ULTRA LOW POWER ASYNCHRONOUS-LOGIC QUASI-DELAY-INESENSITIVE CIRCUIT DESIGN

HQ WENG GENG

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2016
ULTRA LOW POWER ASYNCHRONOUS-LOGIC QUASI-DELAY-INSENSITIVE CIRCUIT DESIGN

HO WENG GENG
School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2016
Acknowledgement

First, I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Gwee Bah Hwee for his professional guidance in technical writing/presentation. I also would like to express gratitude to my senior colleague, Dr. Chong Kwen Siong for his full encouragement/support throughout the years of my postgraduate study.

I also would like to thank Prof. Joseph Chang for reviewing my technical papers for publication, and my colleagues, Dr. Lin Tong, Dr. Zhou Rong, Mr. Ne Kyaw Zwa Lwin, Dr. Liu Nan, Dr. Chang Kok Leong, Mr. Yee Ming Fatt, Mr. Qiu Zhaoxiang and Mr. Zhang Meng for their kind help and support in technical knowledge/skills.

Last but not least, I would like to thank my family for their patience throughout my postgraduate study.
Abstract

This thesis pertains to the investigation of low power, high robustness and yet speed-efficient digital electronics for portable/mobile/secured applications. We adopt the esoteric asynchronous-logic (async) vis-à-vis the conventional synchronous-logic (sync); more specifically, the async quasi-delay-insensitive (QDI) gate-level pipeline micro cell templates. In this thesis, there are three proposed QDI cell templates, namely low power improved Sense-Amplifier-based Pass Transistor Logic (iSAPTL), sub-threshold Autonomous Signal-Validity Half-Buffer (ASVHB) and high-speed Sense-Amplifier Half-Buffer (SAHB), which can be applied to portable/mobile/secured applications. An async network-on-chip (ANoC) based on SAHB cell template is further proposed for multi-core system-on-chip (SoC) platform, which is targeted for highly secured cryptography applications.

First, we present an async 16×16-bit pipeline multiplier based on our proposed iSAPTL with emphases on low power and high energy-delay efficiencies. The multiplier is designed as part of an async multi-core SoC. Based on the simulations @1V, 65nm CMOS process, the async iSAPTL 16×16-bit pipeline multiplier features, on average, 31% faster speed and 21% lower energy per operation, achieving an overall of 46% lower energy-delay product. It also features 16% lesser number of transistors when compared to reported SAPTL approaches.

Second, we propose an async QDI ASVHB realization approach for sub-threshold operation (V_{DD} = 0.2V). We compare our ASVHB realization approach against the competitive reported Weak-Conditioned Half-Buffer (WCHB) and Pre-
Charged Half-Buffer (PCHB) realization approaches. The ASVHB library cells, on average, features ~52% and ~47% lesser transistors than the WCHB and PCHB library cells. With respect to a 3-stage pipeline realization, the ASVHB pipeline, on average, features ~44% and ~33% lesser switching transitions per cycle than the WCHB and PCHB pipelines respectively. We further design an async 32-bit ALU based on the proposed ASVHB realization approach (@65nm CMOS process). Our ASVHB ALU occupies 0.092mm$^2$ and outperforms the WCHB and PCHB counterparts in terms of transistor-count, energy dissipation and data throughput. Overall, our proposed ASVHB design features ~41% and ~29% lesser transistors respectively than the WCHB and PCHB counterparts. At the sub-threshold operating voltage of $V_{DD} = 0.2V$, our design dissipates ~41% and ~62% lower energy respectively, and features ~5% and ~37% faster throughput than the WCHB and PCHB counterparts respectively.

Third, we propose a novel async QDI SAHB cell design approach, with emphasis on high processing speed (~GHz), high operational robustness and yet low energy dissipation. When six rudimentary library cells embodying our proposed SAHB are compared against the conventional async QDI PCHB approach, at nominal voltage of $V_{DD} = 1V @1GHz$, SAHB collectively features simultaneously ~64% lower power, ~21% faster and ~6% smaller IC-area. Three 64-bit Kogge-Stone (KS) pipeline adders based on SAHB, PCHB and sync approaches (@65nm CMOS) are designed. Both async QDI designs feature same excellent operational robustness. For 1GHz throughput and at nominal $V_{DD}$ of 1.2V, the design based on the SAHB approach features simultaneously ~56% lower
energy and ~24% lower transistor-count against PCHB approach. When benchmarked against the ubiquitous sync counterpart which requires worse case timing assumptions, our SAHB dissipates ~39% lower energy at 1GHz throughput but at the expense of ~2× more transistor-count.

Fourth, we propose an 18-bit ANoC router with 5 dual-ports based on the proposed QDI SAHB realization approach for highly secured cryptography applications. We realize the proposed ANoC router (@65nm CMOS), and benchmark it against the reported ANoC router embodying the reported WCHB QDI realization approach. Both our proposed and reported designs feature the high operational robustness. However, our design dissipates 41% lesser energy and occupies 21% smaller area than the reported WCHB counterpart. Overall, the proposed ANoC router occupies 0.105 mm² and can operate at sub-threshold voltage of 0.3V. At \( V_{DD} = 0.3V \), it dissipates 44 fJ per bit and operates at 105 ns per flit.
# Table of Contents

Acknowledgement........................................................................................................i
Abstract.........................................................................................................................ii
Table of Contents.................................................................................................v
List of Author’s Publications...............................................................................viii
Abbreviations.........................................................................................................x
List of Figures......................................................................................................xi
List of Tables......................................................................................................xiv

1. **Introduction**.................................................................................................1
   1.1 Motivation........................................................................................................1
   1.2 Objectives.......................................................................................................4
   1.3 Organization of the Thesis.........................................................................5

2. **Literature Review**.......................................................................................7
   2.1 Asynchronous (Async) Communication Channel....................................7
      2.1.1 Data Encoding: Single-rail, Dual-rail and Quad-rail........7
      2.1.2 Timing Approach: Bundled-Data, Speed-Independent,
           Delay-Insensitive and Quasi-Delay-Insensitive (QDI)....9
      2.1.3 Data-path Pipeline: Block-Level and Gate-Level................11
      2.1.4 Handshake Protocol: Four-phase and Two-phase...........14
   2.2 Classification of Async Design Approaches........................................16
   2.3 Async QDI Pipeline Cell Templates.....................................................19
      2.3.1 Sense-Amplifier-based Pass Transistor Logic
           (SAPTL)..........................................................................................19
      2.3.2 Pre-Charged Half-Buffer (PCHB).............................................22
      2.3.3 Reduced Stack Pre-Charged Half-Buffer (RSPCHB)....25
      2.3.4 Weak-Conditioned Half-Buffer (WCHB)..............................28

3. **Proposed Low Power QDI-like Improved Sense-Amplifier-based Pass
   Transistor Logic (iSAPTL).............................................................................32
   3.1 Introduction.................................................................................................32
   3.2 Proposed QDI-like iSAPTL Cell Template............................................33
   3.3 Async 8-bit Pipeline Adder.................................................................38
3.4 Async 16×16-bit Pipeline Multiplier.................................41
3.5 Summary........................................................................46

4. Proposed Sub-Threshold QDI Autonomous Signal-Validity Half-
Buffer (ASVHB)...............................................................47
4.1 Introduction..................................................................47
4.2 Proposed QDI ASVHB Cell Template.........................48
  4.2.1 Cell Structure and Operation Mechanism.................48
  4.2.2 Comparison with Reported QDI Realization
      Approaches..........................................................55
4.3 Proposed 32-bit ASVHB Arithmetic Logic Unit (ALU).........65
  4.3.1 ALU Architecture...............................................65
  4.3.2 Design Implementation.........................................67
4.4 Simulation Results......................................................68
  4.4.1 Results on Sub-threshold Operation Region.............68
  4.4.2 Comparison with Reported Designs.......................72
4.5 Summary.....................................................................76

5. Proposed High-Speed QDI Sense-Amplifier Half-Buffer (SAHB).....77
5.1 Introduction..................................................................77
5.2 Proposed QDI SAHB Cell Template............................78
  5.2.1 Template Structure.............................................78
  5.2.2 Transistor Configuration and Operating Voltage........85
  5.2.3 Transistor Sizing Optimization and Circuit Layout.....88
  5.2.4 Comparison with Reported Async Approaches........92
5.3 64-bit SAHB Kogge-Stone (KS) Adder.............................97
  5.3.1 64-bit KS Pipeline Adder..................................97
  5.3.2 IC Chip Implementation and Verification...............100
  5.3.3 Fabricated IC Measurement Results....................102
5.4 Summary.....................................................................109

6. Proposed High-Robustness Async Network-on-Chip (ANoC)
   based on SAHB............................................................111
6.1 Introduction..............................................................111
6.2 ANoC Interface Structure .........................................................113
6.3 SAHB Quad-rail Cell Design ..................................................117
6.4 Design Implementation ..........................................................119
6.5 Measurement Results and Comparison ......................................121
6.6 Summary ..............................................................................124

7. Conclusions and Recommendations for Future Work ..............125
   7.1 Conclusions ........................................................................125
   7.2 Recommendations for Future Work ......................................127
       7.2.1 Async Dynamic-Voltage-Scaling Microprocessor ............127
       7.2.2 ANoC-based Multi-core Platform .................................129

Bibliography ..............................................................................130
Appendix I: Improved SAPTL2 ..................................................138
List of Author’s Publications

Journal Publications


Conference Proceedings


## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>Arithmetic Logic Unit</td>
</tr>
<tr>
<td>Async</td>
<td>Asynchronous-Logic</td>
</tr>
<tr>
<td>ANoC</td>
<td>Asynchronous Network-on-Chip</td>
</tr>
<tr>
<td>ASVHB</td>
<td>Autonomous Signal-Validity Half-Buffer</td>
</tr>
<tr>
<td>DIMS</td>
<td>Delay-Insensitive Min-term Synthesis</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>DVS</td>
<td>Dynamic-Voltage-Scaling</td>
</tr>
<tr>
<td>GALS</td>
<td>Globally-Asynchronous-Locally-Synchronous</td>
</tr>
<tr>
<td>iSAPTL</td>
<td>Improved Sense-Amplifier-based Pass Transistor Logic</td>
</tr>
<tr>
<td>NoC</td>
<td>Network-on-Chip</td>
</tr>
<tr>
<td>NCL</td>
<td>Null Convention Logic</td>
</tr>
<tr>
<td>PCHB</td>
<td>Pre-Charged Half-Buffer</td>
</tr>
<tr>
<td>PCSL</td>
<td>Pre-Charged Static-Logic</td>
</tr>
<tr>
<td>PVT</td>
<td>Process-Voltage-Temperature</td>
</tr>
<tr>
<td>QDI</td>
<td>Quasi-Delay-Insensitive</td>
</tr>
<tr>
<td>SAHB</td>
<td>Sense-Amplifier Half-Buffer</td>
</tr>
<tr>
<td>SAPTL</td>
<td>Sense-Amplifier-based Pass Transistor Logic</td>
</tr>
<tr>
<td>STFB</td>
<td>Single-Track Full-Buffer</td>
</tr>
<tr>
<td>Sync</td>
<td>Synchronous-Logic</td>
</tr>
<tr>
<td>SoC</td>
<td>System-on-Chip</td>
</tr>
<tr>
<td>WCHB</td>
<td>Weak-Conditioned Half-Buffer</td>
</tr>
</tbody>
</table>
List of Figures

Fig. 2.1: Interface Signals for Data-encoding (a) Single-rail, (b) Dual-rail and (c) Quad-rail.................................................................8
Fig. 2.2: Illustration of Isochronic Fork.......................................................11
Fig. 2.3: Block-Level Approach....................................................................12
Fig. 2.4: Gate-Level Approach.....................................................................13
Fig. 2.5: (a) Block Diagram and Timing Diagrams for Handshaking Protocols (b) Four-phase, (c) Two-phase.............................................................16
Fig. 2.6: Classification of Async Circuits......................................................17
Fig. 2.7: Reported SAPTL Template............................................................20
Fig. 2.8: Reported Decision-making Muller C-element................................21
Fig. 2.9: Reported NMOS Pass Transistor Stacks: (a) 3-input XOR/XNOR and (b) 3-input CARRY/ICARRY.........................................................22
Fig. 2.10: Pipeline Structure of Reported PCHB Approach.........................23
Fig. 2.11: Reported 2-input PCHB Functional Block....................................24
Fig. 2.12: Reported 2-input PCHB Completion Detection Circuit..................24
Fig. 2.13: Pipeline Structure of Reported RSPCHB Approach......................26
Fig. 2.14: Reported 2-input RSPCHB Functional Block...............................26
Fig. 2.15: Reported 2-input RSPCHB Completion Detection Circuit.............27
Fig. 2.16: (a) Reported 2-input WCHB Template (b) Muller C-element Circuit...29
Fig. 2.17: Weak-Conditioned Functional Blocks for Library Cells (a) AND/NAND, (b) OR/XOR and (c) XOR/XNOR.................................29
Fig. 2.18: Pipeline Structure of Reported WCHB Approach.......................30
Fig. 3.1: Proposed iSAPTL Circuit Template..............................................34
Fig. 3.2: Proposed iSAPTL Signal Transition Graph....................................35
Fig. 3.3: Proposed Decision-Making Muller C-element................................36
Fig. 3.4: Optimized NMOS Pass Transistor Stacks for Full Adder: (a) 3-input XOR/XNOR cell and (b) 3-input CARRY/ICARRY cell.............37
Fig. 3.5: Block Diagram of 8-bit Pipeline Adder........................................38
Fig. 3.6: Block Diagram of 1-bit Full Adder.................................................39
Fig. 3.7: Async 16×16-bit Pipeline Multiplier Architecture............................42
Fig. 3.8: Block Diagrams: (a) Half Adder and (b) Full Adder......................44
Fig. 3.9: Energy Dissipation of the Proposed 16×16-bit Pipeline Multiplier @ 65nm STM CMOS Process, $V_{in} = 0.38V$, $V_{fp} = -0.45V$ ........................................46

Fig. 4.1: Block Diagram of the Proposed n-input ASVHB Cell.........................49

Fig. 4.2: Proposed ASVHB 2-input AND/NAND Cell.......................................50

Fig. 4.3: Proposed ‘Evaluate’ Sections for ASVHB Cells (a) 2-input OR/NOR, (b) 2-input XOR/XNOR, (c) 2-input MUX/IMUX and (d) 3-input AO/AOI.................................................................54

Fig. 4.4: Pipeline Structures (a) Proposed ASVHB, (b) Reported WCHB, (c) Reported PCHB and (d) Reported RSPCHB.................................58

Fig. 4.5: Marked Graph Behaviours (a) Proposed ASVHB, (b) Reported WCHB, (c) Reported PCHB and (d) Reported RSPCHB....................61

Fig. 4.6: The Proposed Async 32-bit ALU Architecture.................................66

Fig. 4.7: Layout View of the Proposed ASVHB ALU.....................................68

Fig. 4.8: Proposed ASVHB ALU at Various Temperature (a) Energy Dissipation and (b) Throughput; normalized to the readings @ 0.2V, 27°C........70

Fig. 4.9: Proposed ASVHB ALU at Various Threshold Voltages (a) Energy Dissipation and (b) Throughput; normalized to the readings @0.2V, SVT........................................................................................................71

Fig. 4.10: Async ALUs within Sub-threshold Voltage Region (a) Energy Dissipation and (b) Throughput; normalized to the readings of the proposed ASVHB ALU@ 0.2V.....................................................73

Fig. 5.1: SAHB Cell Template: (a) Generic Interface Signals, (b) Evaluation Block powered by $V_{DD,L}$ and (c) Sense-Amplifier Block powered by $V_{DD}$......................................................................................80

Fig. 5.2: Circuit Schematic of a Buffer Cell Embodying SAHB: (a) Evaluation Block and (b) Sense-Amplifier Block..............................82

Fig. 5.3: Dual-rail SAHB Library Cells: (a) 2-input AND/NAND, (b) 2-input XOR/XNOR and (c) 3-input AO/AOI.................................84

Fig. 5.4: Transistor Configurations in a 2-input SAHB AND/NAND Cell: (a) Transistors are shared and (b) Transistors are not shared; the drawings depicts the scenario when only input A is valid........86

Fig. 5.5: Timing Characteristics for SAHB Cells: (a) Timing Diagram, (b) A possible critical path of $t_F$ in Sense-Amplifier Block and (c) A possible critical path of $t_B$ in Evaluation Block..........................................89

Fig. 5.6: Normalized Parameters of a SAHB Buffer Cell at Various Critical Transistor Sizing; normalized to the reading at the operating conditions of $V_{DD} = 1V$, $V_{DD,L}=0.3V$ and input toggling rate of 1 GHz.................90

