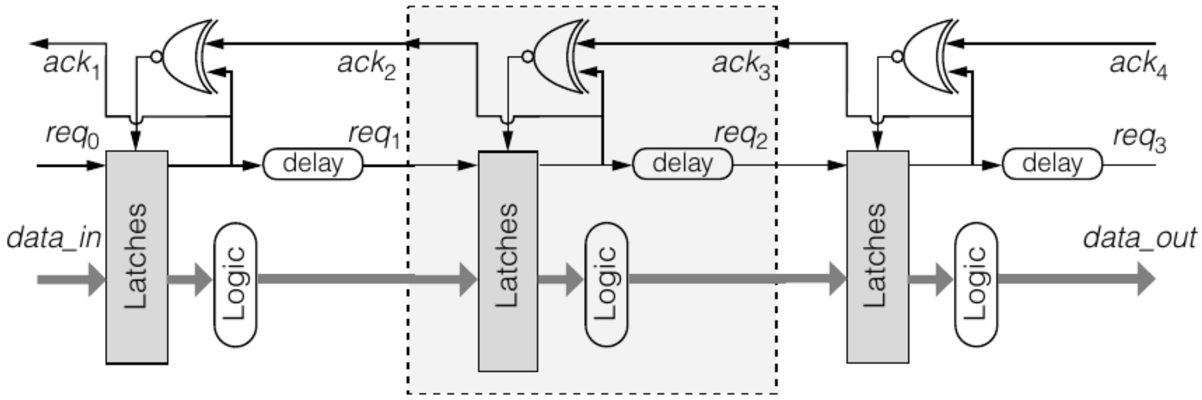


Figure 2.7: Mousetrap pipeline (Singh and Nowick (2007*c*))

Another static pipeline GasP was developed at Sun Research Laboratories Sutherland and Fairbanks (2001). The data path of GasP uses D-latches, similar to Mousetrap. However, the control path contains dynamic logic, custom gates, along with careful path timing. It has some interesting features. First, unlike the Mousetrap pipeline, the data latches are normally in an opaque state, and the pipeline functions based on the pulse-based latch control protocol. Second, the request and acknowledge signals between adjacent stages are multiplexed onto a single bi-directional wire called as state conductor. Initially, this wire is deasserted to logic high. An active-low signal is sent on this state conductor wire from left to right as a request to start a communication. An active-high signal is sent on this wire from right to left as an acknowledgment to stop the communication. At the end of the transaction, the state conductor returns to its default high state. Hence, the GasP pipeline combines the merits of both 2-phase and 4-phase protocols. There is a single round trip communication through the state conductor, like 2-phase protocol. However, the state conductor always returns to the same value after the completion of a transaction as in a 4-phase protocol. The forward synchronization operation of the GasP is similar to previous designs despite the protocol differences.

All the static logic pipelines can be implemented with standard cells and are well suited for automation. On the other hand, dynamic logic requires custom gates, in addition to specialized design and analysis tools for timing and noise immunity verification. Although the dynamic logic needs more significant design effort, it can yield greater performance. Further, the dynamic gates themselves can function as implicit latches, with clever control sequencing. Hence, this work focuses on the dynamic logic pipelines. The following section addresses the dynamic logic pipelines in detail.

2.5.2 Dynamic logic pipelines

Dynamic data paths are very common in high-performance digital systems and they are constructed with domino logic style. As there are no pull-up networks in domino gates, they can provide the benefits of reduced chip area and reduced switched capacitance. Further, domino logic pipelines have a unique feature of being latch-less. This latch-less feature avoids the need for registers between the pipeline stages and hence minimizes some of the key overheads of fine-grain pipelining. The following sections address the details of domino logic style and different existing asynchronous pipeline architectures built based on domino logic.

2.5.2.1 Domino logic style

Domino logic style is a method of designing logic functions using clocked gates. Based on the applied clock signal, the domino gates will be in any one of two phases: precharge or evaluate. The Figure 2.8 shows an AND gate realized using domino logic. During the precharge phase (i.e., when 'Clk' is low), the pull-down path is turned off, and the dynamic node 'x' is precharged to logic 1. In evaluate phase (i.e., when 'Clk' is high), the pull-down transistors preserve or discharge the dynamic node 'x' based on the applied inputs. A CMOS inverter is connected at the dynamic node 'x' to drive the output. This enables proper functioning of domino gates when cascaded. The added inverter also increases the drive strength of the gate. A weak feedback inverter is used as a keeper to recover the logic at the dynamic node. The keeper circuit enhances the immunity of the domino gate against charge loss and charge sharing problems Rabaey *et al.* (2002). As the domino logic contains only one PMOS transistor in its pull-up path, it has lower transistor count and input capacitance compared to conventional CMOS logic.

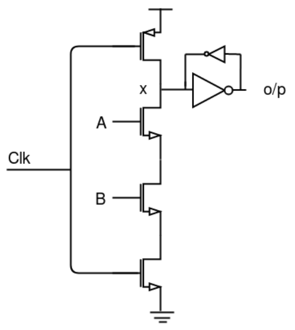


Figure 2.8: Domino logic AND gate (Rabaey *et al.* (2002))

The major drawback of domino logic is that it is a non inverting logic (i.e., only AND, OR and buffer gates can be implemented). As inverting logics (NAND, NOR, and inverter) are inevitable in most of the designs, this would limit the general applicability of pure domino logic. The Dual-rail domino logic is an approach to solve this issue. Figure 2.9 shows the Dual-rail domino AND/NAND gate. This gate needs both the complimentary and uncomplimentary inputs. Any arbitrary function can be implemented using Dual-rail domino logic. Though the DR domino logic uses more transistors than static CMOS and domino logic, this is very useful in logic designs, where it often needs both a signal and its complement simultaneously. If an inverter is used to generate a complementary signal, then the complemented signal is delayed with respect to the non-complemented signal, which causes timing problems, particularly in high-speed designs. The differential output capability avoids this problem.

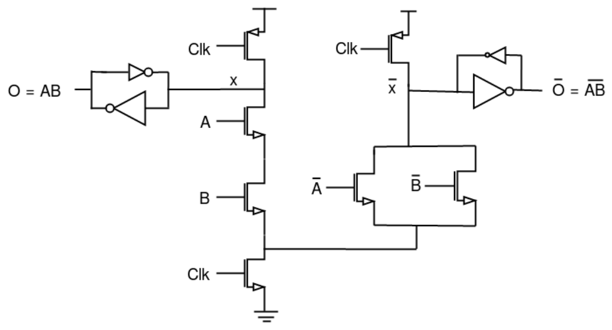


Figure 2.9: Dual-rail domino logic AND/NAND gate (Rabaey *et al.* (2002))

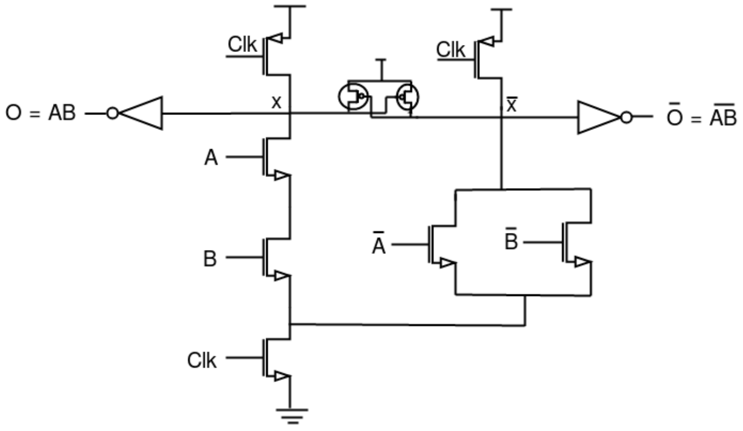


Figure 2.10: DDCSV logic AND/NAND gate (Anis *et al.* (2002))

The problem associated with the Dual-rail domino logic style is its sensitivity to noise. The size of keeper should be increased for compensating the low noise margin. However the high size keepers reduce the gate speed by increasing the contention current in the evaluation phase. The domino differential cascode voltage switch (DDCVS) logic can reduce the power and delay due to the contention Anis *et al.* (2002). Figure 2.10 shows the DDCVS AND/NAND gate. In this instead of weak feed back inverters, two cross connected PMOS transistors are used as keepers. The keeper transistors are turned off during precharge phase and at the beginning of evaluation phase. Hence the problem of contention is eliminated. This in turn improves the gate performance. A few asynchronous high throughput pipelines using different dynamic logic blocks are reported in literature. The following section reports these pipelines in detail.

2.5.2.2 Asynchronous dynamic logic pipeline methods

The William's PS0 (Williams (1990)), is the origin for all the asynchronous gate level pipeline structures. It follows the 4-phase dual rail protocol. The structure of PS0 pipeline is shown in Figure 2.11. Each pipeline stage contains a logic block (LB) and

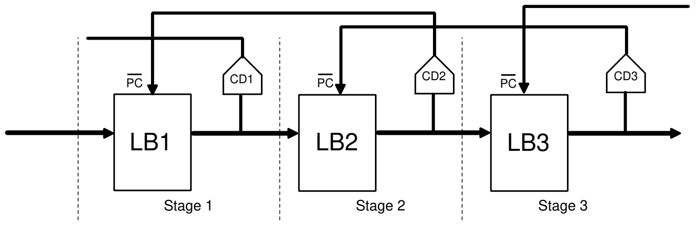


Figure 2.11: Block diagram of PS0 (Williams (1990))

a completion detector (CD). In each logic block, all the data paths are designed using dual rail domino gates. The precharge and evaluate phases of the domino gates are decided by a single control input $'\overline{pc}'$. The completion detector checks all data paths of its corresponding logic block and produces a handshake signal to control input $'\overline{pc}'$ of the preceding stage. The stages in the PS0 pipeline interact by following a simple protocol: "A stage is allowed to precharge phase whenever its next stage completes the evaluation and a stage is allowed to evaluate whenever its next stage completes precharge". By observing how a single data flow through an initially vacant pipeline, the sequence of events (i.e., from one evaluation to the next evaluation) for stage-1 can be given as follows:

1. Stage-1 evaluates and data flows to stage-2.
2. stage-2 evaluates and data flows to stage-3.
3. stage-2 completion detector detects the evaluation completion of its own stage and allows the stage-1 to precharge. At the same time, stage-3 evaluates.
4. stage-1 precharges and the completion detector of stage 3 allows stage-2 to precharge.

5. stage-2 precharges.
6. completion detector of stage-2 detects the precharge completion of its own stage and enables stage-1 to start the next evaluation.

Assuming all the stages are identical, the cycle time (i.e., time from one evaluation to next evaluation) of PS0 pipeline can be estimated from the above sequence of events as in equation(2.2).

$$T_{cycle.ps0} = 3 \cdot t_{ev} + 2 \cdot t_{cd} + t_{pc} \tag{2.2}$$

where t_{ev} is evaluation time of a stage, t_{cd} is delay of the completion detector and t_{pc} is precharging time of a stage. From the event sequence, we can observe that stage-1 is not allowed to process the next data until the stage-3 absorbs the current data. By that time stage-2 will be in precharge phase. Only the alternate stages are holding valid data items. Hence the maximum storage capacity of this pipeline is limited to 50%. Further the DR encoding overhead present in logic blocks and the detection overhead in completion detectors are restricting the use of PS0 pipeline in practical systems, demanding high throughput.

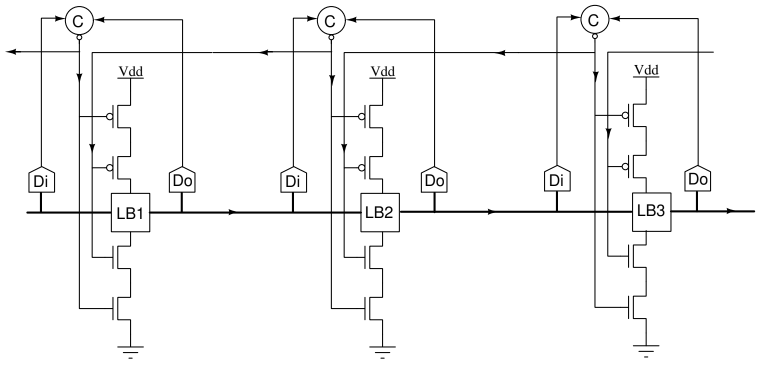


Figure 2.12: Structure of PCHB pipeline (Lines (1998))

Lines et. al. (Lines (1998)), proposed precharge half-buffer pipeline (PCHB). It is a robust pipeline style and its structure is shown in Figure 2.12. It consists of two completion detectors for each stage: one is placed at the input side of LB (Di) and the other is at the output side of LB (Do). The complete event sequence for a stage in the PCHB pipeline is similar to PS0, except that the PCHB stage checks its input data bits. The input completion detector will not allow a PCHB stage into the evaluation phase until the presence of valid bits at the input side. This kind of design detects skew over individual data bits in data paths. Although the PCHB pipeline is more robust, the overhead in control path is double as compared to the PS0. Furthermore, due to dual-rail encoding the PCHB also has the overhead in data path similar to PS0.

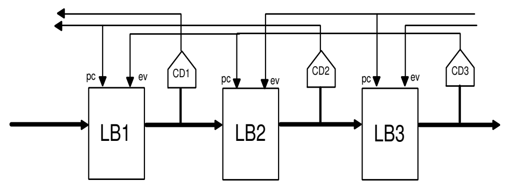
An improved version of PCHB named as Reduced Stack Precharged Half Buffer (RSPCHB) is proposed by (Ozdag *et al.* (2002)). In this, the transistor stack sizes in the logic block are reduced by eliminating the need for internal enable signals of the PCHB pipeline. Further, the throughput of RSPCHB is improved compared to PCHB by reducing the complexity in the right side completion detectors of the RSPCHB pipeline. However, the overhead of this pipeline is yet more compared to PS0, as it uses two completion detectors in each stage.

(Hoyer *et al.* (2002)), have proposed Locally Clocked dynamic logic pipeline method, which is based on single rail logic. This method relies on delay matching in the control path to produce handshake signals, and hence it eliminates the overhead associated with completion detectors. However, it has to satisfy timing constraints imposed by single rail logic. Further, in this pipeline method, every dynamic logic gate is followed by a latch section, which in turn increases the area and the latencies of the pipeline.

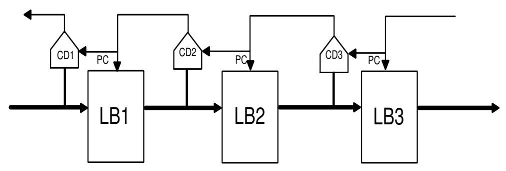
(Choy *et al.* (2001)), proposed a pipeline style named as Locally Distributed Asynchronous pipeline (LDA). This is based on 4-phase, dual-rail handshake protocol. The dual-rail gates in each logic block are designed using the DDCVS logic style. This pipeline can give better performance for smaller operand sizes. However, its performance degrades with higher data widths due to complex handshake circuits and the overhead of function blocks.

(Singh and Nowick (2007*b*)), proposed look ahead pipelines (LP), which are derived from PS0 pipeline style but operate at high throughput by adopting novel optimizations. The optimizations over PS0 approach are
1. Early evaluation: in this, a stage gets the control information from its succeeding stage as well as from the stages further down the pipeline.
2. Early done: in this, a stage generates acknowledgment to its preceding stage when it is about to finish an action, instead of generating after it has finished that action.
3. Mixed approach: This combines both early evaluation and early done schemes.
Different lookahead pipeline styles are developed by employing these optimizations to both dual rail and single rail data paths. The pipeline LP3/1 is based on the early evaluation scheme. The pipeline LP2/2 is based on early done and the pipeline LP2/1, which is the fastest among all look ahead pipelines is based on mixed approach.

In all LP pipelines the numbers 3/1, 2/2 and 2/1 represents the number of evaluation phases/number of precharge phases in a complete cycle. Though the look-ahead pipelines have reduced cycle times compared to PS0, their storage capacity is limited to 50% similar to PS0, as only the alternate stages can hold valid data items. The structure of LP3/1 and LP2/2 pipelines are shown in Fig.2.13a and Fig.2.13b, respectively. In Lp3/1 pipeline structure, a stage-N receives the precharge signal from its succeeding stage-(N+1) and evaluate signal from the stage-(N+2). Therefore stage-N can start the evaluation of next data immediately, without waiting for the precharge completion of its succeeding stage-(N+1). In LP2/2 pipeline structure, the completion detectors are located before the logic blocks and hence every stage signals its preceding stage when it is about to evaluate or precharge.



(a) LP3/1



(b) LP2/2

Figure 2.13: Structures of LP3/1 and LP2/2 pipelines (Singh and Nowick (2007*b*)).

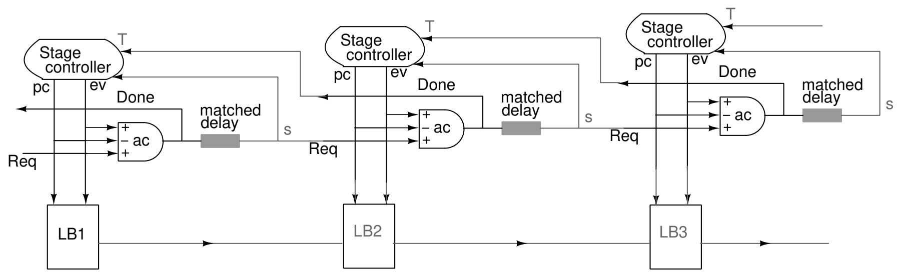


Figure 2.14: Structure of HC pipeline (Singh and Nowick (2007*a*))

(Singh and Nowick (2007*a*)), proposed High capacity pipeline style (HC). The structure of the HC pipeline is shown in Figure 2.14. This approach has fewer interactions between adjacent stages and hence operate at high throughput compared to all the existing pipeline methods. Further, it has full buffering capacity due to the adoption of a special phase called isolate phase to the logic blocks. In isolate phase, pull-up and pull-down paths of all the domino gates are turned off with the help of isolated control inputs. This, in turn, protects the outputs of the gates from all the input changes and the domino gates themselves act like full functional latches. In this pipeline, the data path is encoded using single rail encoding scheme and an asymmetric C-element (ac) is used to generate the request and acknowledge signals.