Fig. 5.7: Layout View of the SAHB Buffer Cell: (a) Various Sub-blocks and (b) Geometry Template.........................................................92
List of Tables

TABLE 2.1: One-bit Value for Single-rail Data Encoding.........................8
TABLE 2.2: One-bit Value for Dual-rail Data Encoding.........................8
TABLE 2.3: Two-bit Value for Quad-rail Data Encoding.........................9
TABLE 3.1: Comparisons of 8-bit Pipeline Adders realized using the Reported SAPTLs and Proposed iSAPTL Approaches
(V_{DD} = 1.0V, 200MHz, 65nm CMOS)........................................40
TABLE 3.2: Comparisons of Async 16x16-bit Pipeline Multipliers realized
using the Reported and Proposed iSAPTL Approaches
(V_{DD} = 1.0V, 200MHz, 65nm CMOS)........................................45
TABLE 4.1: General Features of Various QDI Realization Approaches....55
TABLE 4.2: Transistor-count of ASVHB, WCHB, PCHB and RSPCHB cells;
normalized to the readings of ASVHB cells.................................56
TABLE 4.3: Number of Transitions per cycle of ASVHB, WCHB, PCHB and
RSPCHB pipelines.....................................................................63
TABLE 4.4: Results of Async ALUs on 65nm CMOS process;
normalized to the readings of Proposed ASVHB ALU.........74
TABLE 4.5: Comparison of Various ALUs............................................75
TABLE 5.1: General Characteristics of a Buffer Cell Embodying Various
Async Cell Design Approaches......................................................93
TABLE 5.2: Parameters of Various Library Cells Embodying the SAHB
and PCHB Cell Design Approaches..............................................96
TABLE 5.3: Realization of SAHB Pipeline Blocks in the Group PG Logic..99
TABLE 5.4: Comparison of Various 64-bit Adders.................................109
TABLE 6.1: Comparison of SAHB and WCHB Cells for ANoC
implementations; normalized to readings @SAHB.............119
TABLE 6.2: Measurement Comparison of the Proposed and Reported
ANoCs.................................................................123
Chapter 1: Introduction

1.1 Motivation

With the advancement of the Network-on-Chip (NoC) technology [1], [2], multi-processor (or multi-core computing) platforms [3], [4] have been widely accepted as efficient enablers for highly parallel digital signal processing (DSP) and general purpose computing. The multi-processor platform can be adopted in electronic systems to realize low power dissipation and yet high speed computation applications. These potential applications include audio and video communications [5], [6], acoustic systems [7], [8], wireless sensor networks [9], [10], Internet of Things [11], [12], etc. In fact, each processor (in such platform) can be operated independently, e.g. by completely shutting off the processor (in idle state) or by slowing down the processor whose corresponding computation is not critical to reduce the power dissipation. To achieve such independent control, dynamic-voltage-scaling (DVS) [13], [14] can be employed with the objective of trading off energy dissipation and operating speed. For example, the supply voltage can be scaled down from the nominal voltage (for high speed high power operations) to the near-threshold voltage (for mid-speed low power operations) and to the sub-threshold voltage (for ultra-low speed ultra-low power operations). Studies had shown that the minimum energy operation of most digital circuits is occurred around the sub-threshold voltage regions [15], [16].

Despite the desirable attribute of minimum energy dissipation, sub-threshold operation (and the associated DVS) for digital circuits poses several design
challenges. In the sub-threshold operation, the on-current ($I_{on}$) of the transistor, consequently the associated circuit delay, is exponentially proportional to Process-Voltage-Temperature (PVT) variations [17], [18]. This circuit delay variation translates directly into timing uncertainties and may create data synchronisation issues if the prevalent clock-based sync methodology [19], [20] is adopted. Consequently, excessive amount of safety timing delay margin is often imposed in the clock infrastructure of the conventional synchronous-logic (sync) to accommodate the worst-case critical path delay (including the clock skew, setup-time and hold-time for registers, etc.), hence unnecessarily further (in additional to the long delay sub-threshold operation) slowing down the entire circuit and resulting in higher overall leakage energy dissipation [21], [22].

Alternatively, asynchronous-logic (async) methodology [23], [24] could be fully or in part adopted to alleviate such data synchronisation issues. The basic premise is that async circuits are essentially self-timed circuits based on async handshake protocols [25], [26] to synchronise digital operations, and thus are highly robust for data synchronisation. For reliable data synchronisation, the quasi-delay-insensitive (QDI) delay model [27], [28] is often adopted in view of its practicability to acknowledge the arrival/completion of data. The QDI async operational modality has been well established in electronic community [29], [30].

There are several reported QDI realization approaches to-date, and they can be categorized into the block-level (or also known as coarse-grain) [31], [32] and gate-level (or also known as fine-grain) methods [33], [34]. The QDI realization approaches of the block-level method include the Delay-Insensitive Min-term
Synthesis (DIMS) [35], Null Convention Logic (NCL) [36] and Pre-Charged Static-Logic (PCSL) [37]. However, these realization approaches are not speed-efficient for data propagation due to their undesirable long delay in the multi-cascading cell data-path. In contrast, the QDI realization approaches of the gate-level method potentially maximize the efficiency of the data propagation through the single-cell data-path, and thus more speed-efficient. The gate-level QDI realization approaches [38] are the recent reported Weak-Conditioned Half-Buffer (WCHB) [39], and other earlier reported Pre-Charged Half-Buffer (PCHB) [40] and Reduced-Stack Pre-Charged Half-Buffer (RSPCHB) [29], [46]. Despite its speed-efficiency advantage, the gate-level pipeline potentially suffers from high transistor-count overheads (each cell requires an acknowledgement circuitry) and potentially high power dissipation (each cell’s operation has a high number of switchings) when compared to the block-level pipeline. Furthermore, the implementation of the gate-level pipeline is more challenging than that of the block-level due to the lack of commercial EDA tool support [41], [42]; commercial EDA tools are largely RTL-based and are directly applicable to the block-level pipelines.

The QDI realization approaches can be implemented in several logic families, namely static-logic, dynamic-logic and pass-logic, etc. For robust sub-threshold operation, static-logic is adopted (over other logic families) due to its high robustness towards the PVT variations [43], [44]. Although the WCHB realization approach was reported in static-logic (and hence suitable for sub-threshold operation), it embodies large numbers of high-overhead Muller C-element [39] and independent latches in order to satisfy the QDI constraint. On the other hand, the
PCHB realization approach was reported in dynamic-logic, and might not be robust in the sub-threshold operation due to the cross-coupled inverters holding the outputs state. Although digital library cells [45] embodying the PCHB realization approach may be modified to be static-logic for high robustness, their transistor-count would be undesirably increasing and their energy efficiency would be decreasing. Consequently, both the reported WCHB and PCHB realization approaches suffer from high hardware-overhead problems, especially for robust sub-threshold applications. Put simply, the reported gate-level QDI realization approaches remain unsatisfactory for low overhead low power/energy attributes.

1.2 Objectives

The main objective of this thesis pertains to the investigation of async QDI circuit and system designs in terms of high robustness, sub-threshold low power, and high speed. Specifically, four key areas are investigated as follows:

1. To investigate, design and implement energy-delay efficient async QDI iSAPTL library cell templates and design a 16×16-bit pipeline multiplier based on the iSAPTL cells to achieve <82×10^{-21}Js energy-delay product.

2. To design sub-threshold async QDI ASVHB library cell templates for low power applications and design a 32-bit ALU @V_{DD} = 0.2V based on the ASVHB cells to achieve <1.6pJ energy dissipation.
3. To design high speed low power async QDI SAHB library cell templates for high performance applications and design a 64-bit pipeline adder based on the SAHB cells with >1GHz speed and <57.8fJ energy dissipation.

4. To design a high functionally robustness (against the voltage variation) and energy-efficient 18-bit async NoC (ANoC) for multi-core SoC to achieve <940fJ per-bit energy dissipation.

1.3 **Organization of the Thesis**

The remainder of this thesis is organized as follows.

In Chapter 2, we elaborate the async design styles, i.e. timing approach, data-path pipeline, handshaking protocol and data-encoding. We further presents the classification of the different async circuit design approaches, and review the reported QDI library cell templates.

In Chapter 3, we propose a low power Improved SAPTL (iSAPTL) async QDI-like cell template with the emphasis on high energy-delay efficiency. We demonstrate the advantages of the proposed iSAPTL in an 8-bit pipeline adder. We further present an async 16×16-bit pipeline multiplier based on our proposed iSAPTL. The pipeline adder and pipeline multiplier are benchmarked against the reported SAPTL approaches.

In Chapter 4, we propose a sub-threshold Autonomous Signal-Validity Half-Buffer (ASVHB) async QDI cell template with emphasis on low energy operation at voltage region, $V_{DD} \sim 0.2V$. We compare the proposed ASVHB against
the reported QDI cell templates. We further apply the proposed ASVHB to construct a 32-bit Arithmetic Logic Unit (ALU), which is applicable for DVS and can operate in sub-threshold region. We implement the proposed ALU and the results are benchmarked against the reported designs.

In Chapter 5, we propose a high speed Sense-Amplifier Half-Buffer (SAHB) async QDI cell template with emphasis on high operational robustness and low power dissipation. We compare the attributes of the proposed SAHB against the reported competing async cell templates. We further present a 64-bit Kogge-Stone pipeline adder embodying the proposed SAHB approach for a power management application. We implement the proposed 64-bit SAHB pipeline adder, and the results are benchmarked against the async PCHB and sync counterparts.

In Chapter 6, we propose an 18-bit ANoC router with 5 dual-ports based on the proposed high speed SAHB QDI cell template in Chapter 5. The proposed ANoC offers high robustness and low energy dissipation. We further present the details of our proposed ANoC router architecture and the proposed SAHB quad-rail cell template. We design and implement the proposed ANoC. The results are measured and benchmarked against the reported WCHB counterpart.

In Chapter 7, the conclusions of this thesis are drawn, and two potential areas for future research are recommended.
Chapter 2: Literature Review

2.1 Asynchronous Communication Channel

There are four means to classify async design styles, i.e. data-encoding, timing approach, data-path pipeline and handshaking protocol. This chapter will elaborate the details on these async design styles. Besides, this chapter presents different async circuit design approaches and reviews the reported QDI library cell templates.

2.1.1 Data Encoding: Single-rail, Dual-rail and Quad-rail

There are generally three types of data-encoding schemes [47] in the async design, i.e. single-rail, dual-rail and quad-rail. This is worthwhile to note that the dual-rail and quad-rail also carry timing information of data validity/nullity.

Fig. 2.1 depicts the async interface signals for various data encoding schemes. Single-rail data encoding is commonly used in the BD timing approach where the sender propagates the data $D$ bundled with an additional handshaking signal $Req$ (via a delay element) to the receiver, as shown in Fig. 2.1(a). In single-rail data encoding, $Req$ is used to indicate the validity/nullity for each bit of data. The valid and null values of the data are encoded by $D$ and $Req$ as depicted in Table 2.1. The delay element should be characterized such that $Req$ arrives later than $D$. This strict timing assumption unfortunately affects the operational robustness of the design with large PVT variations.
To implement the highly-robust QDI approach without the timing assumptions, the protocol cannot rely on a single wire to indicate the validity/nullity of the data. Dual-rail data encoding uses two wires named as true-rail ($D.T$) and false-rail ($D.F$) for each-bit of the data propagation, as shown in Fig. 2.1(b). The validity/nullity is encoded in the data itself. The valid and null values are encoded by $D.T$ and $D.F$ as depicted in Table 2.2.

In addition to single-rail and dual-rail, another alternative for async data encoding is quad-rail, which encodes two-bit of data into four wires, named as $D.0$, $D.1$, $D.2$ and $D.3$.
$D.1$, $D.2$ and $D.3$, as shown in Fig. 2.1(c). The valid and null values are encoded by $D.0$, $D.1$, $D.2$ and $D.3$ as depicted in Table 2.3.

<table>
<thead>
<tr>
<th>Bit-value:</th>
<th>Null</th>
<th>Valid ‘0’</th>
<th>Valid ‘1’</th>
<th>Valid ‘2’</th>
<th>Valid ‘3’</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D.0$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D.1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$D.2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$D.3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

*the rest of the combination are considered invalid

2.1.2 Timing Approach: Bundled-Data, Speed-Independent, Delay-Insensitive, and Quasi-Delay-Insensitive

The timing approach [46] dictates the assumptions on the gate and wire delays during the design process. The lesser the delay assumptions, the more robust is the design to the delay variation. Basically, there are four types of timing approaches as follows.

Bundled-data (BD) approach [47] requires bounded delay assumption, which defines the maximum bounded delay values of all gates in the circuit. The circuit is guaranteed to work correctly if these bounds are not violated. However the bounded delay assumptions may be unmatched/insufficient due to the unexpectedly large PVT variations, somewhat akin to sync circuits.

Speed independent (SI) [40], [47] approach assumes arbitrary gate delay and zero (or negligible) wire delay in the async circuits. However, in deep sub-micron technology process, the wire delay could contribute a significant amount of the total circuit delay and consequently make the SI approach somewhat less realistic for
circuit implementation [47]. In view of this, some additional assumptions are required. For example, in the case that one logic gate sends its output to another logic gate, any delay in their interconnection is aggregated into the delay of the transmitting gate; in the case that one logic gate sends its output to multiple gates, isochronicity is the default assumption.

Delay-insensitive (DI) approach [48] does not require any timing assumption on the gate and wire delays, thereby is the most robust amongst all async approaches. However, DI realization approach could be impractical in view of its large area and power dissipation overheads. Only Muller C-elements, buffers, inverters and wires can be used to build a single-output gate-level DI circuit. Hence, only limited number of true DI circuits can be constructed. As each building block of the DI circuit requires an associated Muller C-element to detect the signal validity, it increases the circuit overheads significantly. This motivates the need for less restrictive timing approaches appropriate for a lower overhead practical async design.

Quasi-delay-insensitive (QDI) approach [27], [28] assumes arbitrary delay in all gates and wires. The only timing assumption is the isochronic fork [48] wire delays. Isochronic fork assumes the delays to the different ends of the fork are the same. The purpose of the isochronic fork assumption is to ensure that the ordering of transitions at various inputs of gate, in order to preserve hazard free. Fig. 2.2 illustrates an isochronic fork at the output of a logic gate. If the fork $F$ is isochronic, the rising transitions of $A$ and $B$ at the inputs of the succeeding gates are assumed to arrive simultaneously.
The isochronic assumption of the QDI approach ensures input-completeness and no gate-orphan. Input-completeness means that the outputs only become valid when all inputs are valid and vice-versa. This input-completeness behavior is identical to Seitz’s weak conditions in [98]. In [98] Seitz further explains that if the individual components satisfy the weak conditions, then any valid combinatorial circuit structure of functional blocks also satisfies the weak conditions.

The gate-orphan refers to the unacknowledged signal transition across a gate. In fact, the timing assumption related to an isochronic fork is critical to ensure the functional robustness and it can be accommodated by keeping the logic threshold voltages to be uniform. This is to ensure the voltage level of isochronic forks varies within the bounded transition time [99]. In short, the QDI approach offers a great advantage for design simplicity in accommodating PVT variations.

2.1.3 Data-path Pipeline: Block-level and Gate-level

The data-path pipeline style determines the propagation of data in the async circuits. The adoption of pipeline styles largely depends on the circuit applications, i.e. high-speed or low-power. There are two types of data-path pipelines, i.e. block-level and gate-level approaches.
Fig. 2.3 illustrates the block-level (coarse-grain control-data decomposition) approach [31], [32], which separates the data-path and control-path. The data-path consists of a latch and a functional block. The latch stores the incoming input data, \( D \) before propagating \( D \) to the functional block. The functional block comprises multi-cascading cells to perform logic function. On the other hand, the control-path consists of an input completion tree (ICT) and a handshake controller (Ctrl). The ICT performs completion detection function for all the incoming inputs. The Ctrl sends the handshake signals to the data-path and the preceding pipeline for acknowledgement.

The block-level stage may simultaneously compute several functions and send the results to several output ports. The simplicity and generality of this approach allows ease of implementation, i.e. quick circuit design and synthesis. However, the functional block consists of multi-cascading cells, adding large delay to the pipeline, and in turn the cycle-time may increase significantly. Besides, ICT also causes the long delay, which is proportional to the logarithm of the number of bits in the data-path, to the handshake cycle. Furthermore, all the multi-cascading
cells (in the functional block) are bundled/not-interleaved in the handshake control (e.g. share the same handshake signals), hence requires the explicit storing of the incoming input data in the latch, adding energy and forward latency overheads.

Fig. 2.4 illustrates the gate-level (fine-grain integrated pipeline) approach [33], [34], which comprises a single-cell functional block, a latch and an output completion detection (OCD). The single-cell function block performs a logic function. The latch stores and propagates the evaluated data to the succeeding stage. The OCD detects the completion of operation for all output data $Q$ to $Q_N$.

![Gate-Level Approach](image)

In the gate-level approach, the functional block is integrated with its own handshake control to implement a complete pipeline stage. Each stage only compute one function and send the results to a latch. Although the gate-level pipeline requires additional design efforts for cells placement and slack-matching, it features a shorter pipeline delay due to the single-cell functional block, hence potentially reducing the cycle-time for data-propagation. Since the data-path is decomposed into smaller independent stages, the high-overhead completion trees can be eliminated and replaced by individual OCD for each stage, consequently
reducing the overall delay in the handshaking cycle. Furthermore, since each pipeline stage features its individual handshaking signals, it eliminates the need for the explicit storing of the data in the latch, potentially reducing the overhead in terms of both energy and forward latency. In general, the gate-level pipeline approach is more appropriate for high-throughput and energy-efficiency designs.

2.1.4 Handshake Protocol: Four-phase and Two-phase

In addition to the above, an async circuit can also be classified according to its handshake protocol [25], [26], either in a four-phase (level-signaling) or two-phase (transition-signaling) protocol.

Fig. 2.5(a) depicts the block diagram for illustrating the transmission of data and a handshake signal (Ack) between a sender and a receiver. Fig. 2.5(b) illustrates the timing diagram for a four-phase handshake protocol. The term four-phase refers to the number of communication actions within a handshake cycle. The operation of the four-phase protocol is as follows: (1) the sender issues a “valid” data V, (2) the receiver processes (or computes) the data and asserts Ack high, (3) the sender responds by resetting data to “empty” E (return-to-zero), (4) the receiver acknowledges this by de-asserting Ack low (return-to-zero). After that, the sender may initiate the next communication cycle. The four-phase protocol requires no timing assumptions since the “valid” data in the next cycle can only be processed after the Ack is return-to-zero in the current cycle.
Fig. 2.5(c) depicts the timing diagram for a two-phase handshake protocol. The information of the Ack is encoded as signal transition on the wires and there is no difference between a low to high and a high to low transitions as both transitions represent signal events. The operation of the two-phase protocol is as follows: (1) the sender issues a “valid” data V, (2) the receiver processes the data and sets Ack to either high (if its previous state is low, for example in 1st cycle) or low (if its previous state is high, for example in 2nd cycle) in Fig. 2.5(c). Thereafter, the sender can then initiate next communication cycle.

Since the two-phase protocol does not require return-to-zero transition [7] (as opposed to the four-phase protocol), it potentially reduces time and energy (less number of switchings per data cycle), theoretically resulting in faster operating speed and lower energy dissipation. However, as the circuit implementation in the two-phase protocol requires more complex logic circuit (transition-signaling latches) for realization, it increases the overall circuit area overhead and result in higher leakage power dissipation significantly. The two-phase protocol also requires a timing assumption that the “valid” data in the next cycle must arrive later than the delay for asserting/de-asserting Ack in the current cycle. Therefore, it may be not operational robust with large PVT variations.
2.2 Classification of Async Design Approaches

Fig. 2.6 generally classifies async for the realization of operational robust digital circuits. As sync requires clocking timing assumptions (e.g. clock skews, setup/hold times, etc.) [49], realizing operational robust circuits in sync under large PVT variation environment is challenging, requiring large timing margin to accommodate worse-case conditions. In contrast, async is an alternative to mitigate the worse-case timing assumptions. However, there are other challenges in async approach.
In Fig. 2.6, the classifications of the async digital-logic design philosophy are presented. In the timing approach classification, there are four async approaches: Delay-Insensitive (DI), Quasi-Delay-Insensitive (QDI), Speed-Independent (SI) and Bundled-Data (BD). In this classification, the DI circuits are largely impractical because they operate perfectly without any gate/wire delays, leading to the circuit realization comprising only buffer cells and Muller C-elements [48]. SI circuits, on the other hand, assume that all wire delays are negligible. This assumption is however somewhat unrealistic in the nano-scaled fabrication
processes. QDI and SI approaches have similar self-detection mechanisms [40]. QDI circuits operate error-free for arbitrary wire delays and assume isochronic forks [48], i.e. the same delays are consumed for different wire branches. This assumption can be satisfied easily in practice. For the BD circuits, they are similar to sync circuits, requiring timing assumptions for circuit realization. As their operations rely on bounded gates/wire delays similar to sync circuits, their design, is somewhat challenging to guarantee operational robustness in some conditions. In short, the QDI async approach innately detects the computation delays according to different workloads/operating conditions, offers the most practical approach [26], [33], [39], [44], [50], [51] [52], [53], [54], [55], [56] to accommodate the PVT variations, and achieves the best possible performance under the prevailing operating conditions.

In the pipeline structure classification in Fig. 2.6, both block-level and gate-level pipelines can be realized in the QDI approach. The block-level pipeline separates async controller-path and data-path so that each of them can be designed individually. Such pipeline is simple but less speed-efficient [33] as a bulk of cells would be grouped together to form a block-level pipeline stage, hence resulting in a longer critical path. QDI cell design approaches with the block-level pipeline include DIMS [35], NCL [36], PCSL [9], etc. Conversely, the gate-level pipeline is more speed-efficient since it incorporates a pair of async controller and logic cell to form a micro-cell pipeline stage, hence resulting in shorter critical path.

In the bottom level of cell template approach, QDI cell design approaches with the gate-level pipeline include our Autonomous Signal-Validity Half-Buffer
(ASVHB) [44], our Sense-Amplifier Half-Buffer (SAHB) [55], Pre-Charged Half-Buffer (PCHB) [56] and Weak-Conditioned Half-Buffer (WCHB) [39]; the ASVHB and SAHB are the proposed works herein and will be described later in Chapter 4 and Chapter 5. For completeness, there are other async cell design approaches, including PS0 [50], LP2/1 [33], STAPL [54], STFB [51], SAPTL [52] and our iSAPTL [53], etc. These approaches are considered as ‘QDI-like’ since they still require less restrictive timing assumption and this cause them less operational robust when compared to QDI circuits.

2.3 Async QDI Pipeline Cell Templates

The last row in Fig. 2.6 shows the QDI pipeline cell templates reported to-date. Of particular, SAPTL possess excellent energy-delay efficiency and QDI feature, which is reasonably high robustness towards PVT variations. Besides, PCHB and WCHB satisfy QDI feature, which does not require any timing assumption for robust and error-free operation.

2.3.1 Sense-Amplifier-based Pass Transistor Logic (SAPTL)

This section reviews the reported SAPTL approach [52] and serves as a preamble to our proposed improved SAPTL (iSAPTL) approach in Chapter 3.

Fig. 2.7 depicts the circuit template of the reported SAPTL, comprising a Stack Driver, an NMOS Pass Transistor Stack, an Output Sense-Amplifier and an AND-gate Completion Detection Circuit. The Stack Driver supplies a current at
the node $P$ for the Pass Transistor Stack which in turn evaluates the logic function, and the Output Sense-Amplifier amplifies the outputs of the Pass Transistor Stack (i.e. $S.T$ and $S.F$), generates and latches the dual-rail outputs ($Q.T$ and $Q.F$). Finally, the AND-gate provides a completion signal ($Rreq/Lack$).

Fig. 2.7: Reported SAPTL Template

The operation of a SAPTL circuit is as follows. Initially, $P$, $S.T$ and $S.F$ are ‘0’, the handshake signals ($Lreq$, $Lack$, $Rreq$ and $Rack$), Input $D$ ($A.T/A.F$, $B.T/B.F$...) and dual-rail outputs $Q.T/Q.F$ are ‘1’. The initial ‘1’ indicates that a SAPTL circuit is an active-low circuit (instead of usual active-high circuit). When $D$ arrives and is valid, $Lreq$ will be ‘0’, causing $P$ to be ‘1’ and hence either $S.T$ or $S.F$ will then be charged to ‘1’ (via Pass Transistor Stack according to $D$). The Output Sense-Amplifier will then store the valid $Q.T/Q.F$, triggering $Lack$ to ‘0’ to acknowledge the completion of evaluation; the $P$, $S.T/S.F$ signals will then be reset to ‘0’. $Lack$ ($= Rreq$) also triggers the operation for the succeeding pipeline. When $Rack$ is ‘0’ (and $Lreq$ returns to ‘1’), the $Q.T/Q.F$ become empty, returning to its initial condition (for next operation).

Fig. 2.8 depicts the decision-making Muller C-element adopted in the Output Sense-Amplifier; the $Lreq^*$ and $Lack$ signals in Fig. 2.8 are not shown in the Muller
C-element symbol in Fig. 2.7 (for consistency in [52]). Two NMOS pass transistors, controlled by $L_{req}$ and $L_{ack}$ respectively, serve as a decision-making controller so that the signal $S.T$ (or $S.F$) can trigger the output $Q.F$ (or $Q.T$). Initially, the decision-making controller is turned on and $S.T = Y.T = \text{‘0’}$ (and $S.F = Y.F = \text{‘0’}$). Consider if $S.T$ becomes ‘1’ during evaluation (when $L_{req} = 0$), $Y.T$ will become ‘1’ (via the transistor controller by $L_{ack}$) to evaluate the output $Q.F$ to ‘0’, hence a valid output. Thereafter, $L_{ack}$ will become ‘0’, and the decision-making controller is turned off. Note that $Y.T$ is initially at $V_{dd} - V_t$, and a full voltage of $Y.T$ is restored (and retained) by the keeper. $Y.T$ can only be reset to ‘0’ when $R_{ack} = 0$, $S.T = 0$ (i.e. when $D$ becomes empty), and the decision-making controller is turned on again.

![Diagram](image)

**Fig. 2.8**: Reported Decision-making Muller C-element

Fig. 2.9 (a) and (b) depict the implementations of the reported NMOS Pass Transistor Stacks for a 3-input XOR/XNOR cell and a 3-input CARRY/ICARRY cell. For example in Fig. 2.9(a), once the data-input arrives (valid), the Pass Transistor Stack provides a current path from node $P_{sum}$ to the dual-rail output nodes, asserting either $Sum.T$ or $Sum.F$ (valid).
Due to the dual-rail attribute, $S.F$ is set to be ‘0’ (if $S.T = 1$) during evaluation. In fact, $S.F$ and $Y.F$ are floating (during the evaluation stage). These floating nodes are undesirable, and may result in malfunction and high short-circuit current in the connecting Muller C-element, hence high power dissipation. In short, this reported implementation of the decision-making Muller C-element [52] is neither robust nor power-efficient.

2.3.2 **Pre-Charged Half-Buffer (PCHB)**

This section reviews the reported async QDI PCHB approach [56] and serves as the benchmark to our proposed QDI approaches in Chapters 4 and 5.

Fig. 2.10 illustrates the pipeline structure of the reported PCHB approach. The PCHB pipeline comprises a functional block integrated with a PCHB completion...
detection circuit. The completion detection circuit comprises an input completion detection (ICD), an output completion detection (OCD), an inverted Muller C-element and an inverter chain.

Fig. 2.10: Pipeline Structure of Reported PCHB Approach

Fig. 2.11 depicts the schematic of the reported async QDI dual-rail 2-input PCHB functional block [56], which is a DCVSL circuit comprising a pull-up network and a pull-down network (both are controlled by Rack and En signals), an pull-down NMOS stack, two inverters and two weak keepers (feedback inverters) at both sides of the outputs. The pull-up network pre-charges the outputs while the pull-down network performs evaluation, the pull-down NMOS stack evaluates the logic function from the dual-rail inputs \(A.T, A.F, B.T\) and \(B.F\) to the intermediate outputs \(S.T\) and \(S.F\) and the inverters generate the desired dual-rail outputs \(Q.T\) and \(Q.F\). The weak keepers serves as implicit latches to maintain the output logic states.
Fig. 2.12 depicts the reported 2-input PCHB completion detection circuit [56], which comprises an ICD, an OCD, an inverted Muller C-element and an inverters chain. ICD validates the inputs \((A.T, A.F, B.T \text{ and } B.F)\) validity/nullity and OCD validates the immediate outputs \((S.T \text{ and } S.F)\) validity/nullity. The inverted Muller C-element validates the outputs from both ICD and OCD, to generate \(Lack\) and \(En\) signals through the inverters chain, which serves as buffer in between.

The operation of a PCHB circuit is as follows. Initially, input \(D\) \((A.T, A.F, B.T, B.F)\) and output \(Q\) \((Q.T \text{ and } Q.F)\) are ‘0’ (empty); the handshake signals \((Lack \text{ and } Rack)\), enable signal \((En)\) and intermediate outputs \((S.T/S.F)\) are ‘1’. When \(D\) arrives (valid), the pull-down network evaluates the logic function, asserting \(S.T/S.F\) and \(Q\).
The validities of $D$ and $S.T/S.F$ through ICD/OCD assert $Lack$ and $En$ to ‘0’ (acknowledge the completion of evaluation). Once the succeeding stage has completed the evaluation ($Rack$ to ‘0’), the circuit. When $D$ and $Q$ are both ‘0’ (empty), ICD/OCD re-asserts $Lack$ and $En$ to ‘1’. Noted that once the succeeding stage has completed the reset operation ($Rack$ to ‘1’), a new cycle can then be initiated.

In the pipeline, PCHB cell template embodies the high overhead completion detection circuit to completely validate the input-completeness, and to perform output completion detection. The circuit overhead keeps increasing when the number of inputs increases. Besides, The PCHB cell functional block is designed in dynamic-logic style, in which the cross-coupled inverters serve as latch to maintain the outputs state. However, these cross-coupled inverters causes the PCHB cell less operational robust especially within the sub-threshold voltage region.

2.3.3 Reduced Stack Pre-Charged Half-Buffer (RSPCHB)

This section reviews the reported async QDI RSPCHB approach [29] and serves as the benchmark to our proposed QDI approaches in Chapter 4.

Fig. 2.13 illustrates the pipeline structure of the reported RSPCHB approach. The RSPCHB pipeline comprises a functional block integrated with a RSPCHB completion detection circuit. The completion detection circuit comprises an output completion detection (OCD), an inverted Muller C-element, a pair of series inverter and a single inverter.
Fig. 2.13: Pipeline Structure of Reported RSPCHB Approach

Fig. 2.14 depicts the schematic of the reported async QDI dual-rail 2-input RSPCHB functional block [29], which is a DCVSL circuit comprising a pull-up network and a pull-down network (both are controlled by Rack signal), an pull-down NMOS stack, two inverters and two weak keepers (feedback inverters) at both sides of the outputs. The pull-up network pre-charges the outputs while the pull-down network performs evaluation, the pull-down NMOS stack evaluates the logic function from the dual-rail inputs (A.T, A.F, B.T and B.F) to the intermediate outputs (S.T and S.F) and the inverters generate the desired dual-rail outputs (Q.T and Q.F). The weak keepers serves as implicit latches to maintain the output logic states.

Fig. 2.14: Reported 2-input RSPCHB Functional Block

Fig. 2.15 depicts the reported 2-input RSPCHB completion detection circuit [29], which comprises an OCD, an inverted Muller C-element, a pair of series
inverters and a single inverter. OCD validates the immediate outputs ($S.T$ and $S.F$) validity/nullity. The inverted Muller C-element validates the inputs request signal ($Lreq$) and outputs from OCD, to generate $Lack$ signal through the pair of series inverters, which serves as buffer in between. The single inverter further generates $Rreq$ signal.

![Fig. 2.15: Reported 2-input RSPCHB Completion Detection Circuit](image)

The operation of a RSPCHB circuit is as follows. Initially, the input $D$ ($A.T$, $A.F$, $B.T$, $B.F$), output $Q$ ($Q.T$ and $Q.F$) and request signals ($Lreq$ and $Rreq$) are ‘0’ (empty); the handshake signals ($Lack$ and $Rack$) and intermediate outputs ($S.T$/$S.F$) are ‘1’. When $D$ arrives (valid and $Lreq$ is ‘1’), the pull-down network evaluates the logic function, asserting $S.T$/$S.F$ and $Q$ (valid). $Lreq$ = ‘1’ and the validity of $S.T$/$S.F$ through OCD assert $Lack$ to ‘0’ (acknowledge the completion of evaluation) and $Rreq$ to ‘1’ (indicate the output is now valid). Once the succeeding stage has completed the evaluation ($Rack$ to ‘0’), the circuit de-asserts $S.T$/$S.F$ and $Q$ (empty). When $Lreq$ and $Q$ are both ‘0’ (empty), the completion detection circuit re-asserts $Lack$ to ‘1’ and $Rreq$ to ‘0’. Noted that once the succeeding stage has completed the reset operation ($Rack$ to ‘1’), a new cycle can then be initiated.