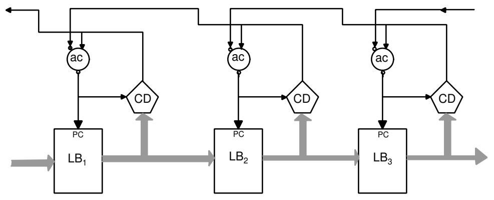


Figure 2.15: Structure of SP pipeline (Midhun *et al.* (2014))

(Midhun *et al.* (2014)), proposed self precharge pipeline (SP). It is based on the dual-rail encoding scheme. The structure of the SP pipeline is shown in Fig.2.15. The SP pipeline can provide higher throughput compared to all existing dual rail pipeline styles. In look-ahead and PS0 pipelines precharge of a stage depends on the evaluation of its succeeding stage. In SP pipeline method, a stage precharges by itself through a self precharge process, hence it is called as SP pipeline. Due to dual-rail encoding, SP pipeline also has the overhead in the data path and control path as in PS0.

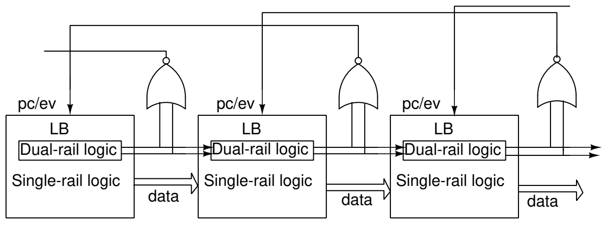


Figure 2.16: Structure of Asynchronous pipeline based on constructed critical data paths (Xia *et al.* (2012*b*))

(Xia *et al.* (2015)), proposed a novel 4-phase pipeline style with hybrid data paths called APCDP. Figure 2.16 shows the structure of APCDP pipeline. This method has combined the merits of SR and DR encoding shemes. In this the critical data paths are constructed with dual rail synchronizing logic gates (SLG) Xia *et al.* (2010) and non critical paths are built by single rail gates. Hence this pipeline method has less detection and encoding overheads as compared to PS0 and PCHB pipelines. Further, the APCDP pipeline is more robust compared to SR pipeline methods, as the hand shake signals are generated by detecting the actual LB outputs rather than from the replication of LB delay. However due to the use of 4-phase protocol, the APCDP has high cycle time and it's storage capacity is limited to 50%. All the domino logic pipeline techniques discussed are summarized in Table 2.1. In addition to SR and DR pipelines, there are some other pipelines like Single-Track asynchronous pipeline Templates Ferretti and Beerel (2002),Yong and Runde (2005), in the state of the art. However, these pipelines are based on 1 of N coding. As 1 of N coding pipelines need more hardware than the Dual rail pipelines, in this work we have not focused on this encoding.

Table 2.1: Summary of asynchronous gate-level pipeline styles

Work	Pipeline style	Data encoding	Pros	Cons
Williams and Horowitz (1991) Choy *et al.* (2001)	PS0 LDA	DR	Robust Simple protocol	Overhead in LB and CD High cycle time 50% Storage capacity
Lines (1998), Ozdag *et al.* (2002)	Precharge half buffer, Reduced stack PCHB	DR	Robust Simple protocol	2 times overhead than PS0 High cycle time 50% Storage capacity
Hoyer *et al.* (2002)	Locally Clocked dynamic logic	SR	High performance	Delay sensitive Need additional latch stages
Singh and Nowick (2007*b*)	Look ahead pipelines	DR/SR	High performance Simple protocol	Delay sensitive 50% Storage capacity
Singh and Nowick (2007*a*)	High capacity	SR	High performance 100% Storage capacity	Delay sensitive
Midhun *et al.* (2014)	Self precharge	DR	High performance	Overhead in LB and CD 50% Storage capacity
Xia *et al.* (2015)	APCDP	Hybrid	Robust Less overhead	50% Storage capacity High cycle time

2.6 PRML read channel

One of the notable applications of the asynchronous gate-level pipeline styles is in the design of the FIR filter for Partial response maximum likelihood (PRML) read channel Pearson *et al.* (1995). To support the high data rates and substantial storage requirements, most of the hard disk drives prefer to use PRML read channels. The job of a read channel is to transform the noisy data collected by read head into a clear stream of 1 and 0 symbols. Figure2.17 shows the structure of a PRML read channel. First, the data picked up by read head is processed by an analog front end. This analog block consists of an amplifier and a high-frequency noise filter. The sampler converts the analog block output to digital data. This digital data progresses as input to the digital unit that performs the equalization process to remove the Inter Symbol Interference (ISI) noise, and the detection step to decode the data values. The digital

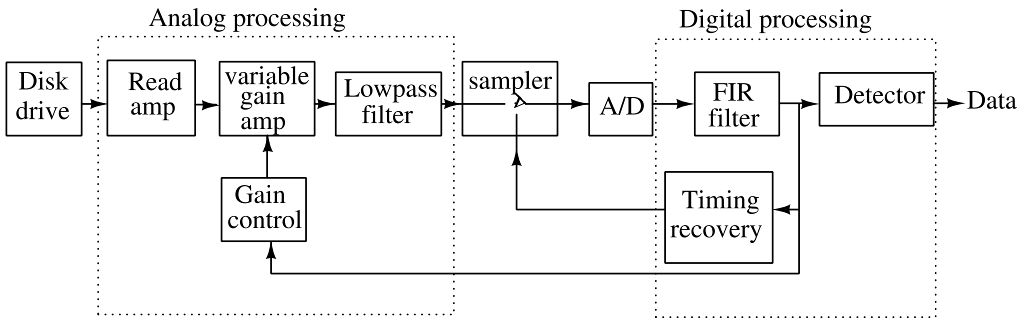


Figure 2.17: Block diagram of disk drive read channelKi *et al.* (2000).

FIR filter can do the equalization process. The design requirements of the FIR filter for the read channel application are as follows.

- The filter must be able to operate at high throughput (upto 1 Giga-items/sec in 180 nm technology), to handle high data rates available from the disk drives.
- The latency of the filter should be less (in the order of ns) as it is present in the feed back loop of read channel.
- The filter should be able to handle wide variations in the data rates, as the data rate changes up to a factor of 5 when the read head moves from one track to the other.

The throughput of FIR filters can be increased by pipelining the data path. High throughput asynchronous pipelines are suitable for such applications. FIR filter is a nonlinear structure. Implementing the simple linear structures using all the existing domino pipelines is straight forward. However, there are two challenges namely, Forks and Joins, involved when extending all these pipelines to non-linear structures. Figure 2.18 shows the Fork and Join structures. In Fork structure, a stage has to be synchronized with multiple destinations, whereas in Join structure, a stage should be synchronized with multiple sources. (Ozdag *et al.* (2002)) addresses how to extend existing high-speed pipelines to non-linear structures.

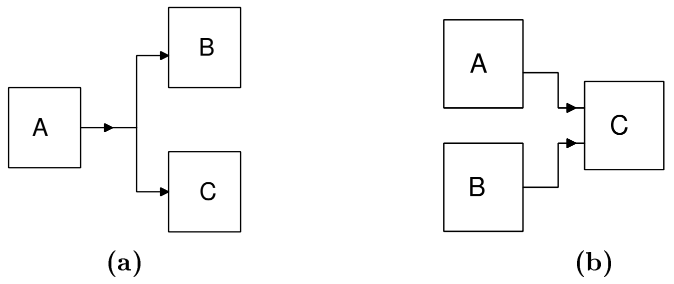


Figure 2.18: (a) Fork (b) Join

In 2009, (Singh *et al.* (2009)) have implemented a mixed synchronous and asynchronous HC-FIR filter for PRML read channel. This filter is based on distributive arithmetic (DA) architecture. The DA architecture is a multiplier less architecture. In this, the multiplication operation is efficiently performed by fetching the pre-computed results from a look-up-table (LUT). The partial sums collected from the LUT are added to get the final filter output. The adder part of the HC-FIR filter has been implemented using high capacity pipeline method. The HC-FIR filter was able to operate at high throughput with minimum latency. However, updating the look-up tables with precomputations is difficult if the input data rate changes very frequently.

2.7 Summary

A detailed review of different gate-level pipeline methods reported in literature has been presented. The dynamic asynchronous pipeline styles are suitable for high throughput applications. In the literature, ample research has been done on asynchronous dynamic logic pipeline designs. However, the research has concentrated either on single rail logic or dual rail logic. When the data paths are designed with single rail logic a timing constraint should be satisfied (i.e., request should reach receiver after the data is stable and valid at the receiver input). There are no timing constraints in the dual rail logic and it is delay insensitive, but the overhead of this logic is very high. The hybrid logic, which combines the merits of single rail logic and dual rail logic has followed 4-phase protocol, due to which its throughput is less and storage capacity is limited to 50%. Hence there is a need to propose new pipeline methods which can address all these issues. One of the important application of high throughput asynchronous pipelines is in the design of digital FIR filter suitable for PRML read channels. To the best of our knowledge there is no report in the literature on the design of fully asynchronous FIR filter.

Chapter 3

HIGH CAPACITY HYBRID PIPELINES

3.1 Introduction

This chapter proposes two novel asynchronous pipeline methods suitable for high-performance applications. The proposed methods are built based on hybrid data paths. Both the proposed methods are fine-grain pipelined, i.e., the depth of each pipeline stage is one gate level. As dynamic gates can support best circuit performance (Lai and Hwang (1997)), domino logic is chosen to design the proposed pipelines. In addition to high speed, the domino gates can also build a latch-less pipeline structure. The output node of a domino gate functions like an implicit latch. In both proposed designs, each stage can hold a distinct data item with the help of isolate phase. This feature leads to the design of high throughput circuits with full buffering capacity. Further, the use of hybrid data paths simplifies the design of handshake controllers as well as improves the robustness of the pipelined circuits. The first proposed pipeline style has adopted the EA protocol and hence named as high capacity early acknowledge hybrid pipeline (EA-Hybrid). The second pipeline style is based on simple 4-phase protocol and named as high capacity hybrid pipeline with post detection (PD-Hybrid).

3.2 Proposed Pipeline Styles

The structure of proposed EA-Hybrid and PD-Hybrid pipeline styles are shown in Figure 3.1, and Figure 3.2, respectively.

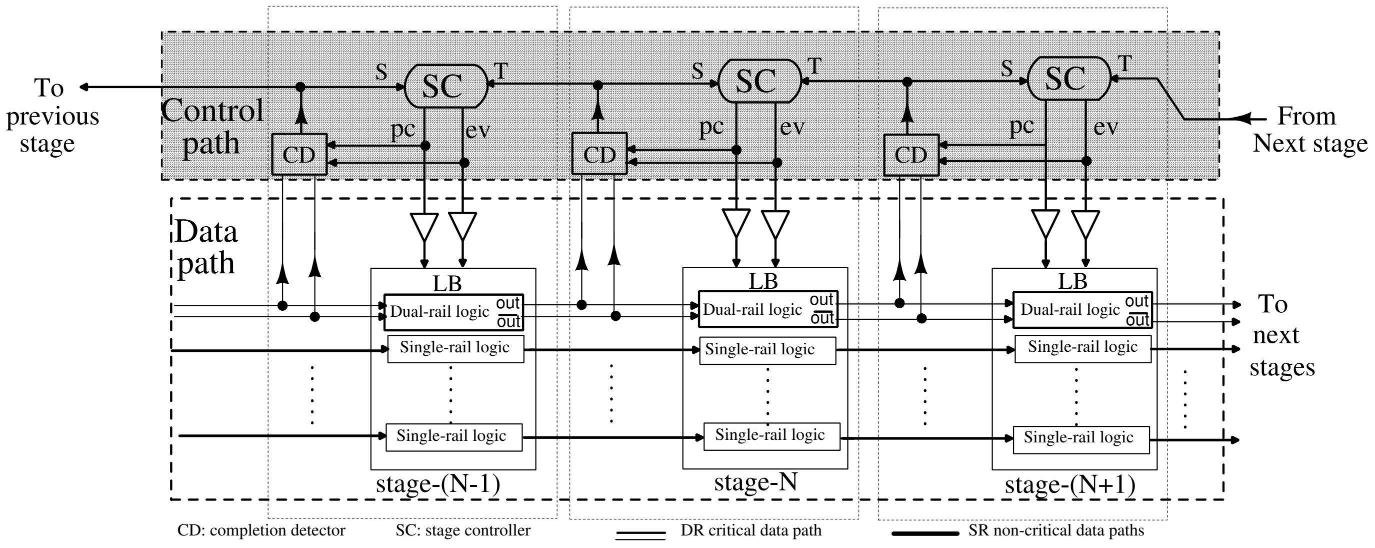


Figure 3.1: Structure of Early acknowledged hybrid logic pipeline (EA-Hybrid).

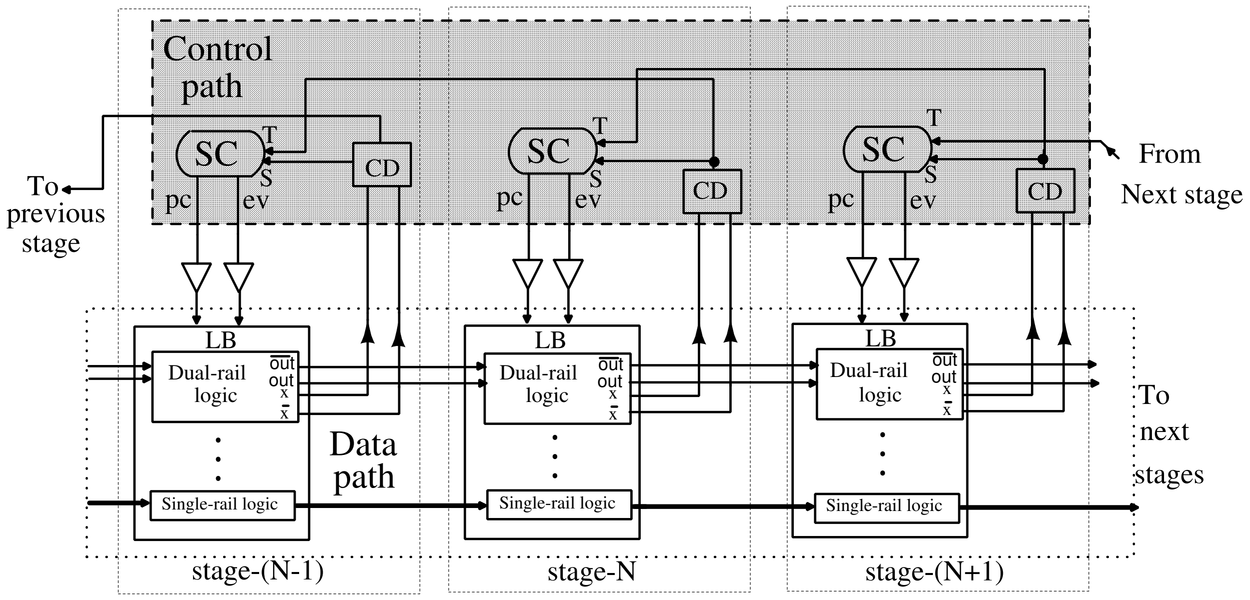


Figure 3.2: Structure of High capacity hybrid logic pipeline with post detection (PD-Hybrid).

The protocol of both the proposed methods is similar. However, they differ in the way how the handshake signals are generated. Figure 3.3 shows a flow chart that describes the sequence of phases that a stage-N has to go through during the pipeline operation, for both the proposed methods. A stage-N starts evaluation if it is ready for evaluation, and its preceding stage supplies valid data. After evaluating its input data, the stage-N enters into the isolate phase by itself and waits for the acknowledgment from its succeeding stage(i.e., stage-(N+1)). Once the stage-N receives an acknowledgment from its next stage, it enters into the precharge phase. After precharge, the stage-N starts the next cycle on the arrival of valid data at its input. Here, there is only one synchronization action between the neighboring stages. This is one of the reasons for the high throughput of the proposed methods.

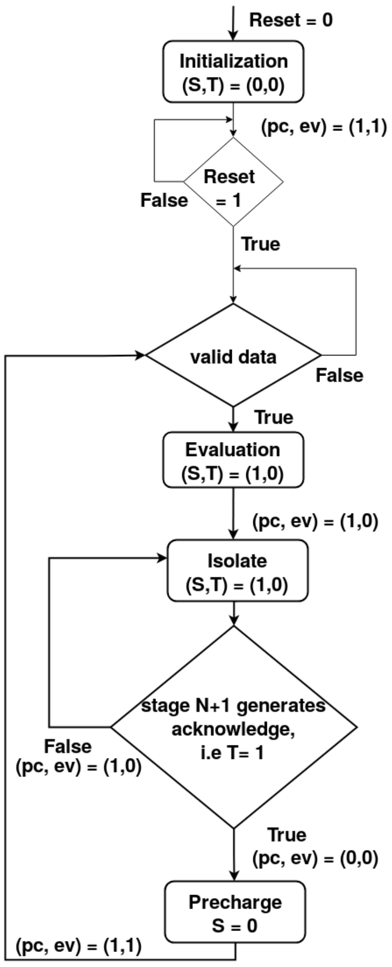


Figure 3.3: Sequence of phases in a stage-N.

In both the proposed methods, the depth of each pipeline stage is one gate-level. Every pipeline stage comprises of a logic block (LB), a completion detector (CD), and a stage controller (SC). The data path is constructed using the logic blocks, while the control path is built using the completion detectors and the stage controllers. The job of the data path is to carry out the computations, and the job of the control path is to supply control signals required for the synchronization of data path. Section-3.2.1 and 3.2.2 discuss the complete details of the data path and control paths, respectively.

3.2.1 Data Path

Data paths in both the proposed pipeline methods are constructed using hybrid logic. In every logic block, the critical path is build using DR domino synchronizing logic gates (SLG)(Xia *et al.* (2010)), while the non-critical paths are build using SR domino gates. The adoption of SLGs, ensure the constant delay through the critical paths for all the applied data patterns.