Basically, RSPCHB serves as the reduced stack version of the reported PCHB (in section 2.3.2). In the pipeline, RSPCHB cell template embodies additional request signals, $Lreq$ and $Rreq$, to indicate the assertion/de-assertion of the input data.
and output data respectively. In the functional block, RSPCHB could remove the internal En signal, thereby reducing the transistor stack sizes. In the completion detection circuit, RSPCHB enables the removal of ICD, thereby saving area and reducing capacitance on the wires. Although the overhead of the completion detection circuit is reduced, RSPCHB suffers from long delay when validating both Lreq and output from OCD (through inverted Muller C-element) to generate Lack and then Rreq. This may potentially slow down the cycle time of the pipeline. Similar to PCHB, the RSPCHB cell functional block is designed in dynamic-logic style, in which the cross-coupled inverters causes the RSPCHB cell less operational robust especially within the sub-threshold voltage region.

2.3.4 Weak-Conditioned Half-Buffer (WCHB)

This section reviews the reported WCHB approach [39] and serves as a benchmark to our proposed QDI approach in Chapter 4.

Fig. 2.16(a) depicts the schematic of the reported async QDI dual-rail 2-input WCHB template [39], which comprises a weak-conditioned functional block, a latch and an output completion detection (OCD). The weak-conditioned functional block validates the inputs validity/nullity and perform the evaluate/reset operation from the dual-rail inputs (A.T, A.F, B.T and B.F) to the intermediate outputs (S.T and S.F). The latch further process S.T/S.F and Rack to generate the desired dual-rail outputs (Q.T and Q.F). The OCD acknowledges the evaluate/reset operation by generating Lack via a NOR gate. The Muller C-element applied here is a 2-input symmetric
state-holding element in static-logic style, and its circuit schematic is depicted in Fig. 2.16(b).

![Fig. 2.16: (a) Reported 2-input WCHB Template (b) Muller C-element Circuit](image)

Fig. 2.16: (a) Reported 2-input WCHB Template (b) Muller C-element Circuit

Fig. 2.17(a)-(c) depicts three weak-conditioned functional blocks for realizing AND/NAND, OR/NOR and XOR/XNOR library cells. Basically, the functional blocks consists only the Muller C-elements and OR gates, to realize the QDI protocol requirement. The Muller C-elements check the inputs validity/nullity, validating the input-completeness of QDI feature for every input combination. WCHB library cells perform evaluation when all inputs are valid, and reset when all inputs are empty/null. The OR gates further evaluate the pre-defined logic functions.

![Fig. 2.17: Weak-Conditioned Functional Blocks for Library Cells (a) AND/NAND, (b) OR/NOR, (c) XOR/XNOR](image)
Fig. 2.18 illustrates the pipeline structure of the reported WCHB approach [39]. The async QDI weak-conditioned functional block integrates with a latch and an OCD to form a pipeline.

The operation of a WCHB circuit is as follows. Initially the input $D$ ($A.T/A.F$ and $B.T/B.F$), intermediate output $S$ ($S.T/S.F$) and output $Q$ ($Q.T/Q.F$) are set to ‘0’ (empty); the handshake signals ($Lack$ and $Rack$) and are set to ‘1’. When $D$ arrives (valid), the weak-conditioned functional block evaluates the logic function, asserting $S$ and $Q$ (valid). The validity of $Q$ de-asserts $Lack$ to ‘0’ (acknowledge completion of evaluation). Once the preceding stage has completed reset operation ($D$ resets to ‘0’), the circuit reset $S$ to ‘0’. Once the succeeding stage has completed evaluation ($Rack$ to ‘0’), the circuit further reset $Q$ to ‘0’. The nullity of $Q$ re-asserts $Lack$ to ‘1’ (acknowledge completion of reset operation). Noted that once the succeeding stage has completed the reset operation ($Rack$ sets to ‘1’), a new computation can be initiated.

In short, the WCHB cell template validates the input-completeness for every incoming input, and evaluate/reset the data output according to the handshake signals for every computation cycle. Hence, WCHB satisfies the QDI protocol,
making it operational robust and insensitive against delay variations. However, WCHB is less efficient in terms of speed/power-dissipation/area since it validates the input-completeness through a series of Muller C-elements in the weak-conditioned functional block where the overhead cost is high. Furthermore, a number of Muller C-elements are required to latch the outputs, which increases the circuit overheads. WCHB also suffers from larger number of transistor switching since it uses an external latch in the pipeline, as opposed to the integrated latch in other competitive reported QDI cell templates like SAPTL and PCHB.
Chapter 3: Proposed Low Power QDI-like Improved Sense-Amplifier-Based Pass Transistor Logic (iSAPTL)

3.1 Introduction

This chapter presents the proposed low power iSAPTL QDI-like cell template with the emphasis on high energy-delay efficiency. We demonstrate the advantages of the proposed iSAPTL using an 8-bit pipeline adder, and benchmark it against the reported counterparts. We further design and implement three 16×16-bit pipeline multipliers, one of them is based on our proposed iSAPTL approach and the other two are based on the reported SAPTL approaches, using the same 65nm CMOS process.

An SAPTL async dual-rail circuit is basically an integrated pipeline circuit where a dual-rail NMOS Pass Transistor Stack and async handshake circuits (including a Stack Driver, an Output Sense-Amplifier and a Completion Detection Circuit) are incorporated into a microcell, thereby realizing a four-phase async pipeline itself. However, the major drawbacks of the reported SAPTL circuits are long delay due to handshake cycle and large circuit overheads, thereby their energy-delay efficiency is potentially reduced.

We achieve high energy-delay efficiency for the proposed iSAPTL by two means. First, we propose to integrate the acknowledge signal Rack into the Stack Driver, hence the transistor-count overheads in the Output Sense-Amplifier can be reduced. Second, we apply transistor sharing technique [49] in the Pass Transistor Stack to reduce parasitic capacitance, hence reducing the leakage power dissipation.
3.2 Proposed QDI-like iSAPTL Cell Template

Fig. 3.1 depicts our proposed iSAPTL circuit template [53], which consists of a Stack Driver, a modified Pass Transistor Stack, an Output Sense-Amplifier and a NOR-gate Completion Detection Circuit; the novelties/difference of the proposed iSAPTL template against the reported SAPTL circuit templates [52], [57] will be elaborated. The Stack Driver supplies a voltage at node \( P \) for the Pass Transistor Stack to evaluate the logic function. The Output Sense-Amplifier consists of a Pull-up Keeper, a Pull-down Keeper and two proposed Decision-Making Muller C-elements. The Output Sense-Amplifier amplifies the outputs from the Pass Transistor Stack \( S.T \) and \( S.F \), then generates and latches the dual-rail output \( Q.T \) and \( Q.F \). The NOR-gate provides the completion signal. In the Output Sense-Amplifier, the Pull-up Keeper restores the voltage levels at nodes \( S.T/S.F \), and the Pull-down Keeper provides connections to the ground for nodes \( S.T/S.F \) (to prevent floating nodes) during the function evaluation. In the proposed iSAPTL circuit template, both the Pull-up and Pull-down Keepers prevent floating nodes during the function evaluation when compared to the reported SAPTL, there are no Pull-up and Pull-down Keepers.
The operation of the iSAPTL circuit is as follows. Initially, $P$ and $S.T/S.F$ are reset to ‘0’, the handshake signals ($Lreq$, $Lack$, $Rreq$ and $Rack$), Data Input and dual-rail outputs ($Q.T/Q.F$) are all set to ‘1’. Note that iSAPTL circuit is based on active-low logic. When Data Input arrives (becomes valid), $Lreq$ is set to ‘0’ ($Rack$ is at ‘1’), causing $P$ to assert to ‘1’, hence either $S.T$ or $S.F$ is charged to ‘1’ (via the Pass Transistor Stack). The Decision-Making Muller C-elements determine to negate either $Q.T$ or $Q.F$ to ‘0’ (valid), and latch the output. Once $Q.T/Q.F$ is valid, the Pull-up Keeper is switched on to restore the ‘1’ voltage level at either $S.T$ or $S.F$, and similarly the Pull-down Keeper is also switched on to hold the voltage level at ‘0’ for the other node. The Completion Detection Circuit triggers $Lack$ ($Rreq$) to go to ‘0’ to acknowledge the completion of function evaluation; $P$ and $S.T/S.F$ is then reset to ‘0’. The Pull-up Keeper and Pull-down Keeper are now turned off (since $Lreq^*$ is set to ‘1’ and $Rreq$ is set to ‘0’). Once the succeeding stage has accepted the output thus deasserting $Rack$ to ‘0’ and the preceding stage has reset the input thus reasserting $Lreq$ to ‘1’, the Decision-Making Muller C-element resets $Q.T/Q.F$
back to ‘1’ (empty). The Completion Detection Circuit then triggers \textit{Lack (Rreq)} to ‘1’ to acknowledge the completion of reset operation. The next computation cycle can be initiated thereafter. The complete signal transition graph of the proposed iSAPTL template is shown in Fig. 3.2.

Fig. 3.2: Proposed iSAPTL Signal Transition Graph

Fig. 3.3 depicts our proposed Decision-Making Muller C-element [53] for the iSAPTL circuit template. It adopts the structure of a 3-input asymmetric static-logic Muller C-element, which accepts the input signals \(\overline{Lreq}\) and \(Rack\) for output reset operation and \(S.T\) (or \(S.F\)) for output evaluation, and generates the output signals \(Q.F\) (or \(Q.T\)) and \(Q.\overline{F}\) (or \(Q.\overline{T}\)) for the inputs to the Completion Detection Circuit. For illustration, its operation concept is presented as follows. During evaluation, \(\overline{Lreq}\) and \(Rack\) is set to ‘1’. \(S.T\) is charged to ‘1’ connecting node \(Q.F\) to ground (discharged to ‘0’) to become valid. The cross-coupled inverters of the Muller C-element maintain the output voltage level simultaneously with the bottom transistors controlled by \(\overline{Lreq}\) and \(Rack\) (both at ‘1’). After the evaluation, \(S.T\) is
reset back to ‘0’ through the Stack Driver. During the reset operation, $Rack$ is reset to ‘0’ after the succeeding stage completes the evaluation. Once the $Data\ Input$ resets ($\overline{Lreq} = \text{‘0’}$), $Q.F$ is set to ‘1’. Again the cross-coupled inverters together with the top transistor controlled by $S.T$ maintain the output voltage level. The Decision-Making Muller C-element is now ready for next computation cycle. The same operation concept applies for $Q.T$.

![Diagram](image)

Fig. 3.3: Proposed Decision-Making Muller C-element

In the proposed Decision-Making Muller C-element, we manage to reduce the number of transistors from 16 to 10 when compared to the reported SAPTL approaches [52], [57]. The transistor switchings may potentially reduce for the same function evaluation. This is because we rearrange the $Rack$ signal into the Stack Driver for evaluation, hence the $Rack$ signal is no longer required in the Decision-Making Muller C-element. Furthermore, the series of transistors in the pull-up network (for reset operation) and pull-down network (for evaluation) are effectively shortened, hence achieving an overall faster operations (for both charging to ‘1’ and discharging to ‘0’).
Figs. 3.4 (a) and (b) depict the implementations of the optimized NMOS Pass Transistor Stacks for constructing a Full Adder cell, which consists of a 3-input XOR/XNOR cell and a 3-input CARRY/ICARRY cell. For example in Fig. 3.4(a), the NMOS transistors for different paths (where possible) are shared to reduce the number of transistors. Once the data-input arrives (valid), the Pass Transistor Stack provides a current path from node $P_{\text{sum}}$ to the dual-rail output nodes, asserting either $\text{Sum}.T$ or $\text{Sum}.F$ (valid). In the optimized NMOS Pass Transistor Stacks, we manage to reduce the number of transistors from 14 to 10 for the 3-input XOR/XNOR function and from 14 to 8 for the 3-input CARRY/ICARRY function, when compared to the reported SAPTL approaches [52], [57].

Fig. 3.4: Optimized NMOS Pass Transistor Stacks for Full Adder: (a) 3-input XOR/XNOR cell and (b) 3-input CARRY/ICARRY cell
3.3 **Async 8-bit Pipeline Adder**

To illustrate the speed and power efficacy of our proposed iSAPTL approach, we compare it against the reported SAPTL approaches [52], [57] by means of an 8-bit pipeline adder. Both our proposed and the reported 8-bit adders are implemented based on the same architecture. Fig. 3.5 depicts the pipeline structure of the 8-bit adder, which consists of eight cascading 1-bit full adders, connecting their intermediate/handshake signals between them. Fig. 3.6 depicts the block diagram of 1-bit full adder, which consists of both Sum Pass Transistor Stack, as depicted in Fig. 3.4(a), and Carry Pass Transistor Stack, as depicted in Fig. 3.4(b).

![Fig. 3.5: Block Diagram of 8-bit Pipeline Adder](image-url)
We simulate and compare 8-bit pipeline adder based on our proposed iSAPTL against the two pipeline adders based on the reported SAPTL approaches [52], [57]. The reported designs are here termed as SAPTL1 and SAPTL2 respectively. For fair comparison, all the 8-bit adders are realized using the same 65nm CMOS fabrication process and simulated at $V_{DD} = 1$V using the Cadence Spectra Simulator. The minimum-size transistor sizing is used for all transistors (except weak keepers). In the proposed iSAPTL, the sizes of the pull-up/down keepers are of minimum-size transistor sizing, 205nm/60nm for PMOS transistor and 135nm/60nm for NMOS transistor. Table 3.1 tabulates the forward delay (from $Lreq_0$ to $Rreq_7$), backward delay (from $Lreq_0$ to $Rreq_7$), total delay (sum of the forward and backward delays), power dissipation (@ 200MHz input switching rate), energy dissipation per operation, energy-delay product and the transistor-count. The %
number in the improvement column indicates our proposed iSAPTL design features a better attribute.

### Table 3.1
**Comparison of 8-bit Pipeline Adders Realized Using the Reported SAPTLs and Proposed iSAPTL Approaches** ($V_{DD} = 1V$, 200MHZ, 65nm CMOS)

<table>
<thead>
<tr>
<th>8-bit Adder</th>
<th>Proposed iSAPTL</th>
<th>Reported SAPTL1 [52]</th>
<th>Reported SAPTL2 [57]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward Delay (ns)</td>
<td>1x (2)</td>
<td>1.81x</td>
<td>1.06x</td>
</tr>
<tr>
<td>Backward Delay (ns)</td>
<td>1x (0.78)</td>
<td>1.53x</td>
<td>1.17x</td>
</tr>
<tr>
<td>Total Delay (ns)</td>
<td>1x (2.78)</td>
<td>1.73x</td>
<td>1.09x</td>
</tr>
<tr>
<td>Energy/operation (fJ)</td>
<td>1x (148)</td>
<td>2.18x</td>
<td>1.79x</td>
</tr>
<tr>
<td>Energy-Delay ($10^{-21}$ Js)</td>
<td>1x (0.4)</td>
<td>3.75x</td>
<td>2.00x</td>
</tr>
<tr>
<td>Transistor-Count</td>
<td>1x (949)</td>
<td>1.15x</td>
<td>1.18x</td>
</tr>
</tbody>
</table>

From Table 3.1, we remark that our proposed 8-bit adder features both faster forward and backward delays, by 30% and 26% faster respectively, resulting in ~28% overall shorter delay. Viewed differently, the 1-bit full adder of our proposed design, on average, can operate at 600ps (or 1.66GHz). Our proposed 8-bit adder dissipates ~49% lower energy per operation due to its much shorter delay i.e. the proposed adder is more speed-efficient. In terms of energy-delay product, our proposed design is highly competitive, by 65% better than the reported counterparts. In term of circuit area, our proposed design occupies ~14% lesser transistor-count when compared to the reported counterparts.

The speed improvement is mainly due to the absence of the pass logic decision-making controller. Particularly, there are only 3 series-connected pass transistors in our 1-bit full adder (as opposed to 4 series-connected pass transistors in the reported full adder). In general, more series pass transistors will translate into slower circuit speed. On the other hand, the power/energy improvement is
mainly due to the reduced number of switchings (256 in our proposed 8-bit adder vis-à-vis 288 in the reported 8-bit adders) and the enhancement of the floating nodes, hence lower short-circuit power dissipation.

3.4 Async 16×16-bit Pipeline Multiplier

In addition to the 8-bit pipeline adder, we further design an async 16×16-bit pipeline multiplier, which is targeted for the async multi-core SoC. We adopt the array multiplier architecture for this pipeline multiplier. Particularly, we apply our iSAPTL circuit template to the pipeline multiplier to achieve gate-level pipeline, hence resulting in higher throughput rate.

Fig. 3.7 depicts the overall architecture of the designed async 16×16-bit pipeline multiplier. The multiplier comprises one Partial Product Generator (PPG), 105 Full Adders (FA) and 15 Half Adders (HA). The PPG consists of 256 2-input iSAPTL AND/NAND gates.

The modulus operandi of the async 16×16-bit pipeline multiplier is as follows. The 16-bit dual-rail-encoded operands Input A and Input B are first input to the PPGs for partial product generation. The generated partial products then propagate through the Full Adders and Half Adders to generate the final 32-bit dual-rail-encoded product output \( P \). The propagation of data among the constituent gates within the multiplier is strictly governed by the four-phase handshake protocol using the local handshaking signals \( (Lreq, Lack, Rreq \text{ and } Rack) \). After the iSAPTL gate has completed the computation and its output data has been processed by the
succeeding gates, the iSAPTL gate will reset and is ready to accept new input data thus achieving gate-level pipeline operation.
Fig. 3.8 (a) and (b) depict the structures of the Half Adder and Full Adder respectively employed in the async 16×16-bit pipeline multiplier. The Half Adder propagates inputs $A$ and $B$ to generate the sum output $Sout$ through an iSAPTL 2-input XOR/XNOR cell and the carry output $Cout$ through an iSAPTL 2-input AND/NAND cell. The Half Adder receives $Lreq$ from the preceding stage and $Rack$ from the succeeding stage, and generates $Lack$ to the preceding stage and $Rreq$ to the succeeding stage. $Lreq$ and $Lack$ are bundled with the input-data, and $Rreq$ and $Rack$ are bundled with the output-data. Similarly, the Full Adder propagates inputs $A$, $B$ and $Cin$ to generate the sum output $Sout$ through an iSAPTL 3-input XOR/XNOR cell and the carry output $Cout$ through an iSAPTL 3-input CARRY/ICARRY cell. The handshaking signals connections are similar to that described in the Half Adder structure. Note that the additional Muller C-elements are also used in the structure for proper synchronization of the handshake signals.
To illustrate the energy-delay efficiency of our proposed iSAPTL approach, we simulate and compare async 16×16-bit pipeline multipliers based on our proposed iSAPTL against the two pipeline multipliers based on the reported SAPTL approaches [52], [57]. The reported designs are here termed as SAPTL1 and SAPTL2 respectively. For fair comparison, all the multipliers are realized in the same 65nm CMOS process and simulated at $V_{DD} = 1V$ using Cadence Spectre and Synopsys Nanosim power analysis tool. Minimum sizing is used for all transistors.
for benchmarking purpose. The simulation results are summarized in Table 3.2. The % number in the improvement column indicates our proposed iSAPTL design features a better attribute.

**TABLE 3.2**

<table>
<thead>
<tr>
<th>Asycn 16×16-bit Pipeline Multiplier</th>
<th>Proposed iSAPTL</th>
<th>Reported SAPTL1 [52]</th>
<th>Reported SAPTL2 [57]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward Delay (ps)</td>
<td>1× (643)</td>
<td>1.88×</td>
<td>1.16×</td>
</tr>
<tr>
<td>Backward Delay (ps)</td>
<td>1× (275)</td>
<td>1.47×</td>
<td>1.16×</td>
</tr>
<tr>
<td>Total Delay (ps)</td>
<td>1× (918)</td>
<td>1.75×</td>
<td>1.16×</td>
</tr>
<tr>
<td>Energy/operation (pJ)</td>
<td>1× (66)</td>
<td>1.36×</td>
<td>1.17×</td>
</tr>
<tr>
<td>Energy-Delay (10⁻¹² Js)</td>
<td>1× (61)</td>
<td>2.36×</td>
<td>1.34×</td>
</tr>
<tr>
<td>Transistor-Count</td>
<td>1× (139,926)</td>
<td>1.18×</td>
<td>1.21×</td>
</tr>
</tbody>
</table>

From Table 3.2, we remark that the async 16×16-bit pipeline multiplier realized using our proposed iSAPTL approach features, on average, ~31% shorter delay. This improvement can be largely attributed to the shorter series of transistors in the Decision-Making Muller C-element, which results in faster output and thus completion signals generation. Besides, our proposed design dissipates ~21% lower energy/operation. This is due to the lesser number of transistor switchings in the optimized Pass Transistor Stack and optimized Decision-Making Muller C-element. Overall, our proposed design features ~46% better energy-delay product and ~16% lesser transistor-count.

Fig. 3.9 depicts the energy dissipation per operation of the proposed 16×16-bit pipeline multiplier when $V_{DD} = 0.3V\leftrightarrow1.0V$. From the graph, we remarks the followings. First, the energy dissipation reduces when $V_{DD}$ is reduced. This is due to the reduction of the dynamic power dissipation. Second, the lowest energy point occurs at 0.4V, and the energy increases from 0.4V to 0.3V. This is due to the
dominant of the leakage power over the dynamic power in the sub-threshold voltage region. Third, when the operating speed of the multiplier is not critical, we can enable DVS to save up to 78% of energy dissipation.

![Energy Dissipation Diagram](image)

Fig. 3.9: Energy Dissipation of the Proposed 16x16-bit Pipeline Multiplier @ 65nm STM CMOS, $V_{tn} = 0.38V$, $V_{tp} = -0.45V$

3.5 Summary

We propose a low power iSAPTL QDI-like cell template with emphasis on high energy-delay efficiency. We demonstrate the advantages of the proposed iSAPTL in 8-bit pipeline adder, which features 28% shorter delay, 49% lower energy per operation, 65% lower energy-delay product and 14% lesser transistor-count than those based on the reported SAPTL approaches. We further design the async 16x16-bit pipeline multiplier based on our proposed iSAPTL approach. The proposed multiplier features 31% shorter delay, 21% lower energy per operation, 46% lower energy-delay product and 16% lesser transistor-count than the reported SAPTL counterparts.
Chapter 4: Proposed Sub-Threshold QDI Autonomous Signal-Validity Half-Buffer (ASVHB)

4.1 Introduction

This chapter presents a sub-threshold Autonomous Signal-Validity Half-Buffer (ASVHB) QDI cell template with emphasis on low energy operation at voltage region, $V_{DD} \sim 0.2V$ [44]. We compare the proposed ASVHB against the reported QDI cell templates. We further apply the proposed ASVHB to construct a 32-bit Arithmetic Logic Unit (ALU), which is applicable for DVS and is ideal for sub-threshold operation. We implement the proposed ALU and the results are benchmarked against the reported designs.

Our proposed ASVHB realization approach features several novel attributes. First, digital library cells embodying our proposed ASVHB realization approach are designed to have an autonomous validity signal for each data, and such autonomous validity signal distinctively makes the overall cell architecture simple without compromising the QDI assumptions. Second, the proposed ASVHB pipeline is based on the gate-level approach, hence potentially maximizing the throughput rate. Third, our proposed ASVHB realization approach belongs to static-logic (and remains QDI), hence is robust towards PVT variations and inherently appropriate for DVS.
4.2 Proposed QDI ASVHB Cell Template

4.2.1 Cell Structure and Operation Mechanism

Fig. 4.1 depicts a generic block diagram of a digital cell embodying the proposed Autonomous Signal-Validity Half-Buffer (ASVHB) realization approach. The digital cell has data of \( n \)-input \((n \geq 1)\) collectively denoted as \( \text{Data-In} \) and an output data denoted as \( Q \). \( \text{Data-In} \) and \( Q \) are represented in the well-established dual-rail encoding that requires two wires to encode a 1-bit datum. For example, for the datum \( Q \), there are a true wire \((Q.T)\) and a false wire \((Q.F)\). Both the true wire \((Q.T)\) and the false wire \((Q.F)\) are initially ‘0’ (at standby) and during the operation, one of the wires will be asserted to ‘1’ to indicate either a valid ‘0’ or a valid ‘1’ data signal. Both wires cannot be ‘1’ simultaneously. The async handshaking signals are single-ended signals \( \text{Lack} \) (i.e. an Acknowledge signal to the preceding pipeline) and \( \text{Rack} \) (i.e. an Acknowledge signal from the succeeding pipeline). The unique feature of our proposed ASVHB cell is that every datum is accompanied by its autonomous (individual) validity signal. For example, the validity signals for \( \text{Data-In} \) of all the \( n \) inputs are \( \text{Lval}_A, \text{Lval}_B, \text{etc.} \). The validity signal for \( Q \) is \( \text{Rval}_Q \).
For the ease of describing our proposed ASVHB realization approach, we depict a 2-input AND/NAND cell as an example in Fig. 4.2. The dual-rail input data signals $A$ and $B$ are $A.T/A.F$ and $B.T/B.F$ respectively, and their corresponding validity signals are $Lval_A$ and $Lval_B$ respectively. The dual-rail output data signal $Q$ is $Q.T/Q.F$, and its corresponding validity signal is $Rval_Q$. These validity signals help to validate the input completeness [40], hence fulfilling the QDI requirement.

The Acknowledge signals $Lack$ and $Rack$ indicate the acknowledgment to the left channel and from the right channel respectively. It may be cursory to note that an ASVHB cell has additional autonomous validity signals when compared to the reported dual-rail QDI cells [26], [39], [44], [50], [51], [52], [53], [54], [55], [56]. Unlike our ASVHB cell where the validity signals are integrated as parts of the individual cell, the reported QDI cells require separate input/output completion detection circuits to decode their respective input/output signals to validate the input completeness. In our ASVHB realization approach, however, we would like to remark that it is in fact the autonomous validity signal (for each datum) making the circuit implementation more efficient, hence resulting in the highly desirable low power dissipation and low transistor-count attributes (when compared to the
reported dual-rail QDI circuits). We will now discuss the novel circuit implementation of a QDI cell embodying our proposed ASVHB realization approach.

As shown in the circuit schematic in Fig. 4.2, the ASVHB 2-input AND/NAND cell consists of an output completion detection (OCD), an inverter and a functional block. The OCD is an NOR gate which generates Lack to collectively acknowledge the input data \( A \) and \( B \), indicating that the input data \( A \) and \( B \) are valid simultaneously.

![Circuit Diagram]

**Fig. 4.2: Proposed ASVHB 2-input AND/NAND Cell**
and $B$ have been computed. The inverter generates $Rval_Q$ from $Lack$ to indicate the validity (and nullity) of the output $Q$.

The functional block consists of a ‘True’ rail sub-block and a ‘False’ rail sub-block. The schematic structures of the sub-blocks are conceptually the same. For each schematic structure, it consists of a ‘Pre-charge’ section, an ‘Evaluate’ section, a Hold ‘0’ section, a Hold ‘1’ section and a cross-coupled inverter. When $Lval_A$, $Lval_B$ and $Rack$ are all ‘0’, the ‘Pre-charge’ section (in both sub-blocks) pre-charges the intermediate signals $S.T$ and $S.F$. This condition implies that the ASVHB cell can now be reset because the input data $A$ and $B$ are currently in the empty state, and the (connecting) succeeding pipeline has received the output $Q$. We further remark that because of the autonomous validity signal for each input data, we need to construct only $n + 1$ PMOS transistors in series for the ‘Pre-charge’ section. Such simple transistor configuration (for reset operation) is much more efficient when compared to the reported QDI cells which require either a separate (and yet complex) input completion detection (ICD) circuit [40] or a high fan-out complex PMOS transistor structure [39].

The ‘Evaluate’ section performs evaluate operation from $Data-In$; the $Data-In$ itself also inherently validates the input availability during the evaluate operation. The Hold ‘0’ section maintains the empty output state before the ‘True’ rail sub-block (or the ‘False’ rail sub-block) is ready for the evaluate operation. The Hold ‘1’ section maintains the valid output state before the ‘True’ rail sub-block (or the ‘False’ rail sub-block) is ready for the pre-charge operation. Both Hold ‘0’ and Hold ‘1’ sections merely serve as feedback circuits implemented in static-logic.
These feedback circuits are highly efficient in part by incorporating the handshake signals into each sub-block; other reported approaches (e.g. DIMS and PCHB) would require separate Muller C-element circuits to interface handshake signals. For the dynamic logic implementation, the Hold ‘0’ and Hold ‘1’ sections can be simplified for low transistor-count. Nonetheless, the dynamic logic implementation requires careful transistor sizing and is inappropriate for sub-threshold operation. The cross-coupled inverter latches the output, making the overall ASVHB cell as a gate-level QDI pipeline cell embodying an integrated latch.

The overall ASVHB circuit realization approach satisfies the QDI attributes where the input-completeness and acknowledgement (between pipelines) are preserved. For the input-completeness, the transistor configurations of the dual-rail input data signals \((A.T/A.F \text{ and } B.T/B.F)\) in the ‘Evaluate’ section validate the data availability and perform evaluate operation. The autonomous validity signals \((Lval_A \text{ and } Lval_B)\) in the ‘Pre-charge’ section validate the data nullity and perform the pre-charge operation. For the acknowledgement, both ‘Evaluate’ and ‘Pre-charge’ sections embody the output Acknowledge signal \((Rack)\) sending from the succeeding pipeline to repeatedly run the alternate four-phase evaluate and pre-charge operation cycles.

The operation of the ASVHB AND/NAND cell is as follows. Initially, the input data signals \((A.T/A.F \text{ and } B.T/B.F)\), the output data signals \((Q.T/Q.F)\), and the autonomous validity signals \((Lval_A, Lval_B \text{ and } Rval_Q)\) are at ‘0’, the Acknowledge signals \((Lack \text{ and } Rack)\) and the intermediate data \((S.T/S.F)\) are at ‘1’. Consider a case when the input data signals arrive such that \(A.T = ‘1’ \text{ (} A.F = ‘0’ \text{ and } Lval_A = \ldots\)
When \( B.T = '1' \) and \( B.F = '0' \) and \( Lval_B = '1' \), \( S.T \) will be switched to ‘0’ (via the ‘Evaluate’ section), and \( Q.T \) will be switched to ‘1’ (via the cross-coupled inverter). Once the output is valid \( (Q.T = '1') \), the OCD negates \( Lack \) to ‘0’ to acknowledge receipt of all the input data (i.e. input-completeness). The OCD further triggers \( Rval_Q \) to ‘1’ (via the inverter) to indicate a valid output (i.e. output-completeness). Once the succeeding pipeline has accepted the output, \( Rack \) will be negated to ‘0’.

When the preceding pipeline has pre-charged the inputs and negated \( Lval_A \) and \( Lval_B \) to ‘0’, \( S.T \) will be switched to ‘1’ (via the ‘Pre-charge’ section), hence discharging \( Q.T \) to ‘0’ (via the cross-coupled inverter). The OCD then triggers \( Lack \) to ‘1’ to acknowledge the completion of pre-charge operation. The OCD further resets \( Rval_Q \) to ‘0’ (via the inverter) to indicate an empty output. Finally, the ASVHB cell is ready for a new operation, and other input cases can be analysed similarly.

Other cells can be similarly constructed based on our proposed ASVHB realization approach. For example, Fig. 4.3 depicts the ‘Evaluate’ section of four other digital rudimentary cells. On the other hand, the ‘Pre-charge’ section is similar for all cells, depending on the number of autonomous validity signals. Although not shown, it is worthwhile to note that the Hold ‘0’ and Hold ‘1’ sections are purely in series-parallel configuration to complement the ‘Evaluate’ and ‘Pre-charge’ sections respectively for all ASVHB cells.
To summarize, our proposed ASVHB realization approach has several significant attributes as follows. First, as a result of the integrated autonomous validity signals, the overall circuit implementation is simple. Second, our ASVHB cell satisfies the QDI requirements, hence eliminating any timing assumptions (save isochronic forks) for data signal propagation. Third, our proposed ASVHB realization approach belongs to a static-logic implementation, which is robust towards the PVT variations, hence making a good candidate for sub-threshold operation. Lastly, as our ASVHB cell forms a gate-level pipeline, hence potentially featuring a high throughput rate (when compared to those block-level pipelines).
4.2.2 Comparison with Reported QDI Realization Approaches

For comparison purposes, we first summarize in Table 4.1 the general features of our proposed ASVHB realization approach and other reported QDI realization approaches, including WCHB [39], PCHB [40], RSPCHB [29], DIMS [35], NCL [36] and PCSL [9]. These realization approaches can be divided into the gate-level (single-cell) and block-level (multi-cascading cells) pipeline methods. In this chapter, as we focus on the gate-level pipeline in view of its speed efficiency, hence, our subsequent comparisons will be based on our proposed ASVHB and reported WCHB, PCHB and RSPCHB.