3.2.1.1 Logic block

Figure 3.4 shows an example of single-rail and dual-rail synchronizing logic domino gates present in a logic block of the proposed pipeline structures. The PMOS transistor M_r precharges the logic gates during initialization. The weak feedback inverter in the

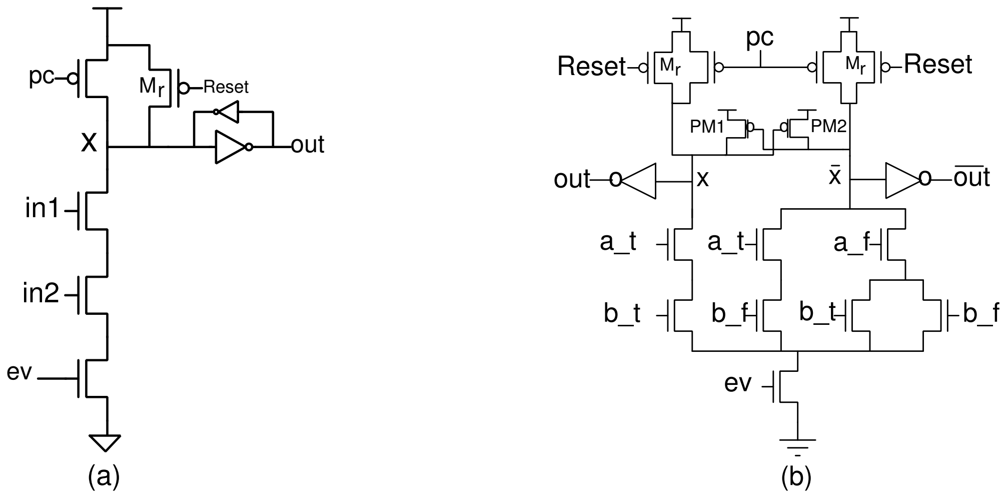


Figure 3.4: Domino logic AND gate. (a) Single rail gate. (b) DR-SLG.

SR gate acts as a keeper and stops the false discharge of the node x. In DR-SLG, instead of weak feedback inverters, two PMOS transistors PM1 and PM2 are used as keepers to minimize the contention cureent. The keeper transistors are turned OFF during the precharge phase and at the beginning of the evaluation phase. Hence, the problem of contention is eliminated. The traditional DR gates are not suitable for constructing stable critical paths. Because in the traditional dual-rail gate, each pull-down path contains the different number of transistors, and more than one pull-down path is activated for some input combinations. Due to this, the gate delay depends on the applied input data. Hence, dual-rail synchronizing logic gates (DR-SLG) are adopted in proposed pipeline methods to construct stable critical paths.

The functionality of DR SLGs is similar to traditional DR gates. In DR SLGs, all the pull down paths contain equal number of transistors and for an applied input combination only one pull down path will be activated. Hence the discharge time for the nodes x and $\overline{x}$ is almost same for all the applied inputs. This in turn leads to the constant delay of the gate for all input patterns. Table 3.1 shows all the pull down path states of SLG AND gate for various combinations of the applied inputs. In addition to evaluate transistor, each pull down path has two transistors and only one path is active for any given input combination, ensuring same discharge time.

Table 3.1: SLG AND gate pull down paths state

input data	**pull down paths state**			
(a_t , a_f) (b_t , b_f)	**(a_t , b_t)**	**(a_t , b_f)**	**(a_f , b_t)**	**(a_f , b_f)**
(0,1) (0,1)	**OFF**	**OFF**	**OFF**	**ON**
(0,1) (1,0)	**OFF**	**OFF**	**ON**	**OFF**
(1,0) (0,1)	**OFF**	**ON**	**OFF**	**OFF**
(1,0) (1,0)	**ON**	**OFF**	**OFF**	**OFF**

Similar to HC-SR, the SR and DR gates of proposed styles have two isolated control inputs evaluate ('ev') and precharge ('pc'). A logic block enters into precharge state if both the control inputs are set to logic-0 and the logic block enters into evaluate state if both the control inputs are set to logic-1. If 'pc' is set to logic-1 and 'ev' is set to logic-0, then the logic block will be in isolate state. In this state both pull-up and pull-down paths of all gates are turned OFF and hence the output of logic block is effectively isolated from its inputs. When a stage-N is in isolate phase, as the inputs of stage-(N+1) are not going to be disturbed, the stage-(N-1) can precharge as well as can safely evaluate the new data item. This in turn allows the neighboring stages to hold the distinct data items without any need for separation by spacers. Hence the proposed styles have high throughput and buffering capacity up to 100%.

3.2.1.2 Stable critical data path construction

In each logic block, critical paths are constructed using SLGs to ensure constant delay for all the input data patterns. While interconnecting the stages of pipeline, critical path of one stage should be connected to the critical path of next stage as shown in the Figure 3.5. This interconnection ensures that the critical path elements (SLGs) will start evaluation after all the data have arrived at the stage. As a result a stable critical path is developed throughout the pipeline.

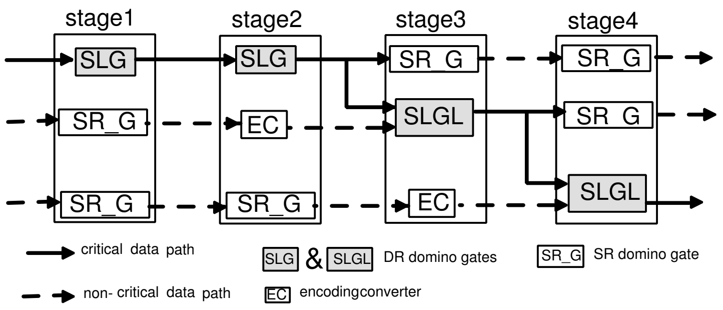


Figure 3.5: An example data path of the proposed pipeline styles.

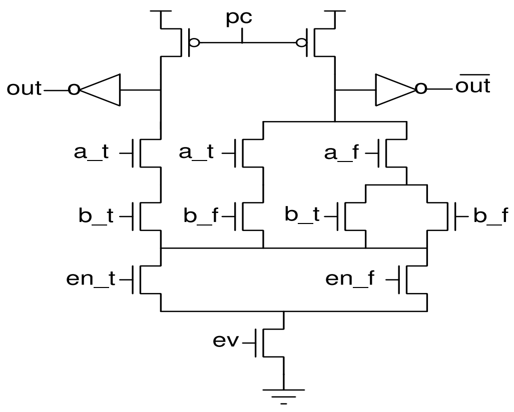


Figure 3.6: SLGL AND gate.

In general, a gate with more number of inputs take longer time to discharge its output nodes, as it contains more number of transistors in the pull down path. Hence in every logic block, a gate with maximum number of inputs is considered as the critical path and is build with SLG. If there are more gates with maximum inputs, then the gate which has a link to the next stage SLG is considered as the critical path. This builds a natural link between SLGs. If SLGs of neighboring stages, N and N+1 have no natural link, then the N+1 stage SLG should be replaced with SLG with latch function(SLGL)(Xia *et al.* (2015)). Figure 3.6 shows the SLGL gate. This gate is similar to the SLG, except it has two additional enable transistors in its pull down path. The outputs of SLG in stage-N should be connected to the enable inputs of SLGL in stage-(N+1). This situation is present between stage-2 and stage-3 of the data path, shown in Figure 3.5.

3.2.1.3 Robustness of critical path

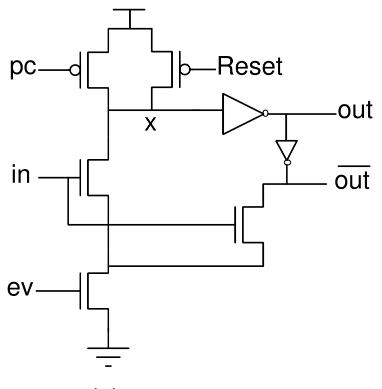
After the construction of stable critical path throughout the pipeline, in each stage, the DR-SLG always operates with maximum delay even when compared to the other gates with same number of inputs. The reasons are listed below.

- The SLG has more number of transistors in its pull down path, which increases the delay of the gate.
- The outputs of SLG are connected to the next stage completion detector and SLG, which results in maximum load on the SLG compared to other gates.

If required, the robustness of the critical path can be improved further by reducing the delay of the non critical data paths with proper transistor sizing. This will increase the power consumption of the pipelined circuit without affecting the throughput. The other way is to increase the delay of the SLGs by using minimum transistor sizes compared to other gates. This will not increase the power consumption but will increase the cycle time.

3.2.1.4 Encoding converter

Since the critical path in each stage is made stable using SLG, it is sufficient if the completion detector detects simply the critical path rather than detecting all data paths. Hence the overhead in completion detectors is reduced, and in the non-critical paths, the overhead can be minimized by employing single rail gates in their design rather than dual-rail gates. However, there will be compatibility problems if a single rail gate links to a dual rail gate. To solve this problem, an encoding converter is suggested by Zhengfan XIA et.al Xia *et al.* (2015). A similar converter design is adopted for the proposed pipeline styles but with the decoupled control inputs (pc,ev) for the pull-up and pull-down networks, as shown in the Figure 3.7(a). Decoupled control inputs are required to allow the converter into the isolate phase. During initialization period the node 'x' is precharged to logic-1 and the output node $'out'$ ($\overline{out}$) is at logic-0 (logic-1). The $'pc'$, and $'ev'$ signals are at logic-1, which keep the converter ready for evaluation. After the release of the Reset signal, the converter gets the input data and starts evaluation. Now, if the input $in = 0$, there is no discharge path for the node 'x' and the value at this node is preserved as logic-1. After the evaluation phase, the stage controller allows the converter to isolate phase by changing (pc,ev) to (1,0). In this phase, as both the paths pull-up and pull-down are turned off, the output nodes maintain their previous state. Once a stage gets acknowledgment from its next stage, the stage controller allows the converter to enter into the precharge phase by assigning (0,0) to (pc,ev). The Figure 3.7(b) shows the converter truth table. Though the converter doesn't produce the spacer in precharge phase, it will not cause the data transfer errors because of its high conversion speed, for most of the applications.



(a)

Reset	pc ev	in	out $\overline{out}$
0	1 1	X	0 1
1	1 1	0 1	0 1 1 0
1	1 0	X	previous state
1	0 0	X	0 1

(b)

Figure 3.7: (a) Encoding converter. (b) Function table.

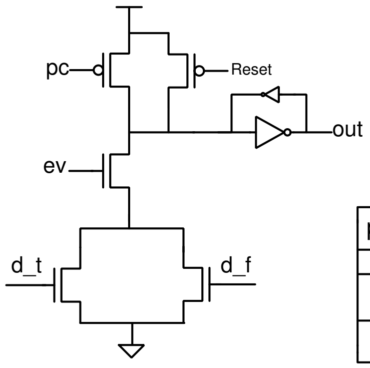
3.2.2 Control Path

In the control path of both the structures, every stage comprises of a completion detector (CD), and a stage controller (SC). The purpose of a completion detector is to identify the evaluation completion and precharge completion of its own stage. The stage controller produces the control signals: evaluate('ev') and precharge('pc'). As the two proposed pipeline styles are based on two distinct communication protocols, the construction of the control path is different for these two styles. The completion detector of EA-Hybrid logic checks the arrival of valid data at the input side of a stage, whereas the PD-Hybrid checks the data validity at the output side of a stage, as shown in Figure 3.1 and Figure 3.2, respectively. However, both the pipeline styles adopt identical stage controllers.

3.2.2.1 Completion Detector

A completion detector in the EA-Hybrid pipeline has to send acknowledgments to its preceding stage before the current stage finishes data evaluation or precharge. With this motive, the completion detectors are positioned before the logic blocks, as shown in the Figure 3.1. To anticipate the precharge or evaluation completion of their corresponding stages, the completion detectors must check the status of their corresponding stages (i.e., whether the stages are ready for evaluate/precharge/isolate) in addition to the critical data path detection. Thus a completion detector has four inputs: decoupled control inputs 'pc' and 'ev' of its corresponding stage and dual-rail data bits (d_t , d_f) of the critical path from the preceding stage. The output of the completion detector is fed to stage controllers of the preceding stage and its stage. The output of completion detector is set to high when the previous stage supplies valid data on critical path (i.e., (0,1) or (1,0)) and when its stage is ready to evaluate (i.e., pc=ev=1). It is set to low when its stage is ready to precharge (i.e., pc=ev=0) and remains unchanged when its stage is ready for the isolate phase. The circuit and truth table of the completion detector are shown in Figure 3.8.

The completion detectors in PD-Hybrid style produce acknowledgments to the preceding stages after their corresponding stages finish the precharge or evaluation phases. Hence they are placed after the logic blocks. They detect the critical data path of their stages at the output side to produce acknowledgments to the preceding stages. As the completion detectors are performing post-detection, they do not detect the status of their corresponding stages. This, in turn, leads to a simplified completion detector design. The complimented data at the output nodes $x, \overline{x}$ of critical path SLG, should be identified as the evaluation completion of the stage, and logic-1 on both the nodes should be identified as the precharge completion of the stage. Hence a simple static NAND gate shown in Figure 3.9, is used as the completion detector in this



(a) CD circuit

pc	ev	d_t	d_f	out
0	0	x	x	0
1	1	1 (or) 0	0 1	1
1	0	x	x	previous state

(b) CD truth table

Figure 3.8: Completion Detector of EA-Hybrid logic pipeline style.

pipeline style.

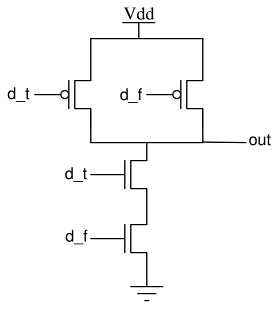


Figure 3.9: Completion Detector of PD-Hybrid logic pipeline style.

In the EA-Hybrid pipeline style, a DR-SLG in stage-N provide inputs to the completion detector and to the gates of stage-(N+1) from its *out* and $\overline{out}$ nodes as in Figure 3.1, where as in PD-Hybrid pipeline, the DR-SLG of a stage-N provide inputs to its completion detector from the output nodes $x, \overline{x}$, and to the stage-(N+1) gates from the nodes $out, \overline{out}$ as shown in Figure 3.2. This results in reduction of the evaluation time contribution to the cycle time of PD-Hybrid, by eliminating the delay of static inverters present between $x(\overline{x})$ and $out(\overline{out})$ nodes of DR domino gates.

3.2.2.2 Stage Controller

The stage controller receives input 'S' from the completion detector of its corresponding stage, input 'T' from the completion detector of its next stage and produces the control signals 'pc' ,'ev'. These control signals must be connected to the logic blocks through additional buffers to avoid the loading effect on stage controller. Based on these control signals, a logic block will enter into precharge, evaluate or isolate states. The Figure 3.10 shows the circuit of stage controller. It has a 3-input NAND gate to generate 'pc' control signal and an inverter to generate 'ev' control signal. From the Figure 3.10, it can be seen that 'ev' signal depends only on 'S', where as 'pc' signal depends on both 'S' and 'T'.

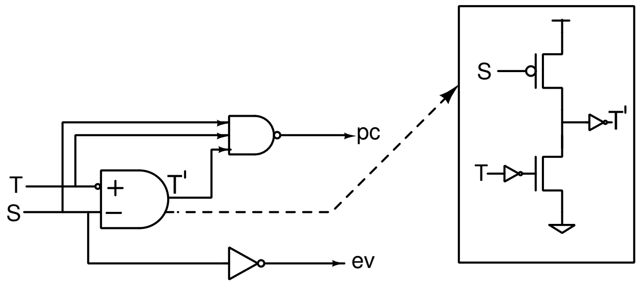


Figure 3.10: Stage Controller.

Table 3.2: Function table of the Stage Controller

S T	$T^{'}$	pc ev	stage 'N' state
0 0	x	1 1	evaluate
1 0	1	1 0	isolate
1 1	1	0 0	precharge
0 1	0	1 1	evaluate
1 1	0	1 0	isolate
1 0	1	1 0	isolate

An additional signal $T^{'}$ is also employed to produce the 'pc' control signal. A stage-N will precharge and then evaluate by itself, through a self evaluate process, after receiving the acknowledgement (i.e T=1) from stage-(N+1). This acknowledgement should be released, i.e., T should become 0, by the time stage-N completes evaluation of new data. If stage-(N+1) is blocked or too slow it may not release T, while the stage-N may interpret this T=1 as the acknowledgement for the new data and enter into precharge phase instead of going into isolate phase. Because of this, stage-N starts next cycle before stage-(N+1) absorbs the current data. To overcome this ambiguity a signal $T^{'}$ is used in the generation of pc signal. The $T^{'}$ is produced using an asymmetric C- element(Singh and Nowick (2007*b*)). The asymmetric C-element designed for proposed pipeline styles is shown in the Figure3.10. The T' signal can note effectively whether stage-(N+1) has absorbed a data item or not. The signal T' is reset to low immediately after stage-N is precharged (i.e S=0), and is set to high after the stage-(N+1) has released the T to low. This can be understood more clearly from function table of the stage controller shown in Table 3.2. Figure 3.11, shows the complete circuit diagram of a single stage.

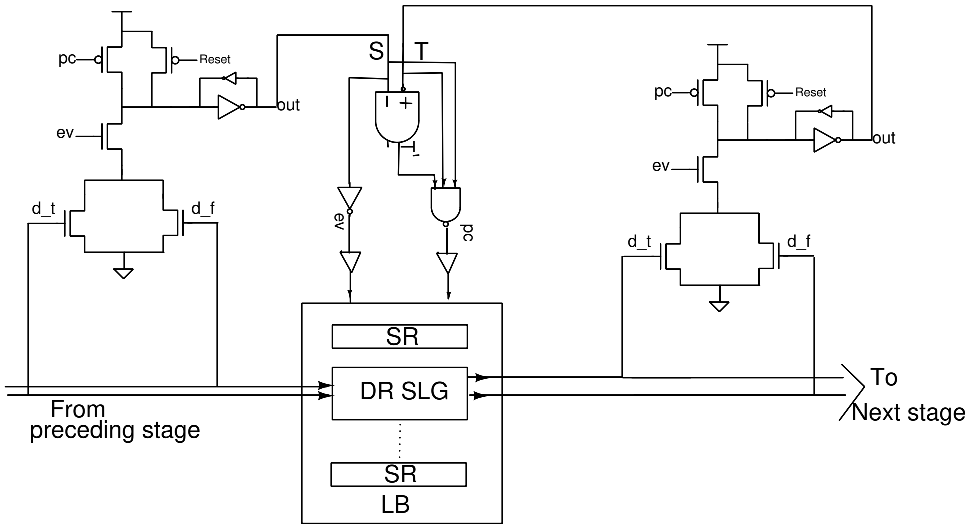


Figure 3.11: Block diagram for a single stage.