### Table 4.1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-path Protocol</td>
<td>Gate-level pipeline</td>
<td>Block-level pipeline</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Insensitive</td>
<td>QDI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timing Assumptions</td>
<td>Isochronic forks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Encoding</td>
<td>Dual-rail encoding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input-completeness</td>
<td>Input-complete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic Implementation</td>
<td>Static-logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robustness towards PVT Variations</td>
<td>Excellent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combinational Logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Latches</td>
<td>Not required</td>
<td>Required</td>
<td>Not required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>Separate Input Completion Detection</td>
<td>Not required</td>
<td>Not required</td>
<td>Required</td>
<td>Not Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>Separate Output Completion Detection</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
<td>Not required</td>
<td>Not required</td>
<td>Not required</td>
</tr>
</tbody>
</table>

Furthermore, in Table 4.1, all the QDI realization approaches are considered to be implemented in the static-logic implementation in order to enable low-voltage (sub-threshold) operations. We remark that the implementation in various logic
families is in fact orthogonal to the handshake protocol and is largely dependent on
the specific operating conditions. For nominal (i.e. super-threshold) voltage
conditions, these realization approaches (including our ASVHB) can be
implemented in dynamic logic for lower transistor-count (but inappropriate for sub-
threshold operations).

To evaluate our proposed ASVHB realization approach qualitatively and
quantitatively, we make the following comparison. First, from the individual
circuit-level perspective, we tabulate in Table 4.2 a comparison of the transistor-
count for six basic library cells embodying our proposed ASVHB realization
approach and that embodying the reported WCHB, PCHB and RSPCHB realization
approaches. The results are normalized with respect to the readings of the ASVHB
cells whose number of transistors are shown in the parentheses. From Table 4.2, it
is clear that our proposed ASVHB cells have the lowest number of transistor-count.
On average, the WCHB, PCHB and RSPCHB cells require ~2.1×, ~1.9× and ~1.1×
more transistor-count respectively.

<table>
<thead>
<tr>
<th>Library Cells</th>
<th>Proposed ASVHB</th>
<th>WCHB [39]</th>
<th>PCHB [40]</th>
<th>RSPCHB [29]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-input Buffer</td>
<td>1x (30)</td>
<td>1.33x</td>
<td>1.87x</td>
<td>1.17x</td>
</tr>
<tr>
<td>2-input AND/NAND</td>
<td>1x (44)</td>
<td>2.09x</td>
<td>1.77x</td>
<td>1.11x</td>
</tr>
<tr>
<td>2-input OR/NOR</td>
<td>1x (44)</td>
<td>2.09x</td>
<td>1.77x</td>
<td>1.11x</td>
</tr>
<tr>
<td>2-input XOR/XNOR</td>
<td>1x (46)</td>
<td>2.00x</td>
<td>1.83x</td>
<td>1.11x</td>
</tr>
<tr>
<td>2-input MUX/IMUX</td>
<td>1x (50)</td>
<td>2.56x</td>
<td>2.36x</td>
<td>1.10x</td>
</tr>
<tr>
<td>3-input AO/AOI</td>
<td>1x (58)</td>
<td>2.62x</td>
<td>1.59x</td>
<td>0.95x</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>1x</td>
<td>2.12x</td>
<td>1.87x</td>
<td>1.09x</td>
</tr>
</tbody>
</table>

Second, from the pipeline perspective, as depicted in Fig. 4.4(a) to (c), we
compare the pipeline structure of the proposed ASVHB realization approach
against those of the reported WCHB, PCHB and RSPCHB realization approaches. The pipelines are based on three pipeline stages where the first pipeline stage is labeled with a subscript \( i \) and the succeeding pipeline stages are labeled as \( i+1 \) and \( i+2 \) accordingly. For ease of comparison, the various blocks (e.g. functional blocks, latches (if required), ICD or OCD, inverters and Muller C-elements) are explicitly shown in their pipeline stages.
From Figs. 4.4(a) to (c), it is easy to appreciate that our ASVHB pipeline is distinctly different from the reported WCHB, PCHB and RSPCHB pipelines.
Among the two pipeline stages, the reported WCHB and PCHB pipelines comprises (2n + 1) wires per input-channel (two wires for each bit of dual-rail data signal and one wire for acknowledge signal) in which n = wordlength. In contrast, the proposed ASVHB pipeline has (3n + 1) wires per input-channel (additional wires for the autonomous validity signals). The autonomous validity signal is used to validate the data nullity and pre-charge the connecting ASVHB cells.

We see the differences between our ASVHB pipeline and the reported WCHB pipeline. Unlike the WCHB pipeline, our ASVHB pipeline does not require independent latches (e.g. L_i, L_{i+1}, L_{i+2} as depicted in Fig. 4.4(b)) to store the outputs; the latch function (i.e. cross-coupled inverter) has been integrated into the functional block in the ASVHB cell. Further, the autonomous validity signals in our ASVHB cells help in-part pre-charging the connecting ASVHB cells, abiding by the input-completeness requirement and yet making the circuit implementation simple. Whereas, for the WCHB pipeline, the functional blocks (that embodying high-overhead Muller C-elements [39]) are used to validate the input-completeness requirement. Hence, it is not unexpected that the WCHB cells, on average, require ~2.1× more transistors (see Table 4.2) than the ASVHB counterparts.

We see the differences between our ASVHB pipeline and the reported PCHB pipeline. The PCHB pipeline requires a separate ICD (e.g. ICD_i, ICD_{i+1}, ICD_{i+2} as depicted in Fig. 4.4(c)) to validate the input-completeness. Whereas, in our ASVHB pipeline, we leverage on the OCD of the preceding pipeline to generate autonomous validity signals (through an inverter) to the current pipeline. The autonomous validity signals effectively perform the input completion detection.
function. Consequently, our ASVHB realization approach is more efficient than the reported PCHB realization approach. The PCHB cells, on average, require \(~1.9\times\) more transistors (see Table 4.2) than the ASVHB counterparts.

Third, from the operation perspective (coupled from the pipeline structures), as depicted in Figs. 5(a) to (d), we compare the marked graph behaviours of the proposed ASVHB realization approach against those of the reported WCHB, PCHB and RSPCHB realization approaches. The marked graph behaviours are based on the event operations from \(i^{\text{th}}\) to \((i+2)^{\text{th}}\) pipeline stages. Their corresponding event operations are represented as nodes. For example, the nodes \(F_{i}^{e}\) and \(F_{i}^{p}\) (e.g. see Fig. 4.5(a) – (c)) are respectively denoted as evaluate and pre-charge operations for the \(i^{\text{th}}\)-stage functional block; the nodes \(L_{i}^{e}\) and \(L_{i}^{p}\) (e.g. see Fig. 4.5(b)) are interpreted accordingly for the \(i^{\text{th}}\)-stage latch. For other nodes, the subscripts ‘+’ and ‘–’ respectively denote an assertion and a negation of the signals applicable to ICD, OCD, inverter (Inv) and Muller C-element (C). The dark bold lines represent a series of event operations for completing one local cycle [33], the longest cycle in the marked graph behaviours.
Fig. 4.5: Marked Graph Behaviours (a) Proposed ASVHB, (b) Reported WCHB, (c) Reported PCHB and (d) Reported RSPCHB
From the local cycles shown in Fig. 4.5(a), (b) and (c), we determine the number of transitions per cycle of the ASVHB (T_{ASVHB}), WCHB (T_{WCHB}), PCHB (T_{PCHB}) and RSPCHB (T_{RSPCHB}) pipelines. Equations (1) to (3) generalize the number of transitions per cycle respectively. For the PCHB pipeline depicted in Fig. 4.5(c), it is interestingly to note that there are two possible local cycle paths (Path-A and Path-B). We represent those two possible local cycle paths as T_{PCHB-A} and T_{PCHB-B} according to the equations (4) and (5) respectively. For brevity in equations (1) to (5), \( t(x) \) denotes the number of transitions for any node \( x \).

\[
T_{ASVHB} = t(F_i^e) + t(F_{i+1}^e) + t(F_{i+2}^e) + t(OCD_{i+2}^-) + t(F_{i+1}^p) + t(OCD_{i+1}^+) \quad (1)
\]

\[
T_{WCHB} = t(L_i^e) + t(F_{i+1}^e) + t(L_{i+1}^e) + t(F_{i+2}^e) + t(L_{i+2}^e) + t(OCD_{i+2}^-) + t(L_{i+1}^p) + t(OCD_{i+1}^+) \quad (2)
\]

\[
T_{PCHB} = \max \{ T_{PCHB-A}, T_{PCHB-B} \} \quad (3)
\]

\[
T_{PCHB-A} = t(F_i^e) + t(F_{i+1}^e) + t(F_{i+2}^e) + t(OCD_{i+2}^-) + t(F_{i+1}^p) + t(OCD_{i+1}^+) + t(C_{i+2}^-) + t(F_{i+1}^p) + t(OCD_{i+1}^-) + t(C_{i+1}^+) \quad (4)
\]

\[
T_{PCHB-B} = t(F_i^e) + t(F_{i+1}^e) + t(F_{i+2}^e) + t(C_{i+2}^-) + t(C_{i+1}^+) + t(OCD_{i+1}^-) + t(C_{i+1}^+) \quad (5)
\]

\[
T_{RSPCHB} = t(F_i^e) + t(F_{i+1}^e) + t(F_{i+2}^e) + t(OCD_{i+2}^-) + t(F_{i+1}^p) + t(OCD_{i+1}^-) + t(C_{i+1}^+) \quad (6)
\]

where \( C_i \) = Muller C-element at stage \( i \)

Table 4.3 summarizes the different possible number of transitions per cycle for pipelines embodying 1-bit, 2-bit and 3-bit functional blocks within the ASVHB, WCHB, PCHB and RSPCHB realization approaches. The pipelines embodying 1-
bit functional blocks are the de-facto example used [33] for cycle-time analysis. The pipelines embodying multiple-bit functional blocks provide additional information on the relationship between the cycle-time and the wordlength of the pipelines. For our comparison in Table 4.3, we assume two transitions for each of the evaluate and pre-charge operations \((F^e\) and \(F^p\)) in the functional blocks for the ASVHB, PCHB and RSPCHB cells. Whereas for the WCHB functional blocks, the number of transitions for each \(F^e\) and \(F^p\) is \(2\times n\). We further assume two transitions for each the latch evaluate and pre-charge operations \((L^e\) and \(L^p\)), one transition for OCD assertion/negation and two transitions for Muller C-element assertion/negation.

**TABLE 4.3**

<table>
<thead>
<tr>
<th>Wordlength</th>
<th>Number of Transitions per cycle</th>
<th>Proposed ASVHB</th>
<th>WCHB [39]</th>
<th>PCHB [40]</th>
<th>RSPCHB [29]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-bit</td>
<td>10</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>2-bit</td>
<td>10</td>
<td>18</td>
<td>15</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>3-bit</td>
<td>10</td>
<td>22</td>
<td>15</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Normalized Average</td>
<td>1×</td>
<td>1.8×</td>
<td>1.5×</td>
<td>1.4×</td>
<td></td>
</tr>
</tbody>
</table>

For the ASVHB pipeline, the number of transition per cycle remains constant at 10 transitions for various wordlength. The low and constant number of transitions (for any wordlength of the ASVHB pipeline) implies potential high speed. We, nonetheless, remark that the actual speed of the ASVHB cell will vary, in part due to the fan-in of the functional blocks.

For the WCHB pipeline, as the number of transitions for \(F^e\) and \(F^p\) increases when the number of inputs increases, the overall number of transitions will increase from 14 (in the 1-bit pipeline) to 22 (in the 3-bit pipeline). The increase of the
number of transitions in $F^e$ and $F^o$ is mainly due to a need of multiple cascading cells to propagate signals to accommodate all the inputs.

For the PCHB pipeline, the longest cycle for a 1-bit pipeline involves Path-A (see Fig. 4.5(c)). Path-A involves $F_{i+2}^e$ and $OCD_{i+2}^+$ subsequently (3 transitions in total) before the $C_{i+2}^-$ node in $(i+2)^{th}$-stage. Hence, the number of transitions per cycle for the 1-bit pipeline is 14 transitions; see equation (4). For the 2-bit pipeline and 3-bit pipeline, the longest cycle now involves Path-B, which includes $ICD_{i+2}^+$ (4 transitions for 2-input and 3-input cells) before the $C_{i+2}^-$ node in $(i+2)^{th}$-stage. As a result, the total number of transitions per cycle for both the 2-bit pipeline and 3-bit pipeline is 15 transitions; see equation (5).

By comparison in Table 4.3, we find that the proposed ASVHB pipeline features the lowest number of transitions per cycle. The reported WCHB, PCHB and RSPCHB pipelines, on average, undesirably require 1.8×, 1.5× and 1.4× more transitions per cycle respectively. This is because the proposed ASVHB pipelines involve only $t(F)$ and $t(OCD)$ in completing one local cycle. Whereas the reported WCHB and PCHB/RSPCHB pipelines respectively require the additional $t(L)$ (for output latching) and the additional $t(C)$ (for validating ICD and OCD), featuring longer delay. For completeness, the throughput analysis of the pipeline by using the number of transitions per cycle remains contentious. This is because the transistor stacking will in part affect the overall speed. For example, the longer PMOS and NMOS transistors in series of the transistor stacking may cause longer delay per transition, and vice-versa. To a large extent, the lower number of
transitions also implies lesser number of switchings, potentially reducing the dynamic power dissipation.

4.3 Proposed 32-bit ASVHB Arithmetic Logic Unit

4.3.1 ALU Architecture

We demonstrate an async 32-bit Arithmetic Logic Unit (ALU) using our novel ASVHB realization approach. The proposed async ALU computes either 32-bit arithmetic operations (Add, Subtract, Accumulate) or logic operations (AND, OR, XOR). Fig. 4.6 depicts the overall architecture of the proposed async 32-bit ALU, comprising two main parts, an Arithmetic Unit and a Logic Unit, together with an Input De-multiplexer and an Output Multiplexer. The input signals are 32-bit dual-rail signals $A$ and $B$, and the output signal is 32-bit dual-rail signal $S$. The various operations are determined by the single-rail control signals $Add\_On$, $Sub\_On$, $Acc\_On$, $AND\_On$, $OR\_On$, $XOR\_On$. For simplicity, the handshake signals within various circuits are not shown. The input signals $A$ and $B$ are the data input to the Input De-multiplexer for either arithmetic or logic operation. After the operation, the Output Multiplexer selects the output from either the Arithmetic Unit or the Logic Unit. Finally the output signal $S$ is computed.

In the Arithmetic Unit, a 32-bit Kogge-Stone (KS) adder forms the core. The KS adder operand $a$ is driven by the input signal $A$ through the Input De-multiplexer. The adder operand $b$ is driven by either the input signal $B$ through the Input De-multiplexer (for add and subtract operations), or the feedback-output of
the KS adder (for accumulate operation). The adder operand $c$ is driven by the 3:1 multiplexer which selects the control signals $Add\_On$, $Sub\_On$ or $Acc\_On$ for add, subtract, or accumulate operations respectively in the Arithmetic Unit. The output of the KS adder (operand $s$) is connected to a 1:2 result de-multiplexer which determines the output is either sent back to the adder operand $b$ (for accumulate operation) or directly sent to the Output Multiplexer.

In contrast, a 32-bit Logic Operation Block is the core component in the Logic Unit. The configuration is much simpler since both operands of the AND, OR and XOR operations are connected to the input signals $A$ and $B$ through the Input De-multiplexer. The output of the Logic Operation Block is connected to a 3:1 multiplexer, and that eventually passes over to the Output Multiplexer.

![Diagram](image)
4.3.2 Design Implementation

The async ALU embodying the proposed ASVHB realization approach is implemented based on the STMicroelectronics (STM) 65nm CMOS process. Minimum sizing is used for all transistors. We adopt a full-custom approach to implement the ASVHB ALU. The layouts of all library cells, including the ASVHB cells and standard (single-rail) library cells, are first hand-crafted using the Cadence Virtuoso Layout tool. The extracted library-exchange-format (LEF) files of the library cells are generated using the Cadence Abstract Generator tool. From the Verilog files, the ASVHB ALU layout is placed and routed using the Cadence First Encounter tool. Eventually, the ASVHB ALU layout is simulated and verified using the Synopsys Nanosim tool. Fig. 4.7 depicts the layout view of our 32-bit async ASVHB ALU. Our proposed ASVHB ALU core features 304µm × 304µm, equivalent to 0.092mm² area in total. We analyse the energy dissipation per operation and throughput of our ASVHB ALU. The energy dissipation per operation represents the power-delay product for the given throughput. The throughput represents the fastest operating speed for the data propagation in each pipeline.
4.4 Simulation Results

4.4.1 Results on Sub-threshold Operation Region

In view of the robust sub-threshold operations for our intended low power applications, we consider temperature variations @ high temperature (100ºC), room temperature (27ºC) and low temperature (-40ºC) within the sub-threshold voltage region, ranging from $V_{DD} = 0.15V$ to 0.4V.

Fig. 4.8(a) depicts the normalized energy dissipation per operation of our proposed ASVHB ALU; the results are normalized to the readings @ 0.2V, 27ºC. From the graph we remark the followings. First, the higher operating temperature causes the higher energy dissipation. This is due to the increase in the leakage
power dissipation when the temperature increases, causing the overall power dissipation to be increased. Second, the minimum energy voltage point increases as the temperature increases. For example the minimum energy voltage point increases from $V_{DD} = 0.2V$ at -40°C, to $V_{DD} = 0.25V$ at 27°C, to $V_{DD} = 0.3V$ at 100°C. Since the leakage power dissipation increases as the temperature increases (while dynamic power dissipation largely remains the same), the domination effect of the leakage power dissipation over the dynamic power dissipation becomes increasingly larger, moving the minimum energy voltage point further towards higher operating voltage.

Fig. 4.8(b) depicts the normalized throughput of our proposed ASVHB ALU; the results are normalized to the readings @ 0.2V, 27°C. From the graph we remark the followings. First, as expected, the throughput reduces as the $V_{DD}$ reduces. Second, the throughput increases when the temperature increases due to the sub-threshold operation effects [15]. This is in contrast to within the nominal voltage region, in which the throughput reduces when the temperature increases due to the slower electron mobility at high temperature [49].
Besides, we consider three threshold voltage variations of high threshold voltage (HVT; $V_T \approx |0.41|$), standard threshold voltage (SVT; $V_T = |0.38|$) and low threshold voltage (LVT; $V_T \approx |0.26|$) process options for sub-threshold operation, ranging from $V_{DD} = 0.15V$ to 0.4V.

Fig. 4.9(a) depicts the normalized energy dissipation per operation of our proposed ASVHB ALU; the results are normalized to the readings @ 0.2V, SVT. From the graph we remark the followings. First, the lower threshold voltage causes the higher energy dissipation. This is due to the increase in the leakage power dissipation when the threshold voltage decreases, causing the overall power
dissipation to be increased. Second, when $V_{DD} < 0.3V$ the design @ SVT dissipates lower energy than that of HVT. This is due to the sub-threshold effects which cause poor delay in design @ HVT although it dissipates lower leakage power dissipation. Third, the minimum energy voltage point remains the same at $V_{DD} = 0.25V$ as the threshold voltage changes.

Fig. 4.9(b) depicts the normalized throughput of our proposed ASVHB ALU; the results are normalized to the readings @ 0.2V, SVT. From the graph we remark the followings. First, as expected, the throughput reduces as the $V_{DD}$ reduces. Second, the throughput increases when the threshold voltage decreases.

Fig. 4.9: Proposed ASVHB ALU at Various Threshold Voltages (a) Energy Dissipation and (b) Throughput; normalized to the readings @ 0.2V, SVT
4.4.2 Comparison with Reported Designs

To better illustrate the energy and throughput advantages of our proposed ASVHB ALU within the sub-threshold voltage region (range from $V_{DD} = 0.15V$ to $0.4V$ @ $27^\circ C$, SVT), we benchmark our design against the async ALUs embodying the reported WCHB and PCHB realization approaches using the same fabrication process. The comparison is based on the pre-layout simulations.

Fig. 4.10(a) depicts the energy dissipation per operation of our proposed ASVHB ALU, the reported WCHB and PCHB ALUs; the results are normalized to the readings of the proposed ASVHB ALU @ 0.2V. From the graph we remark the followings. First, among all designs, our design dissipates the lowest energy within the sub-threshold region. This is due to the novel features of the proposed ASVHB cells, resulting in the lower transistor-count and lesser transistor switchings, hence in turn reducing the overall energy dissipation. Second, although all the designs have the similar minimum energy voltage point at $V_{DD} = 0.25V$; our design is nonetheless more energy-efficient than the reported designs. Third, when further scaling the voltage downwards below $V_{DD} = 0.25V$, the energy dissipation increases again (as the leakage energy now dominates) for all the designs. In this $0.15V \leq V_{DD} \leq 0.25V$ region, the energy dissipation for our design only increases marginally, whereas for the reported WCHB and PCHB ALUs increase significantly. The relatively flat increase of the energy dissipation for our design indicates that our design is highly appropriate for ultra-low voltage low power applications.

Fig. 4.10(b) depicts the normalized throughput of our proposed ASVHB ALU, the reported WCHB and PCHB ALUs; the results are normalized to the
readings of the proposed ASVHB ALU @ 0.2V. From the graph we remark that all the designs have the similar speed at 0.4V, and our design is marginally more speed-efficient when scaling downwards the voltage. We further remark that our intention in this work is for low energy dissipation. Hence we adopt minimum transistor sizing for all designs. For high speed operation, larger transistor sizing can be employed and thus higher energy dissipation is expected.

![Normalized Energy Dissipation and Throughput Graphs](image)

**Fig. 4.10:** Async ALUs within sub-threshold voltage region (a) Energy Dissipation and (b) Throughput; normalized to the readings of the proposed ASVHB ALU@ 0.2V

For completeness, we further compare three async ALUs in terms of transistor-count, energy dissipation per operation and throughput at both the sub-
threshold (0.2V) and at nominal (1V) voltages respectively. Table 4.4 tabulates the normalized results of the async ALUs. The results with respect to the proposed ASVHB ALU are indicated in the parentheses.

<table>
<thead>
<tr>
<th>Async ALUs</th>
<th>Transistor-Count</th>
<th>Energy Dissipation</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$V_{DD} = 0.2V$</td>
<td>$V_{DD} = 1V$</td>
</tr>
<tr>
<td>Proposed ASVHB</td>
<td>1× (68,734)</td>
<td>1× (958fJ)</td>
<td>1× (13.5pJ)</td>
</tr>
<tr>
<td>WCHB [39]</td>
<td>1.65×</td>
<td>1.68×</td>
<td>1.30×</td>
</tr>
<tr>
<td>PCHB [40]</td>
<td>1.40×</td>
<td>2.63×</td>
<td>1.68×</td>
</tr>
</tbody>
</table>

In terms of the transistor-count, our design features the lowest. Particularly, the WCHB and PCHB counterparts require 1.65× and 1.40× more transistors respectively than our design. The normalized result is largely in-line with the analysis of the transistor-count for the library cells of various QDI realization approaches (Section II. (b)). However, the normalized transistor-count ratio of the ALUs is noted to be smaller than that of the library cells. This is because the ALUs of various QDI realization approaches embody not only their own dual-rail library cells but also the standard single-rail cells for handshake signals.

In terms of the energy dissipation at both the sub-threshold and nominal voltages, our proposed design is shown to be dissipated the lowest energy. The WCHB and PCHB counterparts respectively dissipate 1.68× and 2.63× higher energy than our design @0.2V, and respectively dissipate 1.3× and 1.68× higher energy @1V. The energy reduction is smaller @1V as the leakage energy reduction is expected to be smaller.
In terms of the throughput @0.2V, our design performs the fastest (i.e. 7.6MHz). The WCHB and PCHB counterparts are respectively 0.95× and 0.73× slower than our design. However when operating @1V, the PCHB counterpart is the fastest, followed by the WCHB counterpart and our design is the slowest. We attribute the reason to the minimum sizing used (especially for PMOS transistors) for slow speed in our design. As discussed earlier, the operating speed could be enhanced by up-sizing the PMOS transistors but at the cost of higher power.

For completeness, Table 4.5 tabulates the comparison of various ALUs. As the process technology, design architecture, wordlength, implementation of those are different; the comparison of the ALUs is somewhat contentious. Nonetheless, our proposed ASVHB ALU (and the WCHB and PCHB ALUs) feature high robustness (excellent PVT immunity) due to the nature of QDI operation, and they are suitable candidates for sub-threshold operation and for dynamic-voltage-scaling (DVS). Furthermore, our design shows the impressive energy efficiency advantage through dissipating the lowest energy per operation.

TABLE 4.5
COMPARISON OF VARIOUS ALUS

<table>
<thead>
<tr>
<th>ALUs</th>
<th>CMOS (nm)</th>
<th>VDD (V)</th>
<th>Wordlength/ Pipeline structure/ Logic Family</th>
<th>PVT immunity/ Dynamic Voltage Scaling/ Min Op. Region</th>
<th>Energy (pJ)</th>
<th>Throughput (GHz)</th>
<th>Area (mm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vangal [61]</td>
<td>130</td>
<td>1.43</td>
<td>32-bit; NP; Domino-logic</td>
<td>Weak; non-DVS; nom^</td>
<td>74.0</td>
<td>5.00</td>
<td>0.135</td>
</tr>
<tr>
<td>Shimazaki [60]</td>
<td>180</td>
<td>1.8</td>
<td>64-bit; NP; Domino-logic</td>
<td>Moderate; DVS; near-th^</td>
<td>670.0</td>
<td>1.16</td>
<td>0.152</td>
</tr>
<tr>
<td>Mathew [63]</td>
<td>90</td>
<td>1.3</td>
<td>32-bit; NP; Semi-dynamic</td>
<td>Weak; non-DVS; nom^</td>
<td>34.0</td>
<td>7.00</td>
<td>0.073</td>
</tr>
<tr>
<td>Wijeratne [59]</td>
<td>65</td>
<td>1.3</td>
<td>32-bit; NP; Domino-logic</td>
<td>Weak; non-DVS; nom^</td>
<td>1,200.0</td>
<td>9.00</td>
<td>-</td>
</tr>
<tr>
<td>Truong [3]</td>
<td>65</td>
<td>1.3</td>
<td>16-bit; NP; Domino-logic</td>
<td>Moderate; DVS; near-th^</td>
<td>40.4</td>
<td>1.20</td>
<td>0.004</td>
</tr>
<tr>
<td>WCHB</td>
<td>65</td>
<td>1</td>
<td>32-bit; P*; Static-logic</td>
<td>Excellent; DVS; sub-th^-</td>
<td>17.6</td>
<td>0.86</td>
<td>-</td>
</tr>
<tr>
<td>PCHB</td>
<td>65</td>
<td>1</td>
<td>32-bit; P*; Static-logic</td>
<td>Excellent; DVS; sub-th^-</td>
<td>22.7</td>
<td>1.16</td>
<td>-</td>
</tr>
<tr>
<td>Proposed ASVHB</td>
<td>65</td>
<td>1</td>
<td>32-bit; P*; Static-logic</td>
<td>Excellent; DVS; sub-th^-</td>
<td>13.5</td>
<td>0.81</td>
<td>0.092</td>
</tr>
</tbody>
</table>

# Non-pipeline;  
*Pipeline;  
^Nominal;  
&Near-threshold;  
~Sub-threshold
4.5 Summary

We propose a sub-threshold ASVHB QDI cell template for low voltage low power operation. The proposed ASVHB realization approach is compared with the competitive reported WCHB and PCHB realization approaches. From the circuit perspective, the ASVHB library cells, on average, features ~52% and ~47% lesser transistors than the reported WCHB and PCHB library cells. From the pipeline perspective, the ASVHB pipeline, on average, features ~44% and ~33% lesser transitions per cycle than the reported WCHB and PCHB pipelines. We implement the async 32-bit ALU embodying the proposed ASVHB realization approach based on 65nm CMOS process and compare it with the reported WCHB and PCHB counterparts. Among the three implemented designs, our ASVHB ALU occupies 0.092mm$^2$ and features the lowest transistor-count; our design features ~41% and ~29% lesser transistors respectively than the WCHB and PCHB counterparts. At the sub-threshold voltage (0.2V), our design dissipates the lowest energy per operation, i.e. ~41% and ~62% lower energy respectively than the WCHB and PCHB counterparts. At 0.2V, our design also features the fastest throughput, i.e. ~5% and ~37% faster than the WCHB and PCHB counterparts. When further benchmarked against the various reported ALUs [58], [59], [60], [61], [62], [63], our ASVHB ALU features high robustness, is suitable for sub-threshold operation and DVS, and energy-efficient.
Chapter 5: Proposed High-Speed QDI Sense-Amplifier Half-Buffer (SAHB)

5.1 Introduction

This chapter presents the proposed high-speed SAHB - a novel QDI cell template with emphases on high operational robustness and low energy dissipation. We compare the attributes of the proposed SAHB against the reported competing async cell templates. We further describe a 64-bit Kogge-Stone (KS) pipeline adder embodying the proposed SAHB approach for a power management application. We implement the proposed 64-bit SAHB pipeline adder, and the results are benchmarked against the async PCHB and sync counterparts.

There are several novel features in SAHB. First, an SAHB cell incorporates an evaluation block and a sense-amplifier (SA) block [81] to perform an async four-phase QDI operation [28], thereby innately accommodating timing issue in the presence of PVT variations. Second, the SA block embodies a cross-coupled latch with a positive feedback mechanism to speed-up the output, and latches it. Third, the evaluation block and SA block are tightly coupled to reduce the number of switching nodes, resulting in short cycle-time and low power dissipation [64]. Forth, both pull-up and pull-down networks in the evaluation block comprise only NMOS transistors. Particularly, the NMOS pull-up network adopts minimum transistor sizing (vis-à-vis a PMOS pull-up network whose transistor sizing is often 2× larger) to reduce parasitic capacitance/power dissipation [82]. Fifth, the SAHB cell is realized in static-logic style, hence appropriate for dynamic-voltage-scaling (DVS) for $V_{DD}$ from nominal-voltage (1.4V) to deep-sub-threshold (0.3V).
5.2 Proposed QDI SAHB Cell Template

5.2.1 Template Structure

Fig. 5.1 (a) depicts the generic interface signals for the proposed dual-rail SAHB cell template. The data inputs are $Data_{in}$ and $nData_{in}$, and the data outputs are $Q.T/Q.F$ and $nQ.T/nQ.F$. The left-channel handshake outputs are $L_{ack}$ and $nL_{ack}$, and the right-channel handshake inputs are $R_{ack}$ and $nR_{ack}$. $nData_{in}$, $nQ.T$, $nQ.F$, $nL_{ack}$ and $nR_{ack}$ are logical-complementary signals to the primary input/output signals $Data_{in}$, $Q.T$, $Q.F$, $L_{ack}$ and $R_{ack}$ respectively. For brevity, we would only use the primary input/output signals to delineate the operation of an SAHB cell. An SAHB cell satisfies an async four-phase (4-$\phi$) handshake protocol, having two alternate operation sequences – evaluation and reset. Initially, $L_{ack}$ and $R_{ack}$ are ‘0’ and both $Data_{in}$ and $Q.T/Q.F$ are empty, i.e., both of the rails in each signal are ‘0’.

During the evaluation sequence, when $Data_{in}$ is valid (i.e., one of the rails in each signal is ‘1’) and $R_{ack}$ is ‘0’, $Q.T/Q.F$ is evaluated and latched, and $L_{ack}$ is asserted to be ‘1’ to indicate the validity of the output. During the reset sequence, when $Data_{in}$ is empty and $R_{ack}$ is ‘1’, $Q.T/Q.F$ is reset to be empty, and $L_{ack}$ is de-asserted to be ‘0’. Thereafter, the SAHB cell is ready for next operation.