3.2.3 Protocol of Early Acknowledged Hybrid Logic Pipeline

The working principle of the EA-Hybrid pipeline can be understood by observing how a data flows through the initial empty pipeline structure shown in the Figure 3.1. The sequence of events (from one evaluation to the next) for stage-N after initialization are as follows:

(i) A logic block in stage-N starts evaluation when the valid data is present at its input. At the same time, the completion detector of stage-N predicts its corresponding logic block evaluation and sets its output 'S' to 1, which allows the logic block of stage-N into isolate phase by forcing the 'ev' control signal to 0 in the next step.

(ii) The logic block in stage-(N+1) starts evaluation after the logic block in stage-N completes its evaluation. At the same time, the stage-(N+1)s completion detector anticipates its corresponding logic block evaluation and allows it into the isolate phase

and the preceding stage logic block (i.e., stage-N) to precharge in the succeeding step (i.e., the CD of stage-(N+1) send T=1 to the SC of stage-N. So by taking (S,T) as (1,1) the SC of stage-N allows its logic block into precharge phase by resetting both the control signals 'pc' and 'ev' to zero).
(iii) As soon as the logic block in stage-N receives permission to precharge from stage-(N+1), it will start by itself the next cycle: precharge followed by evaluate and isolate. (i.e., when (pc,ev) is set to (0,0), in the next step the CD of stage-N makes it output 'S' as '0'. Now irrespective of 'T' from stage-(N+1), the stage controller of stage-N sets (pc,ev) to (1,1) and allows the logic block of stage-N to evaluate the new data item and from here again the cycle repeats from step-(i) to step-(iii)).
The above order of events decide the cycle time (time from one evaluation to the next evaluation) of the pipeline and is calculated using equation 3.1.

$$T_{cycle} = t_{ev.N} + t_{CD.N+1\uparrow} + t_{NAND3.N\downarrow} + t_{CD.N\downarrow} + t_{NAND3.N\uparrow} + t_{buf.N.pc\uparrow} \tag{3.1}$$

Where $t_{ev.N}$ is evaluation time of logic block in stage-N. $t_{CD.N}, t_{CD.N+1}$ are delays of completion detectors in stage-N, and stage-(N+1), respectively. $t_{NAND3.N}$ is delay of the NAND gate present in the stage controller of stage-N. $t_{buf.N.pc}$ is the delay of the stage-N buffer present in the 'pc' signal path. The $\downarrow$ and $\uparrow$ represent the fall and rise delays of the corresponding elements respectively. By assuming that all the stages are identical in terms of delay, equation 3.1 can be simplified as in equation 3.2.

$$T_{cycle} = t_{ev} + 2.t_{CD} + 2.t_{NAND3} + t_{buf.pc} \tag{3.2}$$

3.2.3.1 Timing Constraints

Since a stage-N in the proposed EA-Hybrid pipeline communicates with neighboring stages using EA protocol, certain timing constraints should be met for the proper functionality of pipeline. The following constraints are derived by assuming that all the stages are identical in terms of delay.
1. Evaluation time: A stage-N should complete its evaluation before its control inputs force it into isolate phase. i.e., the evaluation time of a stage should be less than sum of the delays of completion detector, stage controller-inverter and the buffer in control path.

$$i.e \qquad t_{ev} < t_{CD}\uparrow + t_{inv}\downarrow + t_{buf.ev}\downarrow \tag{3.3}$$

Here $t_{inv}\downarrow$ is the delay of inverter present in SC and $t_{buf.ev}\downarrow$ is the delay of buffer in evaluate signal path.
A stage-N should also complete its evaluation before its inputs are removed by the preceding stage-(N-1). For this purpose

$$t_{ev.N} < t_{CD.N}\uparrow + t_{NAND3.N-1}\downarrow + t_{buf.N-1.pc}\downarrow + t_{pc.N-1} \tag{3.4}$$

$t_{pc.N-1}$ is the precharge time of stage-(N-1). Since it is assumed that all the stages are identical the equation 3.4 simplifies to 3.5

$$t_{ev} < t_{CD}\uparrow + t_{NAND3}\downarrow + t_{buf.pc}\downarrow + t_{pc} \tag{3.5}$$

2. Precharge time: A stage-N should finish its precharge before its control inputs force it into evaluate phase.

$$i.e \qquad t_{buf.pc}\downarrow + t_{pc} < t_{CD}\downarrow + t_{NAND3}\uparrow + t_{buf.pc}\uparrow \tag{3.6}$$

In practice the proposed EA-Hybrid pipeline style can easily meet these timing constraints, since it is pipelined to the depth of gate level.

3.2.4 Protocol of Hybrid Logic Pipeline With Post Detection

Unlike the EA-Hybrid pipeline style, the completion detectors in this pipeline style performs the post detection of logic blocks as shown in Figure 3.2. Due to this a completion detector in a stage produces acknowledgement to its previous stage only after the evaluation or precharge completion of its corresponding stage. The sequence of events, from one evaluation to the next, for a stage-N are as follows:
(i) A logic block in stage-N evaluates, as soon as the valid data arrive at its input.
(ii) The completion detector of stage-N detects the evaluation completion of its corresponding logic block and allows it into isolate phase. At the same time the logic block in stage-(N+1) starts evaluation.
(iii) The completion detector of stage-(N+1) identifies the evaluation completion of its

corresponding logic block and allows it into isolate phase and the preceding stage logic block(i.e stage-N) to precharge through the 3-input NAND gate of stage controller.
(iv) The logic block in stage-N precharges.
(v) The completion detector of stage-N detects the precharge completion of its corresponding logic block and allows it to start the evaluation of new data item through its stage controller.
From here the entire cycle from step-(i) to step-(iv) repeats. Hence the cycle time involves the evaluation time of two logic blocks, delay of two completion detectors, precharge time of one stage and delay of 3-input NAND gates present in the stage controller. It is calculated according to the equation 3.7.

$$\begin{aligned} T_{cycle} = t_{ev.N} + t_{ev^*.N+1} + t_{NAND2.N+1} + t_{NAND3.N} + \\ t_{buf.pc.N} + t_{pc^*.N} + t_{NAND2.N} + t_{NAND3.N} + t_{buf.pc.N} \end{aligned} \tag{3.7}$$

Here t_{ev^*}, t_{pc^*} are evaluation and precharge times of a critical path gate in a stage respectively, till the output nodes x and $\overline{x}$. The delay of the static inverters present in the domino gates is not included in these delays. In general, we can compute t_{ev} as $t_{ev^*} + t_{inverter}$, where $t_{inverter}$ is the delay of the static inverter present at the output node of critical path domino gate. As the completion detector collects the inputs from the nodes x and $\overline{x}$, the t_{ev^*}, t_{pc^*} come into picture rather than t_{ev}, t_{pc}. Assuming all the stages are identical the equation 3.7 can be modified to equation 3.8.

$$T_{cycle} = t_{ev} + t_{ev^*} + 2 * t_{NAND2} + 2 * t_{NAND3} + t_{pc^*} + 2 * t_{buf.pc} \tag{3.8}$$

Since in PD-Hybrid pipeline, a completion detector generates acknowledge signals to its preceding stage, after its corresponding stage completes precharge or evaluation, there are no stringent timing constraints in this pipeline style. Hence this method is robust compared to the EA-hybrid but with a nominal drop in the throughput. However, both the proposed pipeline styles are suitable for high throughput applications. If a high throughput application has to deal with highly variable data path delays, then in such case the PD-Hybrid pipeline will be the best option.

3.2.5 Initialization

The proposed pipeline structures can be initialized by simply pulling the Reset control input of all gates to zero. This forces all the outputs of logic blocks and the completion detectors to zero. By taking the (S,T) input as (0,0) all stage controllers set the control signals (pc,ev) to (1,1) and this result enables all the stages to be ready for evaluation during initialization period. Initialization is released by setting Reset input to logic one. The pipeline starts working immediately after the release of initialization.

3.3 Simulation Results

In this work, the proposed pipeline styles are validated by designing different test circuits like FIFO, adder, multiplier and FIR filter. To optimize the proposed circuits, all the transistors in each logic gate are sized according to the method of logical effort Sutherland *et al.* (1999). Further, the critical paths, that decide the cycle time are optimized for minimum delay in all the circuits. For this, the capacitive load at each node in the critical paths is analyzed carefully to arrive at the best transistor sizes. All the circuits are simulated using CADENCE tool set. The section-3.3.1 discusses the design of FIFO, and the chapters 4 and 5 discuss about the other test circuits.

3.3.1 FIFO

A FIFO is a type of buffer, and does not process any data. Many electronic systems require high-speed FIFOs to interface the components that function at different speeds. This section validates the proposed pipeline styles by designing a high-speed FIFO. Moreover, it compares the performance of proposed pipeline styles with the existing domino logic pipelines.

A 4-bit,10-stage FIFO is designed based on proposed EA-Hybrid pipeline style and simulated using 180 nm technology. The Figure3.12 shows the simulation result of stage-2 for a single bit. This sample waveform shows the correctness of data flow between stages. The (pc , ev) are the control inputs, and the (in, inbar) are the data inputs of stage-2. During initialization period (i.e., when Reset=0), the (pc , ev)=(1 , 1) and stage-2 is ready for evaluation. After the release of Reset, the valid data (i.e.,(in , inbar)=(1 , 0)) has arrived at stage-2, and it enters into evaluate phase. At this time the stage controller of stage-2 forces ' ev ' to 0 and stage-2 enters into isolate phase ((pc , ev)=(1 , 0)). After stage-3 completes the evaluation of its data, it sends an acknowledgment to stage-2 (i.e., T=1), and then the stage controller of stage-2 forces 'pc' to 0 which allows stage-2 to precharge phase ((pc , ev)=(0 , 0)),

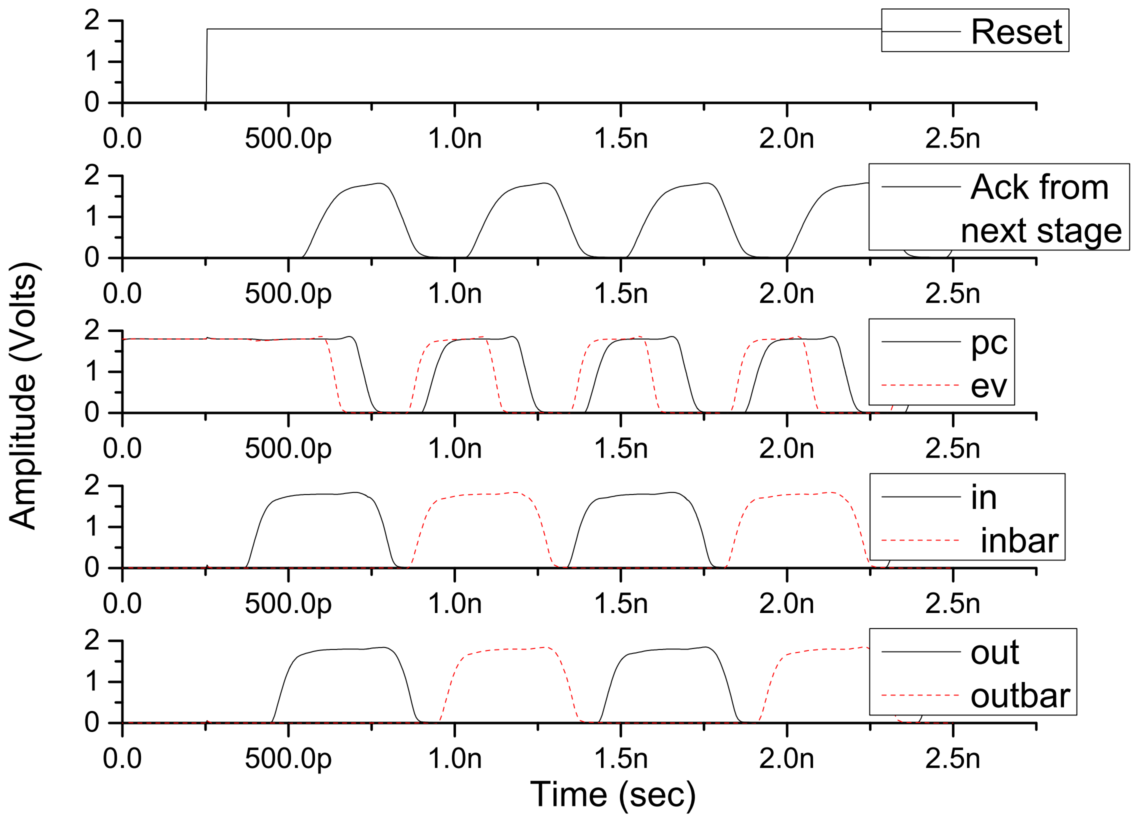


Figure 3.12: Sample input and output waveforms for stage-2 in FIFO.

and the cycle repeats.

The FIFO has been designed using six different pipeline styles: HC-DR, HC-SR, LP2/2-DR, LP2/2-SR, and the proposed pipeline styles EA-Hybrid and PD-Hybrid. All these structures are simulated using UMC 180 nm technology and the results are tabulated in Table 3.3. All these results are in good agreement with the results reported in (Singh and Nowick (2007*b*)),(Singh and Nowick (2007*a*)) , which shows the validity of simulation procedure carried out in this work. From Table 3.3 it can be observed that the proposed EA-Hybrid method has a high throughput of 2.2 Giga-items/sec, and moderate transistor count as compared to other existing pipeline styles. The second proposed method PD-Hybrid has obtained a throughput of 1.68 Giga-items/sec, which is higher than DR pipelines. Though the second proposed method has lower throughput than HC-SR, it is more robust than SR. As the proposed pipeline styles contain DR gates only in the critical path, they have lower transistor count compared to DR pipeline methods and a little high transistor count compared to SR pipeline styles.

Table 3.3: Simulation results of FIFO in 180 nm technology.

Pipeline method/ Parameter	DR		SR		Hybrid	
	LP2/2	HC	LP2/2	HC	EA	PD
Cycle time (ps)	909	649	625	552	454	595
Throughput (k) (Giga-items/sec)	1.1	1.54	1.60	1.81	2.2	1.68
Transistor Count	804	902	440	490	562	554

The FIFO based on proposed EA-Hybrid pipeline style has been laid out using UMC-180 nm technology and is shown in the Figure 3.13. The throughput obtained from the post-layout simulation is tabulated in Table 3.4 and compared with the published results of other pipeline styles. The post-layout results also show that the proposed pipeline method has high throughput as compared to the existing pipeline methods.

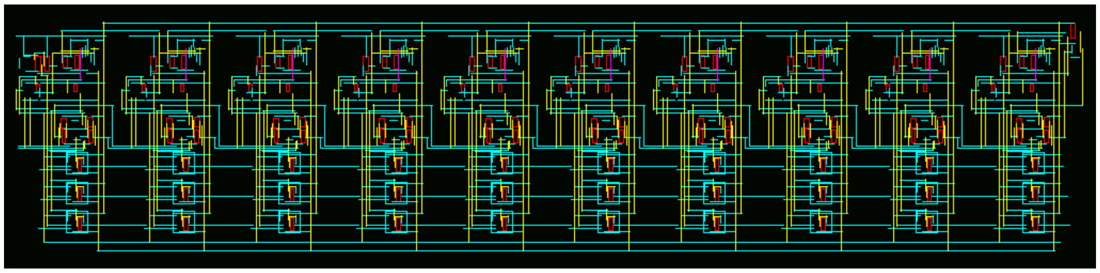


Figure 3.13: Layout of 4-bit,10-stage FIFO.

The cycle time/throughput of an asynchronous pipelined circuit is mainly decided

Table 3.4: Throughput of different pipeline designs for FIFO test case.

Pipeline design	Technology	cycle time (T)	Throughput (k)
	nm	Analytical formula	Giga-items/sec
PS0 DR Singh and Nowick (2007*b*)	180	$3.t_{ev} + 2.t_{CD} + t_{pc}$ (1.9 ns)	0.51**
LP2/2 DR Singh and Nowick (2007*b*)	180	$2.t_{ev} + 2.t_{CD}$(1.1 ns)	0.9**
LP2/2 SR Singh and Nowick (2007*b*)	180	$2.t_{ev} + 2.t_{ac}$(0.763 ns)	1.31**
HC SR Singh and Nowick (2007*a*)	180	$t_{ev} + t_{pc} + t_{ac} + t_{nand3} + t_{inv}$	1.75** (0.571 ns)
SP DR Midhun *et al.* (2014)	90	$2.t_{ev} + t_{CD} + t_{ac}$ (0.358 ns)	2.79**
APCDP Xia *et al.* (2012*a*)	65	$3.t_{ev} + 2.t_{CD^\star} + t_{pc}$(0.16 ns)	6.17*
EA Hybrid logic	180	$t_{ev} + 2.t_{CD^\star} + 2.t_{nand3} + t_{buf.pc}$ (0.5 ns)	2**
	90	(0.245 ns)	4.08*
PD Hybrid logic	180	$t_{ev} + t_{ev^*} + 2 * t_{NAND2} +$	1.68*
		$2 * t_{NAND3} + t_{pc^*} + 2 * t_{buf.pc}$ (0.595 ns)	

** denotes post-layout results and * denotes pre-layout results.
$t_{CD^\star}$: delay of CD that detects only the critical path.
$t_{pc^\star}$ & $t_{ev^\star}$: Precharge & evaluation times of critical domino gate without considering the inverter delays present in domino gate.

by the evaluation (t_{ev}) time of the logic blocks. However, depending on the pipeline structure, the cycle time also depends on delays involved in the precharge / completion detection mechanisms. From Table 3.4, it can be observed that, except HC-SR and the proposed EA-Hybrid designs, all the designs have 2 or 3 evaluation times. Hence the proposed EA-Hybrid and HC-SR designs have higher throughput as compared to the other pipeline designs. Though HC-SR and proposed EA-Hybrid designs have almost similar terms in their cycle time, the EA-Hybrid is faster than HC-SR. The delay of asymmetric C-element (ac-element) t_{ac} in HC-SR is higher than the delay of completion detector in the EA-Hybrid design. This reduction in delay of the EA-Hybrid can be attributed to the less loading effect on completion detector as compared to the ac-element in the HC-SR pipeline. Hence the proposed EA-Hybrid has a throughput of 2 Giga-items/sec which is 14.29 % $(((k_{EA-Hybrid} - k_{HC-SR}) \div k_{HC-SR}) * 100)$ more than the HC-SR pipeline in 180 nm technology. Even with approximately two evaluation times involved in cycle time, the PD hybrid has a comparable throughput as that of HC-SR. This is due to the simplified completion detectors of PD hybrid.

Further, the robustness of the proposed designs is very high due to the detection of data path instead of having a separate control path for acknowledge generation. The simulation of EA-Hybrid logic FIFO is also carried out in 90 nm technology and the result is tabulated in Table 3.4. The throughput of FIFO is 4 Giga-items/sec which is almost double as compared to the throughput obtained in 180 nm technology. In the 90 nm technology the EA-Hybrid has achieved 46.2% higher throughput than the SP-DR pipeline style. Further improvement in throughput can be obtained at lower technology nodes.

3.4 Summary

This chapter has suggested two efficient asynchronous domino logic pipeline styles for high throughput applications. The proposed structures are realized based on the hybrid data paths. The use of hybrid data path has reduced the overhead in handshake control logic, which in turn has improved the throughput of the proposed structures. The presence of the isolate phase has improved the storage capacity of both the pipelines up to 100 %. Furthermore, the adoption of early acknowledge handshaking in EA-Hybrid has greatly improved the throughput of this pipeline style. Both the proposed pipelines are validated by designing and simulating a 4-bit,10-stage FIFO. The FIFO circuit based on EA-Hybrid has obtained higher throughput compared to all other FIFO circuits. The PD-Hybrid FIFO has obtained throughput comparable to the HC-SR and higher than all other FIFO circuits. Hence, the proposed pipeline styles are suitable for the gate-level pipelined systems that require high throughput, buffering capacity, and robustness.

Chapter 4

DESIGN OF HIGH SPEED COMPUTATIONAL UNITS

4.1 Introduction

The adder and the multiplier units are the commonly used computational units in many digital systems. This chapter addresses the design of high-speed ripple carry adder and array multiplier units based on the proposed pipeline methods. Thus the suitability of proposed pipeline structures for data processing circuits is validated in this chapter. Though several adder and multiplier architectures can minimize the computation time, the throughput of a fully pipelined system could not be improved by these structures. The regular structure and simple pipeline-stage partitioning features let the ripple carry adder, and the array multipliers more suitable for fully pipelined systems (Lu and Samueli (1993)).

4.2 Ripple Carry Adder

A ripple carry adder (RCA) is a digital circuit that performs the addition of two n-bit numbers. In an n-bit RCA, n number of full adders are cascaded such that the carry-in to each full adder comes from the carry-out of its preceding full adder. As each carry bit gets rippled into the next stage, this adder is called as the ripple carry adder. Figure 4.1 shows the block diagram of a 4-bit RCA. Though the structure of this adder is quite simple, the delay of this adder is very high due to the rippling of the carry from the least significant stage to the most significant stage. This delay increases with an increase in data size.

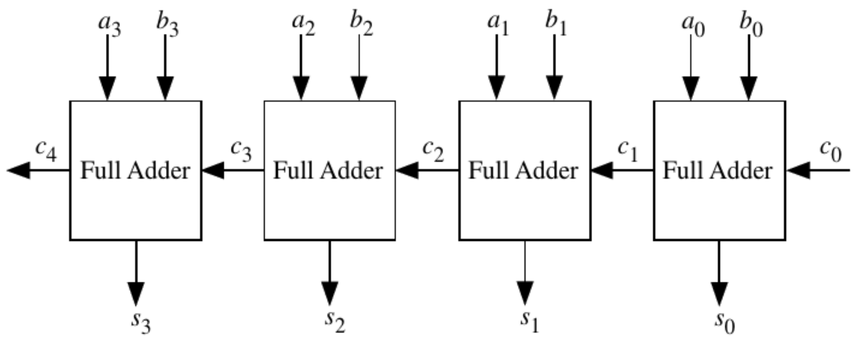


Figure 4.1: 4-bit ripple carry adder(Rabaey *et al.* (2002))

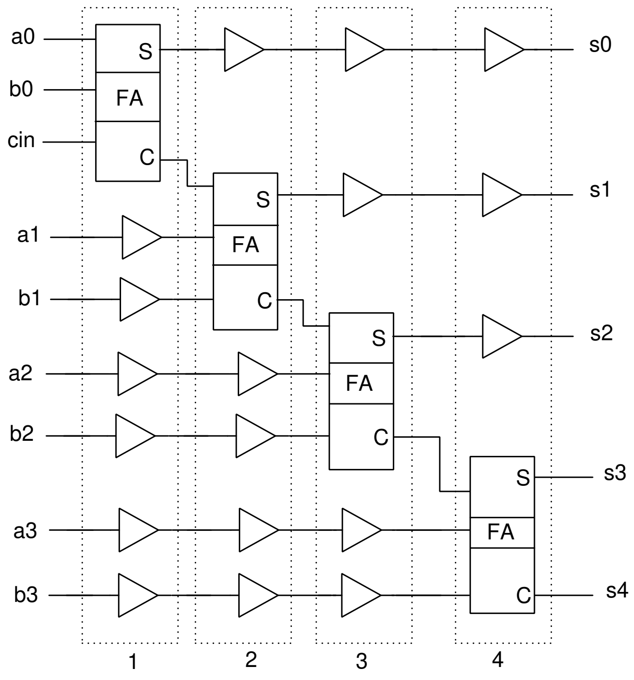


Figure 4.2: Data path of fine grain pipelined ripple carry adder(Xia *et al.* (2015))

Pipelining is an efficient technique to improve the speed of the RCA. Figure 4.2 shows the data path of a 4-bit gate-level pipelined RCA. Each stage contains a full adder and buffer units. The first stage sums the least significant bits (a0,b0) and initial carry. The second stage adds the next higher significant bits (a1,b1) and the carry-out from the first stage. This process continues till the carry reaches to the final full adder that adds the most significant bits. A similar design procedure can be extended to higher data widths. The full adder sum generator is in the non-critical path. Hence in the adder circuits designed based on proposed styles, full adder sum circuit is designed using single rail domino logic. The full adder carry generator is used as a critical element. Hence it is designed as DR-SLG, which provides a constant delay for all the applied data patterns. The design of full adder sum and SLG-carry gates are shown in Figure 4.3.

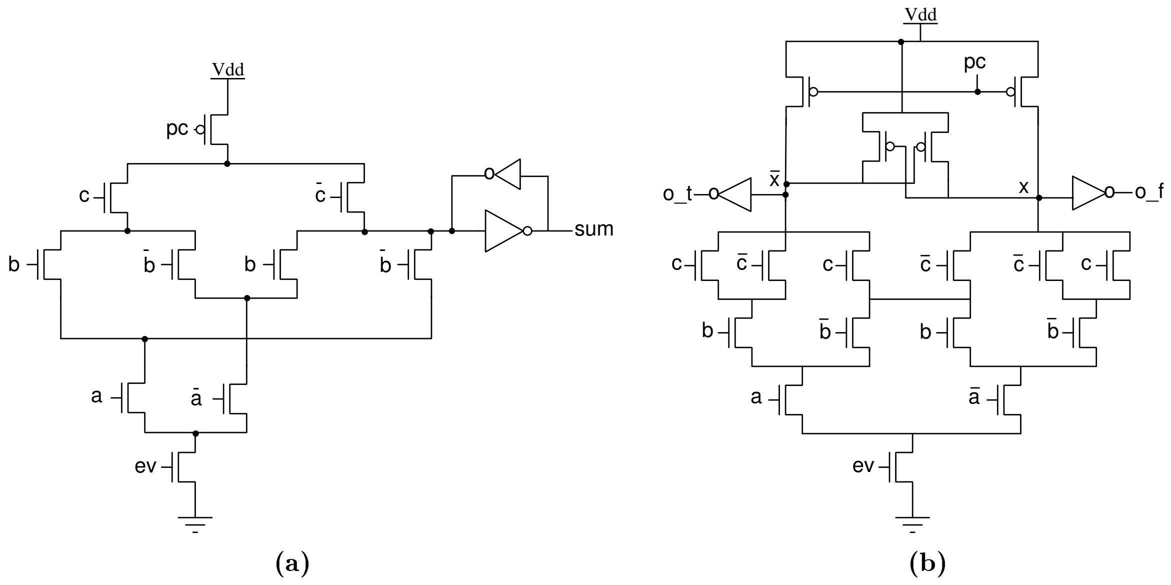


Figure 4.3: (a) FA-sum SR gate (b) FA-carry-SLG.

A 16-bit ripple carry adder is designed in 90 nm technology using four different pipeline styles: HC-SR, APCDP, and proposed EA-hybrid, PD-hybrid logic styles. Figures 4.4 and 4.5 show the sample input and output of EA-Hybrid and PD-Hybrid 16-bit adders, respectively for 4 input cycles. From these figures, valid output can be observed after the initial latency. It can also be observed that a spacer follows every valid output. In addition to this correctness of data flow can also be seen from Figures 4.4 and 4.5.

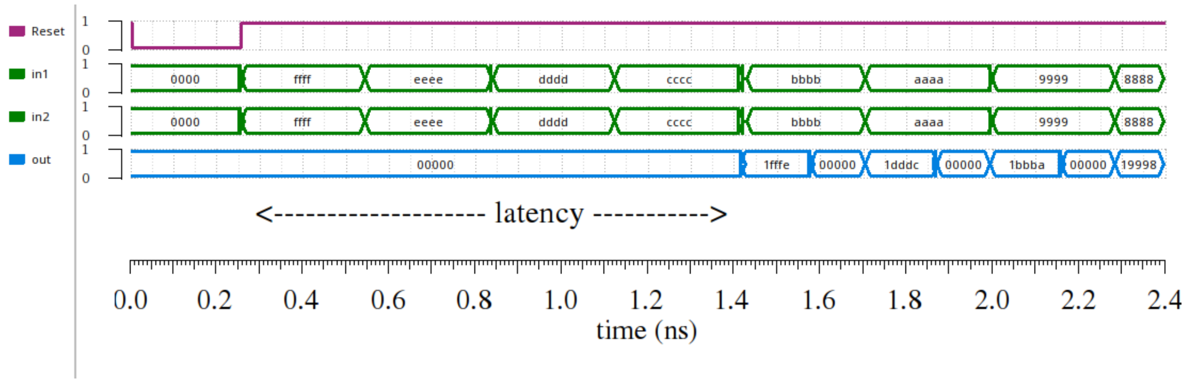


Figure 4.4: Screen-shot of 16-bit EA-Hybrid adder output.

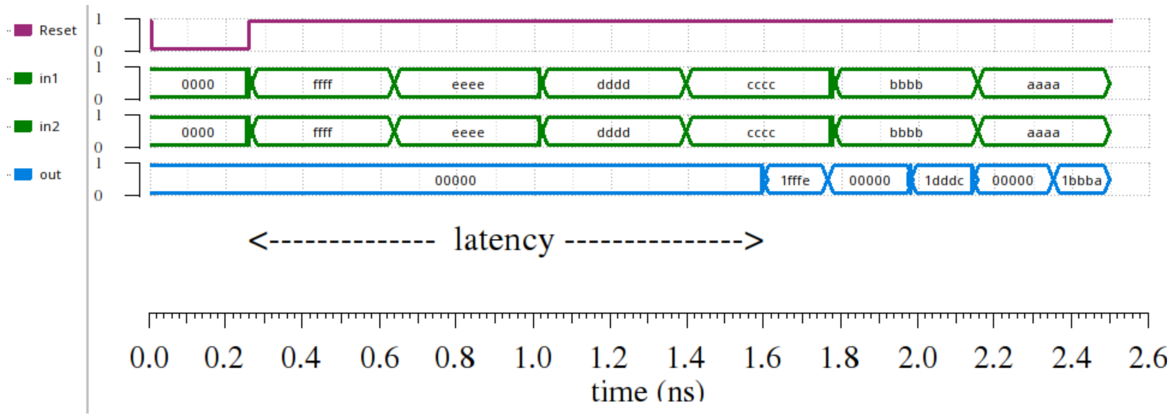


Figure 4.5: Screen-shot of 16-bit PD-Hybrid adder output.

Table 4.1 compares the performance metrics of all the designed adders. The motivation behind choosing HC-SR and APCDP pipelines is their comparable throughputs with the proposed pipeline methods for FIFO test case. The adder designed using the proposed EA-Hybrid pipeline method has obtained a throughput of 3.44 Giga-items/sec, which is 13.53% higher than HC-SR and 79.16% higher than APCDP. The adder based on the PD-Hybrid pipeline style has achieved 36.9% higher throughput than APCDP. The transistor count and power consumption of proposed methods are a bit higher than the APCDP and comparable with the HC-SR. The reason for this is the use of additional elements like stage controllers and SLG's, to achieve high throughput. Energy-delay^2 product $(E.T^2)$ is an excellent metric for comparing energy and delay efficiency. There will always be a trade-off between power consumption and speed. Hence, to compare the performance of different circuits, usually, the met-

Table 4.1: Evaluation results of 16-bit adder (90 nm).

Pipeline design	Transistor count	latency	Throughput	Power consumption	Energy-delay^2 product(ET^2)
		ns	Giga-items/sec	mW	$(pJ * ns * ns)$
APCDP	3694	1.12	1.92	5.8	0.811
HC SR	4004	1.05	3.03	13.4	0.482
EA hybrid logic	4025	1.08	3.44	13.53	0.33
PD hybrid logic	3970	1.1	2.63	11.91	0.653
Synchronous	11458	1.2	2.5	18.7	1.19

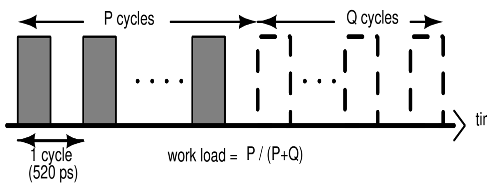
ric Energy-delay square product ($E.T^2$), which includes both power and delay metrics, is used. The lower the $E.T^2$ value, the better the circuit performance Martin *et al.* (2002). From Table 4.1 it can be observed that the $E.T^2$ of the proposed EA-hybrid pipeline is minimum as compared to other pipelines. Though the $E.T^2$ of the HC-SR and the proposed PD-hybrid pipeline seems to be comparable, the robustness of the PD-hybrid logic is high. This is because, in SR designs, critical path delays are replicated in the request signal path, whereas in both the proposed styles, the control signals are generated from the critical path itself. The latency of all the pipeline styles is nearly equal, as their data paths are almost similar.

Further, to examine whether the timing constraints mentioned in Chapter-3 are met or not, the delay of each component is measured for 16-bit, EA-Hybrid adder and listed in Table 4.2. From these delays it can be observed that the timing constraints of EA-Hybrid method are satisfied with sufficient margins. The first constraint mentioned in equation (3.3) is satisfied with a margin of 167.12 %, the second constraint shown in equation (3.5) is met with a margin of 533.26 % and the last constraint given in equation (3.6) is met with a margin of 33.31 %.

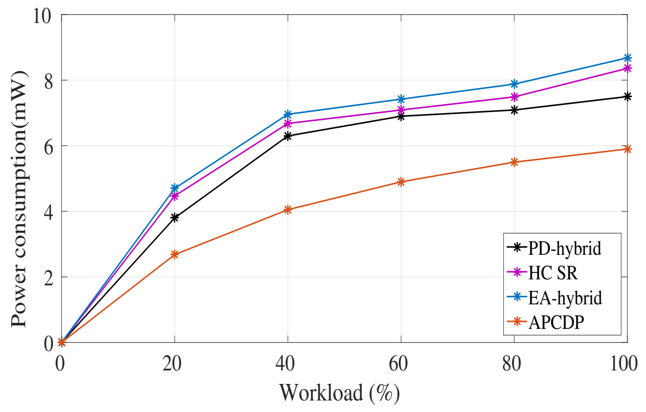
The workload is a parameter used to evaluate the power consumption of pipelined circuits Xia *et al.* (2015). It is defined as the ratio between the number of active state cycles and the total number of cycles. Figure 4.6a conceptually defines the workload. It is measured as P/(P+Q), where P is the number of successive data injection cycles (active cycles) and Q is the number of successive empty cycles. Empty cycles

Table 4.2: Delay of individual components in 16-bit, EA-Hybrid adder

EA-Hybrid pipeline
t_{ev}: 49.71 ps
$t_{inv\uparrow}$:11.1 ps
$t_{inv\downarrow}$:9.35 ps
$t_{nand3\downarrow}$:22.71 ps
$t_{nand3\uparrow}$:42 ps
$t_{CD\uparrow}$:50.2 ps
$t_{CD\downarrow}$:90.8 ps
t_{pc}:92.2 ps
$t_{buf.pc\downarrow}$:44.7 ps
$t_{buf.pc\uparrow}$:49.71 ps
$t_{buf.ev\downarrow}$:28.95 ps
$t_{buf.ev\uparrow}$:21.5 ps



(a)



(b)

Figure 4.6: (a) Definition of workload Xia *et al.* (2015). (b) Power consumption at different workloads.

mean we keep the stages such that they do not process any data. Figure 4.6b shows the power consumption of the pipeline designs at different workloads when all the designs are operating at 1.9 Giga-items/sec. Compared to APCDP, the proposed pipeline styles are consuming high power due to the use of additional blocks like SC, CD, and SLGs. However, the cycle time and the ET^2 of the proposed design are quite low as compared to APCDP. Though the proposed designs have DR critical paths and completion detectors, their power consumption is still comparable with the HC-SR pipeline.

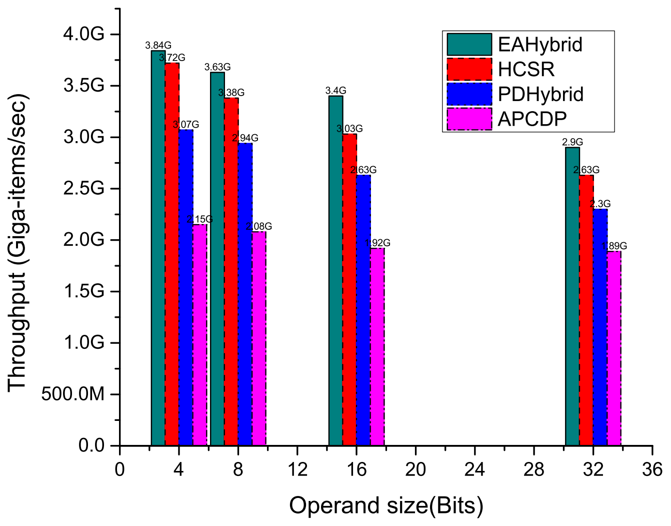


Figure 4.7: Throughput vs. operand size.

In order to test the sensitivity of the throughput with data width the adders with different operand sizes are designed, and the simulation results are presented using bar charts in the Figure 4.7. Ideally, the throughput of SR and hybrid data path designs should be independent of the data width because in their cycle time equations the terms t_{ev} and t_{cd} do not vary with operand size. However, in practice, as the data width increases, the pipeline stage extends vertically, and this boosts the loading effect on the stage driving buffers, which in turn increases the delay of buffers. Hence as the data width increased from 4-bit to 32-bit, the throughput of EA-hybrid, HC-SR, PD-hybrid, and APCDP are dropped by 24.4%, 29.3%, 25.08%, and 12.09%, respectively. In DR designs this drop rate will be very high because as the data width increases, the delay of completion detectors in these circuits increases by a significant value.

A 4-bit adder based on the proposed EA-Hybrid pipeline style has been laid out using UMC-180 nm technology, and is shown in Figure 4.8. The throughput obtained from the post-layout simulation is 1.23 Giga-items/sec, which is just 4% less than the performance obtained from schematic simulation. As there are no standard cells available for DR-SLGs, all the test cases in this work are full custom designs.

In the proposed PD-Hybrid pipeline style shown in Figure 3.2, the completion detectors collect the inputs from the intermediate nodes x and $\overline{x}$ of DR-SLG's. Though

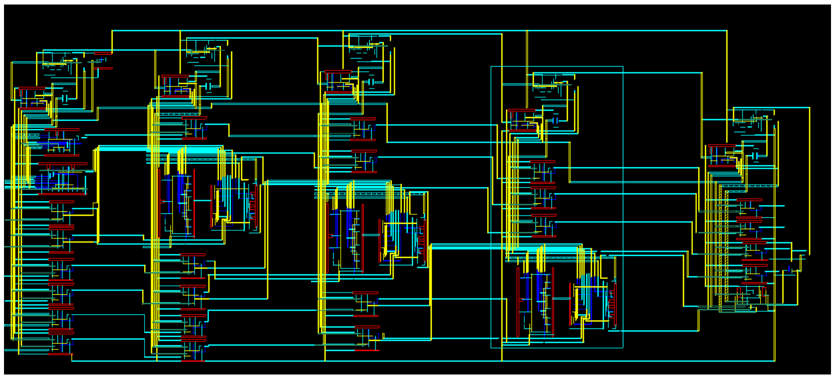
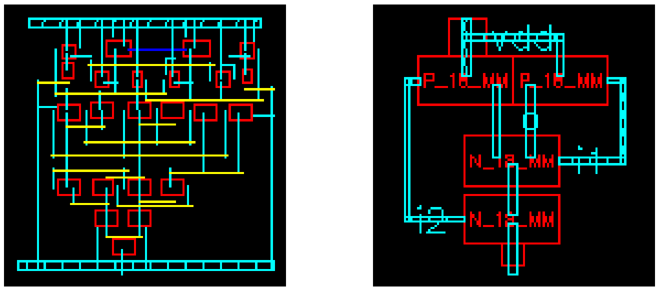


Figure 4.8: Layout of 4-bit adder.



(a) (b)

Figure 4.9: (a) FA Carry circuit layout. (b) Completion detector layout.

the completion detector brings additional load on the intermediate nodes, this loading effect can be neglected as the completion detector's in PD-Hybrid are simple 2-input, static NAND gates. To prove this, the critical element in adder and multiplier circuits (i.e.,Full adder carry generator) and the completion detector used in the PD-Hybrid pipeline style are laid out in 180 nm technology, and shown in Figure 4.9. The inputs for the completion detector are given from the intermediate nodes x, and $\overline{x}$ of full adder carry gate. The post layout simulation is carried out with R-C-CC extraction. It is observed that the functionality of the Full adder is not disturbed due to completion detector loading even in post-layout simulation. The delay of the signals at the intermediate nodes x and $\overline{x}$ with respect to control signals 'pc' and 'ev' , is measured in both schematic simulation and layout simulation and given in Table 4.3. From these results, it can be observed that the impact of completion detector loading on intermediate nodes is quite negligible. To ensure the proper functionality of full adder carry generator even with further delay variations, it is also simulated by connecting a capacitive load, whose value is five times that of completion detector gate capacitance, at the nodes x and $\overline{x}$. Even at this load, the FA carry circuit is functioning properly without much degradation in its speed.

Table 4.3: Delay of signals at nodes x and $\overline{x}$

Node name	x	$\overline{x}$
Delay from Schematic simulation	118.2 ps	123.7 ps
Delay from Post layout simulation (With CD load)	120 ps	129.7 ps
Delay from Post layout simulation With a capacitive load equal to 5 times of CD gate capacitance)	124.6 ps	136 ps

4.3 Array Multiplier

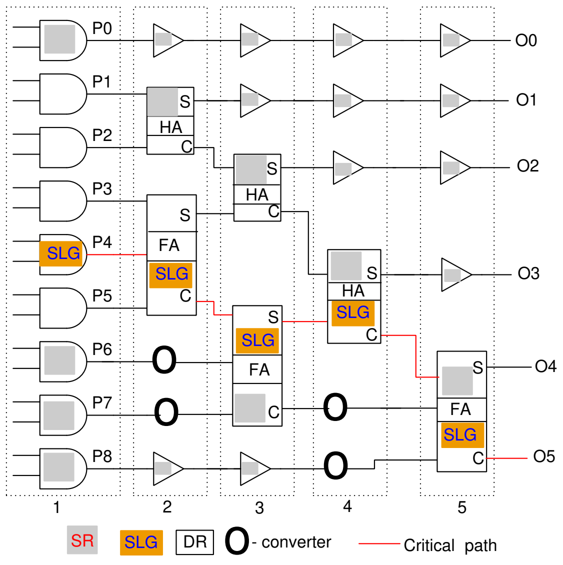


Figure 4.10: (a) Data path of 3x3 array multiplier

The Figure 4.10 shows the data path construction of a 3x3 array multiplier with hybrid data paths. The first stage in the array multiplier generates partial products using an array of domino AND gates. The succeeding stages sum the partial products using half adders and full adders. In the 5th stage FA-carry circuit is selected as critical element and converted to SLG. In the 4th stage the HA-Carry circuit is considered as critical element as it has more number of inputs compared to the other buffer gates and it also has a link to its successor stage SLG. In the same fashion, the FA-Sum circuit in the 3rd stage, FA-carry circuit in the 2nd stage and an AND gate in the first stage are considered as critical elements and converted to SLGs. The remaining gates in the stages are constructed as either SR or DR gates depending on the next stage requirements. Almost all non-critical paths are constructed using SR gates. If a gate output in stage-N has to be utilized by a FA in stage-(N+1), then the gate in stage-N is constructed as DR gate, since the FA need inputs in both complimentary and uncomplimentary forms. The similar design procedure is extended to 8x8 array multipliers.

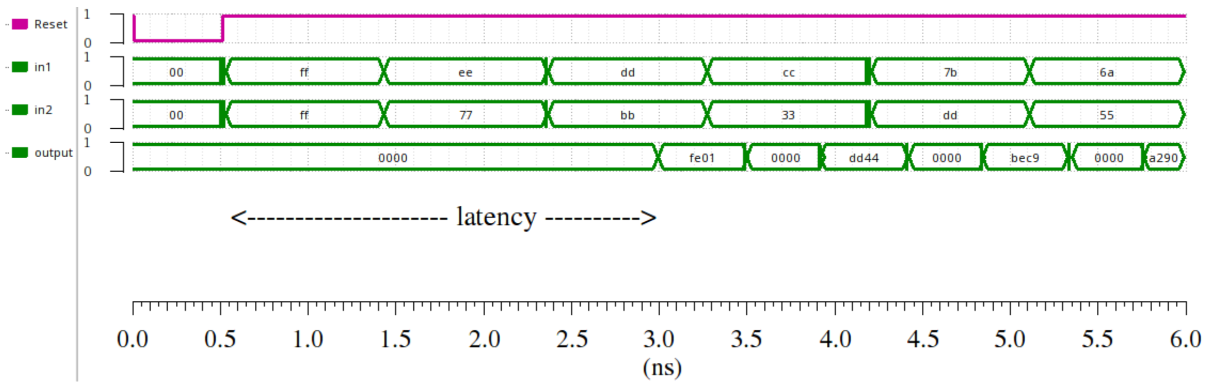


Figure 4.11: Screen-shot of 8X8 EA-Hybrid array multiplier output.

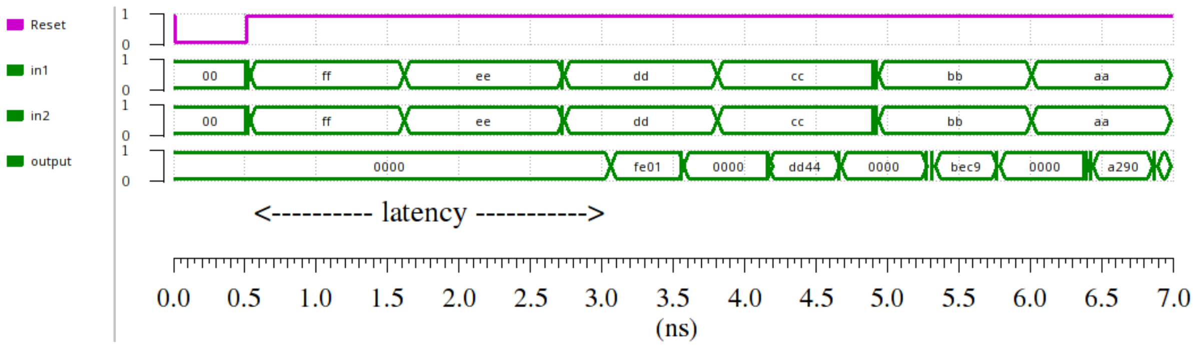


Figure 4.12: Screen-shot of 8X8 PD-Hybrid array multiplier output.

Four 8 x 8 array multipliers are designed using APCDP, HC-SR and proposed pipeline methods. All the multiplier designs are simulated in UMC-180 nm technology at 1.8 V supply. Figures 4.11 and 4.12 show the sample input and output of EA-Hybrid and PD-Hybrid 16-bit multipliers, respectively for 4 input cycles. The correctness of data flow can be observed from these figures.

Table 4.4: Evaluation results of 8x8 array multiplier (180 nm)

Pipeline Style	Transistor count	Latency	Throughput	Power consumption	$E.T^2$
		(ns)	Giga-items/sec	(mW)	$(pJ * ns * ns)$
APCDP Xia *et al.* (2015)	5439	2.49	0.77	19.7	40.91
HC-SR Singh and Nowick (2007*a*)	5741	2.18	0.97	27.9	30
EA-HYBRID	5757	2.28	1.08	31.6	24.6
PD-HYBRID	5673	2.35	0.909	25.4	33.8

The performance metrics of multipliers are summarized in Table 4.4. The simulation results show that the multiplier designed using proposed pipeline method EA-Hybrid has obtained a throughput of 1.08 Giga-items/sec, which is 11.34% higher than HC-SR and 40.25% higher than APCDP. The pipeline style PD-Hybrid has achieved 0.909 Giga-items/sec throughput and it is 18.05% faster than APCDP. This is on account of t_{ev^*} and t_{pc^*} in PD-Hybrid style. Further the CD's in this method are 2-input NAND gates, where as they are 2-input NOR gates in APCDP. Though this design has 6.3% lesser throughput than the HC-SR, it is more robust as it need not deal with any timing constraints. Unlike in SR methods, where critical path delay is replicated as matched delays in the request signal path, both the proposed methods detects the critical path itself. Hence they are highly robust compared to the SR pipeline methods. Further, the overhead in the data path and control path of the proposed designs is quite less compared to DR pipeline styles due to the adoption of hybrid data paths. From Table 4.4, it can be observed that though the EA-Hybrid method has little more transistor count and power consumption compared to HC-SR, it has achieved the lowest $E.T^2$. The proposed PD-Hybrid has obtained 17.37% lower $E.T^2$ than APCDP and has lower transistor count and power consumption than HC-SR. The latency of both the proposed methods is less compared to the APCDP.

In order to see the effect of technology scaling on the proposed methods, and also to

Table 4.5: Simulation results of 8x8 array multiplier at lower technologies

Pipeline Style	Technology nm	Throughput Giga-items/sec
APCDPXia *et al.* (2015)	65	4
EA-Hybrid	90	3.44
	65	5.71
PD-Hybrid	90	2.56
	65	4.44

compare the proposed methods with the published result of APCDPXia *et al.* (2015), the multipliers are also simulated in UMC 90 nm and 65 nm technologies at a supply voltage of 1.2 V. Table 4.5 depicts these results. The proposed methods have achieved the higher throughput compared to APCDP even at lower technology nodes. Here the supply voltage 1.2 V is considered for fair comparison of proposed methods with the published APCDP result. However, the proposed methods can properly function over a range of supply voltages. The variation of throughput as the function of supply voltage at 65 nm technology is shown in the Figure 4.13. As seen from the Figure 4.13, the throughput increases with increase in the supply voltage due to the reduction in the gate delays with enhancement of supply voltage.

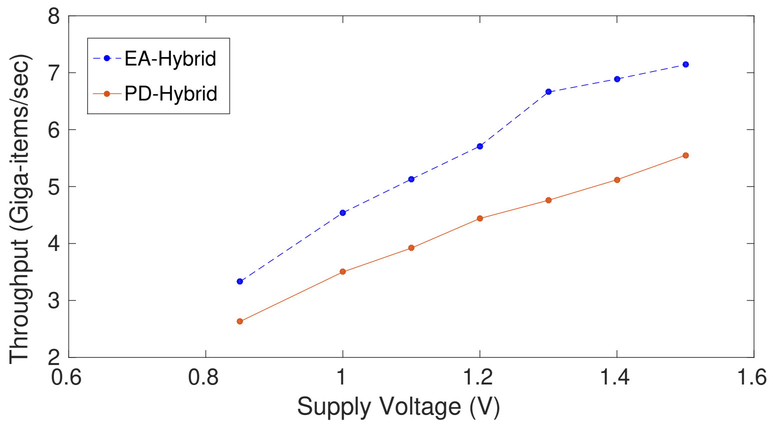


Figure 4.13: Throughput VS Supply voltage.

Further, the delay of each component in the EA-Hybrid multiplier circuit is measured in 65 nm technology, and listed in Table 4.6. From these delays, it can be observed that the timing constraints of the EA-Hybrid method are satisfied with sufficient margins. The first constraint mentioned in equation (3.3) is satisfied with a

Table 4.6: Delay of individual components in 8x8, EA-Hybrid multiplier (65 nm)

EA-Hybrid pipeline
t_{ev}: 22.07 ps
$t_{inv\uparrow}$:6.38 ps
$t_{inv\downarrow}$:14.45 ps
$t_{nand3\downarrow}$:18.8 ps
$t_{nand3\uparrow}$:15.9 ps
$t_{CD\uparrow}$:34.9 ps
$t_{CD\downarrow}$:52.47 ps
t_{pc}:55.2 ps
$t_{buf.pc\downarrow}$:30.51 ps
$t_{buf.pc\uparrow}$:28.3 ps
$t_{buf.ev\downarrow}$:33.68 ps
$t_{buf.ev\uparrow}$:28.06 ps

margin of 214.18 %, the second constraint shown in equation (3.5) is met with a margin of 531.67 % and the last constraint given in equation (3.6) is met with a margin of 12.78 %.

Figure 4.14 shows the power consumption of the proposed pipeline designs at different work loads, when both the designs are operating at a throughput of 0.909 Giga-items/sec. At all the workloads the proposed methods are functioning properly without any leakages in the internal nodes. Compared to EA-Hybrid, the PD-Hybrid pipeline consumes low power due to the simple CDs. However both the proposed designs are suitable for high performance applications.

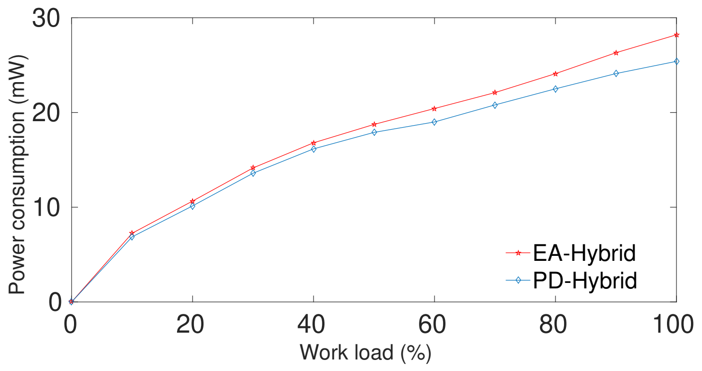


Figure 4.14: Testing of array multiplier at different Work loads

4.4 Summary

In this chapter, the proposed pipeline styles are validated for data processing circuits by designing ripple carry adder and array multiplier. The adder and multipliers designed based on proposed EA-hybrid have higher throughput compared to other existing pipeline styles. Though the circuits based on PD-hybrid have lower throughput than HC-SR, they are more robust due to the adoption of hybrid data paths. The power consumption of the proposed pipeline styles at different workloads is also evaluated. It has been observed that the proposed pipeline styles have comparable power consumption with HC-SR and has higher power consumption than APCDP. However, the ET^2 of the proposed EA-Hybrid pipeline is very low compared to APCDP and HC-SR pipelines. It is also observed that the throughput fall of the proposed pipeline styles with an increase in operand size is very less. Further, the effect of technology scaling and supply voltage on the throughput of proposed methods is also observed.

Chapter 5

HIGH THROUGHPUT DIGITAL FIR FILTER FOR PRML READ CHANNEL APPLICATION

5.1 INTRODUCTION

These days, most of the electronic devices like digital cameras, personal computers, video recorders and MP3 players extensively use the hard disk drives as storage units. To satisfy the ever increasing demand on high data rates and storage capacity requirements, the hard disk drives adopt the sophisticated signal processing methods like Partial Response signaling with Maximum Likelihood detection (PRML) for their read channels. A high throughput, low latency digital FIR filter is the primary requirement of the read channel for equalization process Pearson *et al.* (1995). This chapter presents the design of an asynchronous digital FIR filter suitable for high performance PRML read channel ICs. To achieve the enhancement in speed and reduction in latency parameters of the FIR filter, its computational units are deeply pipelined using high speed EA-hybrid pipeline method.

5.2 Proposed Asynchronous Digital FIR filter

High throughput and low latency are the primary requirements of the FIR filter for the read channel application. The asynchronous domino logic gate-level pipeline styles support high throughput designs. These methods also avoid the latches between the pipeline stages, which leads to low latency designs. Further, the asynchronous circuits by nature can handle variable data rates, and hence the asynchronous FIR filter can tolerate the variations in the data rate. Hence the asynchronous gate-level pipeline styles are quite suitable to design an FIR filter for read channel application. To the best of our knowledge, there is no fully asynchronous gate-level pipelined FIR filter in the literature. As mentioned earlier, asynchronous pipelines have various benefits over the synchronous pipelines. Hence as a part of this research work, an attempt is given to design a fully asynchronous gate-level pipelined FIR filter. As the asynchronous EA-Hybrid pipeline can produce high throughput circuits, this pipeline method is chosen to design the FIR filter.

5.2.1 Specifications of FIR filter

The general data format of the FIR filter for read channel application is shown in the Table 5.1. Hence in this work, an 8-tap FIR filter is designed with the data sizes given in Table5.1. A low pass FIR filter, with pass edge frequency of 50 MHz and stop edge frequency of 500 MHz, based on direct form architecture, is designed.The gate-level pipeline methods are best suitable for simple architectures. Hence the direct form architecture is chosen to design the filter. The filter coefficients are computed using MATLAB for the given specifications. These coefficients are fractional values. As the proposed EA-Hybrid pipeline supports integer data type, the filter coefficients should be rounded to the integer values. The rounded integer coefficients are obtained

Table 5.1: General data format of the FIR filterKi *et al.* (2000)

Input data size	6-bit
Coefficient data size	6-bit
Number of taps	8-tap
Output data size	15-bit

by multiplying the fractional coefficients with 2^5. To see the effect of rounding on filter output, few simulations are carried out using Matlab. An FIR filter with these specifications is designed and simulated in MATLAB, with the original coefficients and the rounded coefficients. The output of the filter is shown in Figure 5.1. From the Figure 5.1 we can observe that the error between the output with original coefficients and the output with rounded coefficients is with in $\pm 0.018\,V$, which is very less. Hence in this work, the FIR filter is designed with rounded integer coefficients.

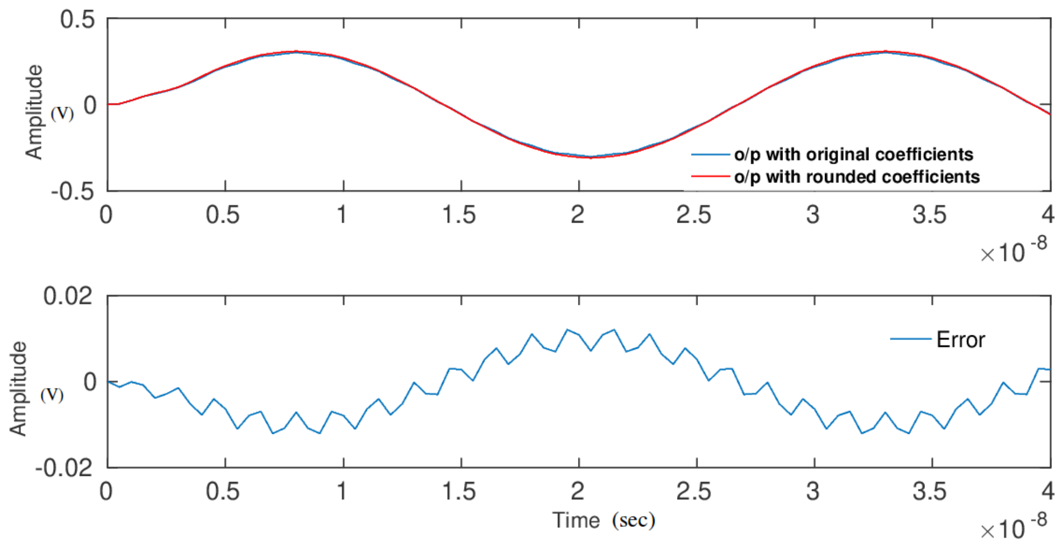


Figure 5.1: Filter output waveforms and Error signal

5.2.2 Filter architecture

Figure 5.2 shows the architecture of the FIR filter. This is a direct form FIR structure, with EA-Hybrid gate-level pipelined multiplier and adder units. The fine grain pipelined multiplier and adder units improve the throughput of the FIR structure by reducing its critical path to a single gate level. There are two significant challenges in the design of this structure:

(i) The adder units are receiving inputs from their preceding multiplier units. Hence simple linear EA-Hybrid pipeline structure is not sufficient to realize the adder units.

(ii) The input data and the data from the registers are flowing in two directions(i.e., to their succeeding registers and the first stages of the multiplier). Hence it is difficult to provide synchronization between registers and the multipliers first stages.

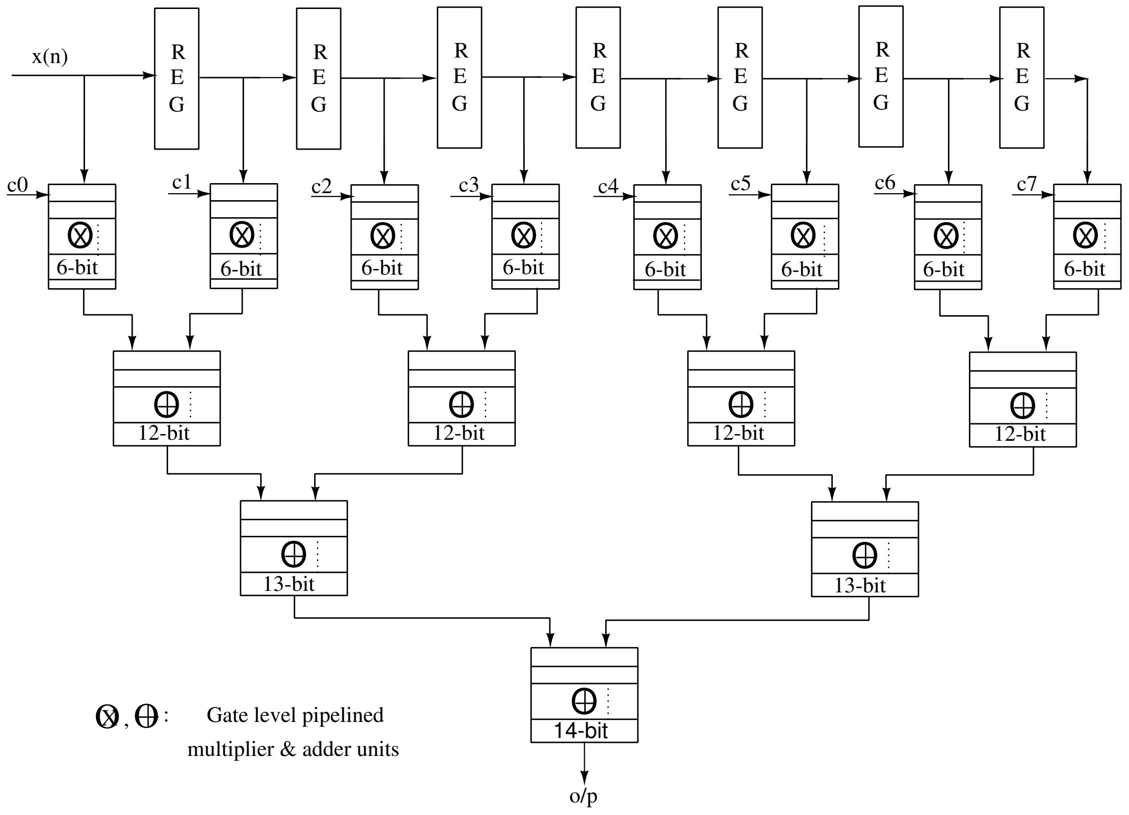


Figure 5.2: Structure of FIR filter data path.

The first challenge can be met by designing adders using EA-Hybrid JOIN structure as shown in Figure 5.3. When extending the EA-Hybrid pipeline to JOIN structure, the completion detector of the first stage in an adder has to detect all the critical paths of its preceding multipliers.

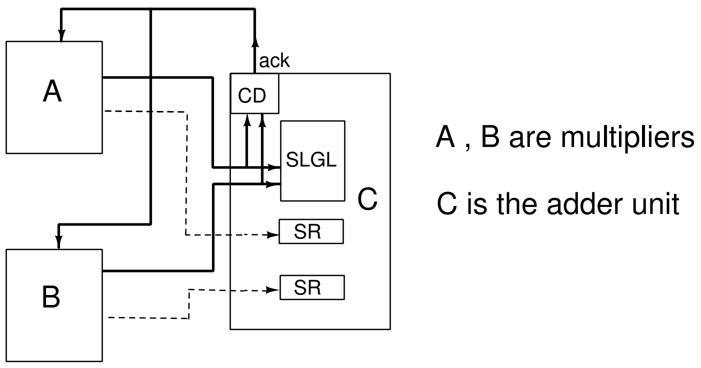
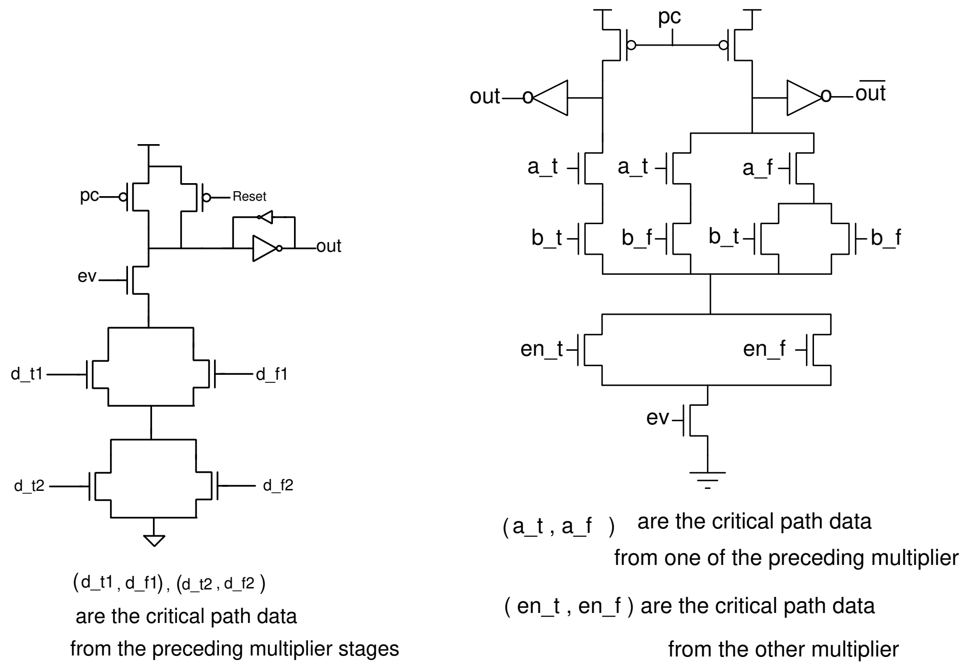


Figure 5.3: Join Structure for EA-Hybrid pipeline.

Each additional critical path can be accommodated by placing an extra NMOS transistor in the pull-down path of completion detector. Further, the SLG/SLGL of adder first stage should also be linked to all the incoming critical data paths. Figure 5.4 shows the Completion detector and the SLG of adder first stage.


pc
Reset
out
ev
d_t1
d_f1
d_t2
d_f2
(d_t1 , d_f1), (d_t2 , d_f2)
are the critical path data
from the preceding multiplier stages
pc
out
$\overline{\text{out}}$
a_t
a_t
a_f
b_t
b_f
b_t
b_f
en_t
en_f
ev
(a_t , a_f) are the critical path data
from one of the preceding multiplier
(en_t , en_f) are the critical path data
from the other multiplier

(a) **(b)**

Figure 5.4: (a) completion detector of adder 1st stage (b) SLG of adder 1st stage.

Considering the JOIN structure in Figure 5.3 , the time taken by the precharge signal to reach multipliers from the adder is calculated as in equation 5.1.

$$T_{del} = t_{CD^2\uparrow} + t_{NAND3} + t_{buf.pc} \tag{5.1}$$

Here t_{CD^2} is the delay of the CD that detects critical paths of two multipliers. t_{NAND3}, $t_{buf.pc}$ are the delays of NAND gate and the driving buffer present in multiplier, respectively. The time for the adder first stage to enter into next evaluation phase is calculated as in equation 5.2

$$T_{nxt.ev} = t_{ev^2} + t_{CD\uparrow} + t_{NAND3} + t_{CD^2\downarrow} + t_{NAND3} + t_{buf.pc} \tag{5.2}$$

Here t_{ev^2} is the delay of the adder first stage SLG that accommodates critical paths

of two multipliers.
To avoid the pipeline failure $T_{del} < T_{nxt.ev}$. The margin time can be calculated as in equation 5.3. If the delay variations are smaller than T_{margin}, no pipeline failure happens.

$$T_{margin} = T_{nxt.ev} - T_{del} \tag{5.3}$$

The registers in the proposed structure are built using D-Latches. To meet the second challenge, all these registers are controlled by a separate control signal. This signal allows registers into the isolate state when the first stages of all multipliers are in the evaluate phase. Hence during this time, the registers can supply stable inputs to the multipliers. The control signal allows the registers into a transparent state when the first stages of the multiplier are in the isolate phase. During this time, the new data arrives at the input of all the multipliers. During the initialization period, the output of all the registers and computational units is reset to zero. The following steps explain the functionality of the FIR structure.

- After the release of initialization, all the units are ready for evaluation.

- Once the valid data x(n) arrives at the input, the first stage of all the multipliers starts evaluation process and their corresponding completion detectors anticipate the evaluation of the stages and allow the stages into isolate phase in the next step.

- When the first stages of all the multipliers are in isolate phase, before they receive precharge signal from their next stages, the registers will enter into transparent state. The input of second multiplier will change to x(n) and the first multiplier to new data x(n+1).

- Once all the multiplier first stages receives the precharge signal from their next stages, the stages will precharge and then enter into evaluation phase. By this time all the registers will enter into isolate phase, so that they can supply stable inputs to the multipliers.
  The above process repeats continuously and the output is produced at the final adder unit.

5.2.3 Simulation Results

A high throughput, 6-bit, 8-tap low pass FIR filter, with EA-Hybrid pipelined multiplier and adder units, is designed and simulated using UMC-65 nm and 180 nm technologies. Figure 5.5 shows the input and output waveforms of the FIR filter in 65 nm technology. This filter is designed with a pass edge frequency of 50 MHz, stop edge frequency of 500 MHz. An input signal that has the frequency components of 40 MHz and 600 MHz is applied to the filter through an ADC. The output signal is produced after passing the filter output through a DAC. These ADC and DAC circuits are taken from the "ahdlLib" library.

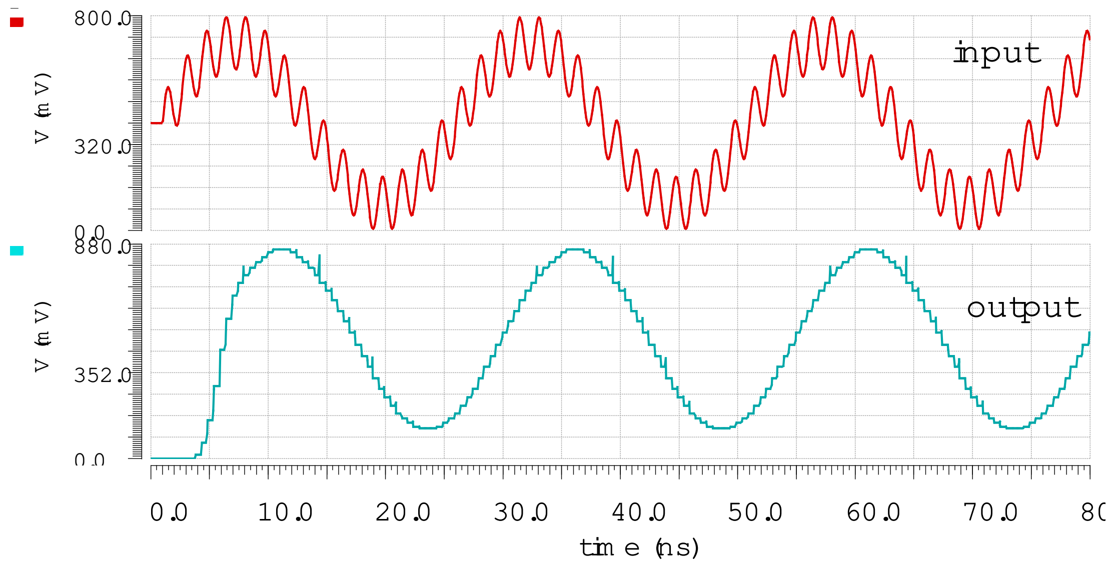


Figure 5.5: Filter input, output wave forms

The input signal is applied with a delay of 1 ns to the filter. This 1 ns duration is utilized to initialize the filter circuit. The output of the filter is obtained after an initial latency of 2.6 ns and the filter is operating at a maximum throughput of 2.3 Giga-items/sec. The filter is also tested by varying its input data rates from 500 MHz to 2.3 GHz. At all these rates the filter is functioning properly. Figure 5.6 shows the comparison of the smoothened output signal generated from the designed FIR filter and the ideal output generated from MATLAB simulations. Both the signals have the frequency component of 40 MHz, which is in the pass band limit. There is a phase shift in the actual output signal due to the latency and initialization delay of the filter circuit. As the filter is designed for unsigned numbers, and the ADC circuit needs biasing, the dc component is added to the input of the EA-Hybrid filter. Hence the output of the designed filter has a dc shift.

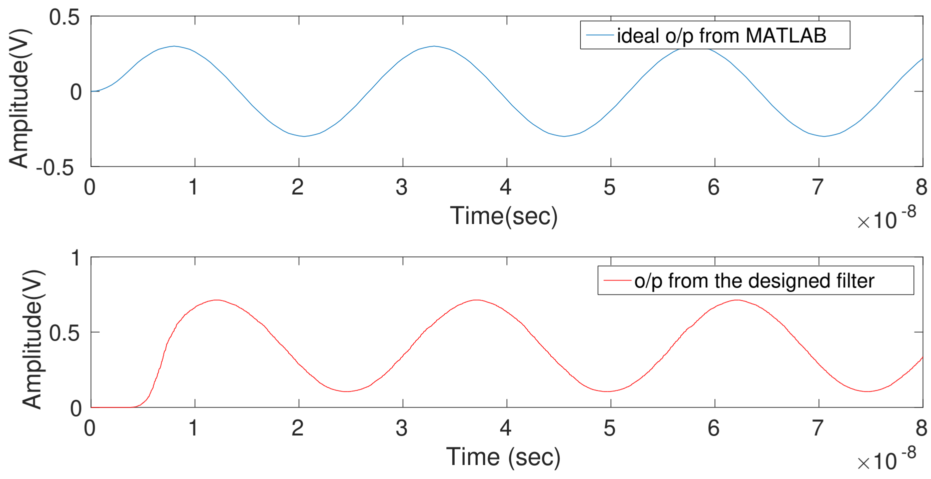


Figure 5.6: Comparison of filter output waveforms.

Table 5.2: Simulation results of FIR filter

Technology	180 nm	65 nm
Throughput	1.17 Giga-items/sec	2.3 Giga-items/sec
Latency	7.1 ns	2.6 ns
Power consumption	820 mW	78 mW
Energy-delay square product ($E.T^2$)	473.96 PJ*ns*ns	6.376 PJ*ns*ns
Range of tolerable input data rate	260 MHz - 1.17 GHz	500 MHz - 2.3 GHz

Table 5.2, shows performance parameters of the FIR filter. The results show that the FIR filter designed based on EA-hybrid technique is satisfying all the requirements of the FIR filter for PRML read channel application. Table 5.3 compares the simulation results of the proposed FIR filter with the high capacity FIR filter (HC) Singh *et al.* (2010), which is one of the best-reported FIR filters for PRML read channel ICs. The results show that the proposed FIR filter has achieved comparable throughput of HC FIR filter with the simple direct form architecture. The proposed FIR filter consumes 820 mW power. However, this number can not be compared with HC FIR filter as the reported power is only for a subpart of the filter. The robustness of the proposed design is high as it is detecting the critical path itself rather than replicating

the critical path delay in request signal path. Further, the proposed FIR filter does not require any precomputations, which in turn simplifies the coefficient updating process when the data rate changes rapidly.

Table 5.3: Comparison of HC and EA-Hybrid FIR filters (in 180 nm technology)

Pipeline method	HC (Singh *et al.* (2010))	EA-Hybrid
Architecture	Distributive arithmetic	Direct form
Maximum Throughput (Giga-items/sec)	1.3	1.17
Maximum initial Latency (ns)	10	7.1
Operating range (GHz)	0.2 - 1.3	0.26 - 1.17

5.3 Conclusion

In this chapter, an asynchronous, 6-bit, 8-tap FIR filter has been designed and simulated using UMC-180 nm and UMC-65 nm technologies. The multiplier and adder units of the filter are pipelined to the depth of gate level using EA-Hybrid pipeline method. Due to this fine grain pipelining, the critical path of FIR filter is reduced to a single gate level. The simulation results show that the asynchronous filter can operate up to the throughput of 1.17 Giga-items/sec with a latency of 7.1 ns at 2.1 V supply in 180 nm technology. The filter is also simulated using UMC-65 nm technology to see the effect of technology scaling. The throughput of the filter was improved as the technology scales down. The results demonstrate that the asynchronous EA-hybrid FIR filter concretely suits for high-performance PRML read channels.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Summary and Conclusions

In this work, two effective asynchronous domino logic pipeline styles for high throughput applications are introduced. The proposed structures are realized based on the hybrid data paths. The adoption of hybrid data paths has reduced the overhead in the logic blocks as well as in the completion detectors of the pipeline structures. This, in turn, leads to the throughput improvement of the proposed methods. The throughput of the proposed methods is further improved by designing the critical path gates using DDCVS logic instead of the DR domino logic. Furthermore, the adoption of the Early Acknowledge protocol for the EA-Hybrid method has greatly improved the throughput of this pipeline style. The presence of the isolate phase has improved the storage capacity of both the pipeline methods up to 100%.

Different digital circuits such as FIFO, Ripple carry adder, and Array multiplier are designed based on the proposed pipeline styles and simulated in 180 nm, 90 nm, and 65 nm technologies using CADENCE virtuoso tool set. As the standard cells are not available for the DR-SLG's, all the circuits in this work are full custom designs. Further, in all the circuits designed, the critical paths that decide the cycle time are optimized for the minimum delay. The simulation results in chapters 3 and 4 have proved that the proposed pipeline styles are suitable for systems that require high throughput, buffering capacity, and robustness. Though the power consumption of the proposed methods is high, their $E.T^2$ is quite good compared to other existing methods. Further, the layouts of 4-bit,10-stage FIFO and a 4-bit adder have been implemented in 180 nm technology. The FIFO has obtained 9 % and the adder has obtained 4 % lower throughputs than their schematic implementations. In Chapter 5, a high-speed asynchronous digital FIR filter suitable for PRML read channel application has been designed and simulated using UMC-180 nm and UMC-65 nm technologies. The multiplier and adder units of the filter are pipelined to the depth of gate level using EA-hybrid pipeline method. The results demonstrate that the asynchronous EA-hybrid FIR filter concretely suits for high-performance PRML read channels. The outcomes of this research work are listed below.

- Two high throughput asynchronous gate-level pipeline styles namely, EA-Hybrid and PD-Hybrid, are proposed for high throughput applications.
- The circuits FIFO, adder, and multipliers are designed based on proposed styles. The simulation results proved that the proposed pipeline styles are suitable for high-performance applications. Simulation results show that the throughput of these circuits is better than state of the art asynchronous pipelines in 180 nm, 90 nm, and 65 nm technologies. Throughput drops marginally even in post-layout simulations.
- A fully asynchronous FIR filter is designed for the first time. The results showed that the filter suits well for high-speed and low latency applications.

6.2 Scope of Future Work

The work can be extended by designing the FIR filter for signed numbers. Further, the complex modules such as FIR filter banks and RISC processors can be designed based on proposed methods. Design automation becomes an important issue while applying the proposed pipeline methods to complex modules. Design automation for layout is one of the significant issues. Standard placement and routing tools try to reduce the critical path delay in logic blocks by optimizing circuit layout. However, such optimization is not suitable for proposed methods as it might degrade the robustness of the constructed critical data path. Hence, a specific placement and routing method has to be developed to increase the delay on the constructed path. Timing verification is another critical issue in design automation. There is almost no EDA support for verifying domino gates. A high-quality domino gate library has to be developed to apply static timing analysis for verifying the timing constraints in proposed methods.

Bibliography

Anis, M., **M. Allam**, and **M. Elmasry** (2002). Impact of technology scaling on cmos logic styles. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, **49**(8), 577–588.

Beerel, P. A., **R. O. Ozdag**, and **M. Ferretti**, *A Designer's Guide to Asynchronous VLSI*. Cambridge University Press, 2010.

Calhoun, B. H., **Y. Cao**, **X. Li**, **K. Mai**, **L. T. Pileggi**, **R. A. Rutenbar**, and **K. L. Shepard** (2008). Digital circuit design challenges and opportunities in the era of nanoscale cmos. *Proceedings of the IEEE*, **96**(2), 343–365.

Carmona, J., **J. Cortadella**, **M. Kishinevsky**, and **A. Taubin** (2009). Elastic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **28**(10), 1437–1455.

Choy, C.-s., **J. Butas**, **J. Povazanec**, and **C.-f. Chan**, A fine-grain asynchronous pipeline reaching the synchronous speed. *In ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No. 01TH8549)*. IEEE, 2001.

De, V. and **S. Borkar**, Technology and design challenges for low power and high performance. *In Proceedings of the 1999 international symposium on Low power electronics and design*. 1999.

Ferretti, M. and **P. A. Beerel**, Single-track asynchronous pipeline templates using 1-of-n encoding. *In Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2002.

Hoyer, G. N., **G. Yee**, and **C. Sechen** (2002). Locally clocked pipelines and dynamic logic. *IEEE transactions on very large scale integration (VLSI) systems*, **10**(1), 58–62.

Ki, H.-J., **C.-S. Lee**, **W.-H. Paik**, **I.-C. Hwang**, **K.-Y. Chae**, **J.-S. Yoo**, and **S.-W. Kim** (2000). A low power 8-tap digital fir filter for prml read channels. *International journal of electronics*, **87**(4), 445–455.

Kuentzer, F. A., **M. Herrera**, **O. Schrape**, **P. A. Beerel**, and **M. Krstic**, Radiation hardened click controllers for soft error resilient asynchronous architectures. *In 2020 26th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, 2020*a*.

Kuentzer, F. A., **L. R. Juracy**, **M. T. Moreira**, and **A. M. Amory**, Test oriented design and layout generation of an asynchronous controller for the blade template. *In 2020 26th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, 2020*b*.

Lai, F.-s. and **W. Hwang** (1997). Design and implementation of differential cascode voltage switch with pass-gate (dcvspg) logic for high-performance digital systems. *IEEE journal of solid-state circuits*, **32**(4), 563–573.

Lines, A. M. (1998). Pipelined asynchronous circuits.

Lu, F. and **H. Samueli** (1993). A 200 mhz cmos pipelined multiplier-accumulator using a quasi-domino dynamic full-adder cell design. *IEEE Journal of Solid-State Circuits*, **28**(2), 123–132.

Mannakkara, C. and **T. Yoneda** (2010). Asynchronous pipeline controller based on early acknowledgement protocol. *IEICE TRANSACTIONS on Information and Systems*, **93**(8), 2145–2161.

Martin, A. J., **M. Nyström**, and **P. I. Pénzes**, Et2: A metric for time and energy efficiency of computation. *In Power aware computing*. Springer, 2002, 293–315.

Midhun, C., **J. Joy**, and **R. Kavitha** (2014). High-speed dynamic asynchronous pipeline: Self-precharging style. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **22**(10), 2235–2239.

Nowick, S. M. and **M. Singh** (2015). Asynchronous designâĂŤpart 1: Overview and recent advances. *IEEE Design & Test*, **32**(3), 5–18.

Ozdag, R. O., **M. Singh**, **P. A. Beerel**, and **S. M. Nowick**, High-speed non-linear asynchronous pipelines. *In Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2002.

Parhi, K. K., *VLSI digital signal processing systems: design and implementation*. John Wiley & Sons, 2007.

Pearson, D. J., **S. K. Reynolds**, **A. C. Megdanis**, **S. Gowda**, **K. R. Wrenner**, **M. Immediato**, **R. L. Galbraith**, and **H. J. Shin** (1995). Digital fir filters for high speed prml disk read channels. *IEEE Journal of Solid-State Circuits*, **30**(12), 1517–1523.

Rabaey, J. M., **A. P. Chandrakasan**, and **B. Nikolic**, *Digital integrated circuits*, volume 2. Prentice hall Englewood Cliffs, 2002.

Rezaei, H., **S. A. Moghaddam**, and **A. Rahmati** (2017). High-performance dynamic elastic pipelines. *Microprocessors and Microsystems*.

Singh, M. and **S. M. Nowick** (2007*a*). The design of high-performance dynamic asynchronous pipelines: High-capacity style. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **15**(11), 1270–1283.

Singh, M. and **S. M. Nowick** (2007*b*). The design of high-performance dynamic asynchronous pipelines: Lookahead style. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **15**(11), 1256–1269.

Singh, M. and **S. M. Nowick** (2007*c*). Mousetrap: High-speed transition-signaling asynchronous pipelines. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **15**(6), 684–698.

Singh, M., **J. A. Tierno**, **A. Rylyakov**, **S. Rylov**, and **S. M. Nowick** (2009). An adaptively pipelined mixed synchronous-asynchronous digital fir filter chip operating at 1.3 gigahertz. *IEEE transactions on very large scale integration (VLSI) systems*, **18**(7), 1043–1056.

Singh, M., **J. A. Tierno**, **A. Rylyakov**, **S. Rylov**, and **S. M. Nowick** (2010). An adaptively pipelined mixed synchronous-asynchronous digital fir filter chip operating at 1.3 gigahertz. *IEEE transactions on very large scale integration (VLSI) systems*, **18**(7), 1043–1056.

Smirnov, A., **A. Taubin**, **M. Karpovsky**, and **L. Rozenblyum**, Gate transfer level synthesis as an automated approach to fine-grain pipelining. *In Workshop on Token Based Computing (ToBaCo)*. Citeseer, 2004.

Spars, J. and **S. Furber**, *Principles Asynchronous Circuit Design*. Springer, 2002.

Sutherland, I. and **S. Fairbanks**, Gasp: A minimal fifo control. *In Proceedings Seventh International Symposium on Asynchronous Circuits and Systems. ASYNC 2001*. IEEE, 2001.

Sutherland, I. E. (1989). Micropipelines. *Communications of the ACM*, **32**(6), 720–738.

Sutherland, I. E., **R. F. Sproull**, **B. Sproull**, and **D. F. Harris**, *Logical effort: designing fast CMOS circuits*. Morgan Kaufmann, 1999.

Williams, T., *Latency and throughput tradeoffs in self-timed speed-independent pipelines and rings*. Computer Systems Laboratory, Stanford University, 1990.

Williams, T. E. and **M. A. Horowitz** (1991). A zero-overhead self-timed 160-ns 54-b cmos divider. *IEEE Journal of Solid-State Circuits*, **26**(11), 1651–1661.

Xia, Z., **M. Hariyama**, and **M. Kameyama** (2015). Asynchronous domino logic pipeline design based on constructed critical data path. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **23**(4), 619–630.

Xia, Z., **S. Ishihara**, **M. Hariyama**, and **M. Kameyama** (2010). Synchronising logic gates for wave-pipelining design. *Electronics letters*, **46**(16), 1116–1117.

Xia, Z., **S. Ishihara**, **M. Hariyama**, and **M. Kameyama** (2012*a*). Design of high-performance asynchronous pipeline using synchronizing logic gates. *IEICE transactions on electronics*, **95**(8), 1434–1443.

Xia, Z., **S. Ishihara**, **M. Hariyama**, and **M. Kameyama**, Dual-rail/single-rail hybrid logic design for high-performance asynchronous circuit. *In Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*. IEEE, 2012*b*.

Yoneda, T., **A. Matsumoto**, **M. Kato**, and **C. Myers**, High level synthesis of timed asynchronous circuits. *In Asynchronous Circuits and Systems, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on*. IEEE, 2005.

Yong, X. and **Z. Runde**, Single-track asynchronous pipeline controller design. *In Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. 2005.

Publications based on the thesis

Refereed International Journals

1. Sravani, K., and Rathnamala Rao. "A High Performance Early Acknowledged Asynchronous Pipeline using Hybrid-logic Encoding." Integration (2019). Elseveir publisher. **(SCI)**

2. K S, Rao R. "Design of high throughput asynchronous FIR filter using gate level pipelined multipliers and adders." Int J Circ Theor Appl (2020). Wiely **(SCI)**

3. Sravani, K., and Rathnamala Rao. "Novel Asynchronous Pipeline Architectures for High Throughput Applications ." AJSE (2020). Springer. **(SCI)**

Refereed International Conferences

1. Sravani,K.,& Rao,R.(2017, April). High throughput and high capacity asynchronous pipeline using hybrid logic. In Innovations in Electronics, Signal Processing and Communication (IESC), 2017 International Conference on (pp. 11-15). IEEE.

2. K. Sravani, Rathnamala Rao. DDCVS Logic for Asynchronous Gate-Level Pipelined Circuits. In International Conference Automation, Signal Processing, Instrumentation and Control-Feb,2020. Springer (**Accepted**)

Bio-data (FOR PHD)

Name	: K SRAVANI
Address	: Dept.of E&C, NITK, Surathkal, Mangalore, Karnataka - 575025, India. Ph: 9492866062.
Email	: sraav.eng@gmail.com
Educational Qualification	: **M.Tech** in Embedded systems, RGM College of Engineering and Technology, Nandyal, A.P. **B.Tech** in Electronics&Communication Engineering, RGM College of Engineering and Technology, Nandyal, A.P.
Teaching Experience	: 5 years.