Figs. 5.1 (b) and Fig. 5.1 (c) respectively depicts two blocks – an evaluation block powered by $V_{DD,L}$ and an SA block powered by $V_{DD}$ – that collectively constitute an SAHB cell. The $V_{DD,L}$ and $V_{DD}$ can be the same voltage or different; see Section 5.1.2 later. The evaluation block comprises an NMOS pull-up network and an NMOS pull-down network to evaluate/reset the dual-rail output ($Q.T/Q.F$).
Of particular interest, the NMOS pull-up network features low parasitic capacitances (lower than the usual PMOS pull-up network whose transistor sizing is often 2× larger than that in the NMOS). On the other hand, the SA block comprises an SA cross-coupled latch, complementary buffers and a completion detection circuit. The SA cross-coupled latch amplifies (with a positive feedback mechanism) and latches \( Q.T/Q.F \). The complementary buffers generate the complementary output signals (\( nQ.T/nQ.F \)), and the completion detection circuit generates the left-channel handshake signals (\( L_{\text{ack}}/nL_{\text{ack}} \)).
For ease of illustration, Figs. 5.2 (a) and (b) respectively depict the circuit schematic of the evaluation block and SA block of a buffer cell embodying SAHB; various sub-blocks are shown in the dotted lines. The NMOS transistor in grey with RST is optional for cell initialization. Initially, A.T, A.F, Rack and Lack are ‘0’ and nA.T, nA.F, nRack and nLack are ‘1’. During the evaluation phase, for example, when
A.F = ‘1’ (nA.F = ‘0’) arrives, the voltage at node Q.F is partially charged up to V\textsubscript{DD,L} by the NMOS pull-up network in the evaluation block, and Q.T remains as ‘0’ (via the NMOS pull-down network). At the same time, as the input is now valid, the SA cross-coupled latch is turned on by connecting the virtual supply V\textsubscript{DD,V} to V\textsubscript{DD}, and amplifies Q.F to be ‘1’. Q.F is then latched (together with the PMOS feedback transistors and the cross-coupled inverters), and nQ.F becomes ‘0’ (to disconnect the node Q.F from the V\textsubscript{DD,L} in the evaluation block to prevent any short-circuit currents). L\textsubscript{ack} is asserted to ‘1’ (nL\textsubscript{ack} = ‘0’) to indicate the validity of the dual-rail output. During the reset phase, the input is empty (nA.T and nA.F are ‘1’) and R\textsubscript{ack} = ‘1’, the dual-rail output becomes empty and L\textsubscript{ack} is de-asserted to be ‘0’, thereby ready for a new operation. Both the evaluation block and SA block are tightly coupled to reduce the number of switching nodes, hence enhancing high speed and low power operation. Furthermore, as both the evaluation block and SA block operate in static-logic style, their transistor sizings are not critical.
Fig. 5.2: Circuit Schematic of a Buffer Cell Embodying SAHB: (a) Evaluation Block and (b) Sense-Amplifier Block
Figs. 5.3 (a) – (c) depict the circuit schematic of three basic SAHB library cells: 2-input AND/NAND, 2-input XOR/XNOR and 3-input AOI/AOI cells. These library cells will be used for benchmarking and for realizing the 64-bit SAHB pipeline adder (delineated in Section III later). Other SAHB library cells are designed similarly.
Fig. 5.3: Dual-rail SAHB Library Cells: (a) 2-input AND/NAND, (b) 2-input XOR/XNOR and (c) 3-input AO/AOI
5.2.2 Transistor Configuration and Operating Voltage

In the evaluation block, there are two ways to configure the connection of transistors for a multiple-input SAHB cell. For example, Figs. 5.4 (a) and (b) depict the two transistor configurations for $Q.F$ in a 2-input AND/NAND SAHB cell. The transistor configuration in Fig. 5.4 (a) is the configuration adopted in our library for its lesser transistor-count. However, their voltage supplies $V_{DD,L}$ and $V_{DD}$ need to be carefully chosen to prevent an early output transition before all the inputs ($A.F$ and $B.F$) are valid. For example in Fig. 5.4 (a), $Q.F$ will be partially charged up to $V_{DD,L}$ when either $A.F$ or $B.F$ is ‘1’. Conversely, the transistor configuration in Fig. 5.4 (b) is less power-delay-efficient, but it ensures that the current $I$ is only conducted when all the inputs are valid, hence the voltage supplies $V_{DD,L}$ and $V_{DD}$ are not critical and/or can be connected together.
We will now further elaborate on the voltage condition in SAHB cells using the transistor configuration in Fig. 5.4 (a). Assume that $Q.F$ is initially at 0V, and we express $V_X$ in (7), as the switching threshold voltage that causes the inverter (in the SA block) to switch.
\[ V_X \approx \frac{k \cdot V_{DD}}{l + k} \]  \hspace{1cm} (7)

where \( k \) = the PMOS over NMOS transistor width and \( l \) = the electron over hole saturation mobility.

Assuming \( nR_{ack} = V_{DD}, A.F = V_{DD} \) and \( nQ.F = V_{DD} \) as depicted in Fig. 5.4(a), the voltage at \( Q.F (V_{Q.F}) \) can be expressed in (8).

\[
V_{Q.F} = \begin{cases} 
V_{DD} - V_{in}, & \text{if } V_{DD} \leq V_{DD,L} \\
V_{DD,L} - V_{in}, & \text{if } V_{DD} \geq V_{DD,L} \text{ & } V_{DD} - V_{DD,L} \leq V_{in} \\
V_{DD,L}, & \text{if } V_{DD} \geq V_{DD,L} \text{ & } V_{DD} - V_{DD,L} > V_{in} 
\end{cases} \hspace{1cm} (8)
\]

When input \( A \) is valid, the current \( I \) will charge \( Q.F \) despite the input \( B \) being empty. Hence \( V_{Q.F} \) must be smaller than \( V_X \) in order to prevent the dual-rail output from being valid, as expressed in (9). Otherwise, the SAHB cell may operate too early, potentially violating the transition sequences with its neighboring SAHB cells.

\[ V_{Q.F} < V_X \]  \hspace{1cm} (9)

For \( V_{DD} = 1V \) and \( l \approx 3 \), and for the chosen 65nm CMOS process where \( V_{in} = 0.38V \), we design the inverter with \( k \approx 1.6 \) (hence \( V_X \approx 0.35V \) as ascertained from (7)) and setting \( V_{DD,L} \leq 0.3V \) so that the \( V_{Q.F} (\leq 0.3V \) as ascertained from (8)) is lower than \( V_X \), fulfilling condition (9). Since the evaluation block is not the speed-critical block, a low voltage for \( V_{DD,L} \) does not compromise the overall speed but desirably somewhat reduces the leakage power dissipation.
5.2.3 Transistor Sizing Optimization and Circuit Layout

To evaluate an SAHB cell, we first depict in Fig. 5.5 (a) its general timing characteristics. The forward delay $t_F$ is defined as the time duration when $Data_{in}$ is valid (and $R_{ack} = '0'$) until $L_{ack}$ is asserted (during the evaluation phase). The backward delay $t_B$ is defined as the time duration when $Data_{in}$ is empty (and $R_{ack} = '1'$) until $L_{ack}$ is de-asserted (during the reset phase). For completeness, note that $t_F + t_B$ constitutes the fastest delay of an SAHB cell.

With reference to the SAHB buffer cell (see Fig. 5.2), Fig. 5.5 (b) depicts a possible critical path that asserts $Q.T$ until $L_{ack}$ is asserted. For $t_F$, the PMOS transistor sizing along the dark line is critical. On the other hand, Fig. 5.5 (c) depicts a possible critical path that de-asserts $Q.T$ until $L_{ack}$ is de-asserted. For $t_B$, the NMOS transistor sizing along the dark line is critical.
Fig. 5.5: Timing Characteristics for SAHB Buffer Cell: (a) Timing Diagram, (b) A possible critical path of \( t_F \) in Sense-Amplifier Block and (c) A possible critical path of \( t_B \) in Evaluation Block.
Specifically, the critical PMOS transistors (those shown along the dark lines in Fig. 5.5(b)) are critical-path minimum-sized to be 410nm/60nm, and the critical NMOS transistors (those shown along the dark lines in Fig. 5.5(c)) are optimize-sized to be 270nm/60nm. Fig. 5.6 depicts the normalized delay, power dissipation and diffusion (active) area of an SAHB buffer cell based on various critical transistor sizings; the critical transistor sizing is normalized to the minimum sizing (denoted as ×1 where their unit values are 147ps, 7.12μW and 6665nm² respectively). For critical transistor sizing having a multiple value ×n, those critical PMOS and NMOS transistors would then have their corresponding width ×n.

![Fig. 5.6: Normalized Parameters of a SAHB Buffer Cell at Various Critical Transistor Sizing; normalized to the reading at the operating conditions of V_{DD} = 1V, V_{DD,L}=0.3V and input toggling rate of 1 GHz](image)

From Fig. 5.6, we remark the following. First, when the critical transistor sizing increases, the delay decreases until a minimum point (where the transistor sizing is ×6) where the delay is ~0.75× of the delay for unit transistor size. Second, as expected, when the critical transistor sizing increases, both the power dissipation and the diffusion area increase significantly. For the critical transistor sizing of ×6,
the SAHB buffer cell is 1.95× higher power dissipation and 3.15× larger diffusion area than the unit transistor. To mitigate the power and area overheads, we adopt critical-path minimum-sized transistor sizing for the PMOS and NMOS transistors in the critical path. For completeness, other (less critical) PMOS and NMOS transistors (those not shown along the dark lines in Figs. 5.5 (b) and (c)) are sized to 205nm/60nm and 135nm/60nm respectively.

Fig. 5.7 (a) depicts the layout view of the SAHB buffer cell whose total area is 5µm × 4.6µm. We implement our SAHB library cells based on the fixed-height standard cell approach where the height of the cells is fixed at 5µm, and their width is a multiple of 0.2µm (depending on their complexity). Fig. 5.7 (b) depicts some geometry distances/rules so that, without violating any design rules, our SAHB library cells can be placed together. At both edges, the width of both the PMOS guard ring (N+) and NMOS guard ring (P+) is 0.355µm. The width of the N well (for PMOS transistors) and P substrate (for NMOS transistors) is 2.52µm and 1.77µm respectively. The width of both the supply rails $V_{DD}$ and ground (gnd) is 0.56µm, and the width of the supply rail $V_{DD,L}$ is 0.31µm. All SAHB cells are verified by Cadence’s Abstract Generator, and their LEF (library exchange format) file is generated for the auto place-and-route process.
5.2.4 Comparison with Reported Async Approaches

Table 5.1 tabulates several characteristics of a buffer cell embodying various async cell design approaches; the buffer cell is the de facto circuit for analysis (although the results may vary for other cells). Consider first the overall perspective of the various approaches as a preamble to the interpretation of the benchmarking. As tabulated in Table 5.1, the SAHB, PCHB and WCHB buffer cells are fully QDI, hence featuring excellent robustness. The PS0, LP2/1, SAPTL, STAPL and STFB buffer cells require timing assumptions for their implementation/operation, hence their robustness are somewhat compromised. From Table 5.1, as expected, the two-phase (2-ϕ) handshaking protocol buffer cells, i.e. STAPL and STFB, feature fast cycle-time and good static slack. The cycle-time is defined as the number of switching transitions to complete one cycle in a 3-stage pipeline ring. The static slack is the maximum token occupancy in one pipeline stage during the operation; the 2-ϕ and 4-ϕ buffer cell have 100% (full-buffer) and 50% (half-buffer) occupancies respectively. The STFB buffer cell has the best cycle time, and the
PS0 buffer cell has the least transistor-count.

### TABLE 5.1
**GENERAL CHARACTERISTICS OF A BUFFER CELL EMBODYING VARIOUS ASYNC CELL DESIGN APPROACHES**

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Fully QDI Template</th>
<th>Timed (QDI-like) Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic family implementation</td>
<td>Static</td>
<td>Static</td>
</tr>
<tr>
<td>Robustness (timing)</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>Handshake</td>
<td>4-ϕ</td>
<td>4-ϕ</td>
</tr>
<tr>
<td>Cycle time (transitions)</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>Forward latency (transitions)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Area (transistors)</td>
<td>34</td>
<td>44</td>
</tr>
<tr>
<td>Static slack (%)</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

In view of our intended power management application with full-range DVS, we will now focus on the fully QDI cell templates – our proposed SAHB and the reported PCHB that both feature excellent operational robustness. To appreciate their different circuit realizations, the buffer cell embodying PCHB is depicted in Fig. 5.8. In comparison, our SAHB buffer cell (see Fig. 5.2 earlier), the SA block in our SAHB buffer cell incorporates both the input and output detection circuits (instead of being separate entities in the PCHB buffer cell). This enables our SAHB cell to feature lesser number of internal switching nodes, hence more area-/power-efficient. Further, as the SA block incorporates the integrated sense-amplifier circuitry, the speed of the data propagation in our SAHB cell is enhanced, hence the SAHB cell is more speed-efficient. From Table 5.1, both our SAHB and the PCHB buffer cells have the same forward latency and static slack. However, SAHB desirably features a faster cycle-time and requires a smaller IC area realization.
On the basis of simulations, Table 5.2 benchmarks the power dissipation, delay, power×delay product, power×delay² product and IC area of six library cells embodying our SAHB and the competing PCHB cell design approaches. For ease of interpretation, the readings of the PCHB library cells are normalized with respect to those of the SAHB library cells whose actual values are shown within.
parentheses. The average attributes of the six library cells are tabulated in the last row.

It is apparent from Table 5.2 that the library cells embodying PCHB, on average, dissipate 2.8× higher power and operate 1.27× slower speed than those embodying SAHB. Consequently, in terms of power×delay and power×delay² products, the library cells embodying PCHB are uncompetitive, on average, by 3.58× and 4.71× worse respectively, than those embodying SAHB. In terms of IC area, the library cells embodying both the PCHB and SAHB are largely comparable; on average the PCHB library cells occupy 1.06× larger IC area. In short, the library cells embodying SAHB are simultaneously superior in terms of power, delay and IC area than those embodying PCHB.

As tabulated in Table 5.2, the library cells embodying WCHB, on average, dissipate 1.52× higher power and operate 1.35× slower speed than those embodying SAHB. Consequently, in terms of power×delay and power×delay² products, the library cells embodying WCHB are, on average, by 2.02× and 2.79× worse respectively, than those embodying SAHB. In terms of IC area, on average the WCHB library cells occupy 1.3× larger IC area. In short, the library cells embodying SAHB are better as well in terms of power, delay and IC area than those embodying WCHB.
**Table 5.2**

**Parameters of Various Library Cells Embodying the SAHB, PCHB and WCHB Cell Design Approaches**

<table>
<thead>
<tr>
<th>No</th>
<th>Library Cells</th>
<th>Power (µW) @ 1V, 1GHz</th>
<th>Delay t\textsubscript{F} + t\textsubscript{B} (ps) @ 1V</th>
<th>Power × Delay (10\textsuperscript{-12} J)</th>
<th>Power × Delay\textsuperscript{2} (10\textsuperscript{-21} Js)</th>
<th>IC area (µm × µm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SAHB</td>
<td>PCHB</td>
<td>WCHB</td>
<td>SAHB</td>
<td>PCHB</td>
</tr>
<tr>
<td>1</td>
<td>1-input Buffer</td>
<td>1× (7.1)</td>
<td>1.37×</td>
<td>1.95×</td>
<td>1× (114)</td>
<td>1.38×</td>
</tr>
<tr>
<td>2</td>
<td>2-input AND/NAND</td>
<td>1× (14.1)</td>
<td>2.74×</td>
<td>1.45×</td>
<td>1× (196)</td>
<td>1.36×</td>
</tr>
<tr>
<td>3</td>
<td>2-input OR/NOR</td>
<td>1× (11.1)</td>
<td>2.82×</td>
<td>1.49×</td>
<td>1× (190)</td>
<td>1.40×</td>
</tr>
<tr>
<td>4</td>
<td>2-input XOR/XNOR</td>
<td>1× (12.1)</td>
<td>2.61×</td>
<td>1.41×</td>
<td>1× (244)</td>
<td>1.15×</td>
</tr>
<tr>
<td>5</td>
<td>2-input MUX/IMUX</td>
<td>1× (14.1)</td>
<td>2.61×</td>
<td>1.41×</td>
<td>1× (272)</td>
<td>1.13×</td>
</tr>
<tr>
<td>6</td>
<td>3-input AO/ADI</td>
<td>1× (13.8)</td>
<td>2.65×</td>
<td>1.40×</td>
<td>1× (245)</td>
<td>1.22×</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>1× (11.6)</td>
<td>2.80×</td>
<td>1.52×</td>
<td>1× (216)</td>
<td>1.27×</td>
</tr>
</tbody>
</table>
5.3 64-bit SAHB Kogge-Stone Adder

Consider now the benchmarking for a larger circuit. This section describes the implementation of a 64-bit KS pipeline adder embodying SAHB cell design approach, and benchmarked to PCHB.

5.3.1 64-bit KS Pipeline Adder

Fig. 5.9 depicts a simplified architecture of the proposed SAHB adder. The primary input operands are \( A = A_{63} \cdots A_0 \), \( B = B_{63} \cdots B_0 \) and the Carry-in \( C_{in} \), the primary output operands are \( S = S_{63} \cdots S_0 \) and the Carry-out \( C_{out} \); for sake of illustration, the async handshake signals (and their complementary signals) are not shown. The SAHB adder (consisting of a Bitwise Propagate-Generate (PG) Logic, a Group PG Logic and a Sum Logic) is constructed in a multiple carry look-ahead tree-level so that the carry propagation time is shortened, hence increasing speed [70]. Overall, 8 pipeline stages are required in the SAHB adder, resulting in a (forward) latency of 8 pipeline delays and a throughput rate of an inverse of cycle-time for one pipeline stage. The analytical equations to compute the various Propagate signals \( P(i)_n \), \( P(i:k)_n \) and \( P(k-1)_n \), and various Generate signals \( G(i)_n \), \( G(i:k)_n \) and \( G(k-1)_n \) at pipeline \( n \) are well-documented [70]. From Fig. 5.9, four SAHB library cells, i.e. Buffer, AND/NAND, XOR/XNOR and AO/AOI cells, are used, and their schematics were depicted in Figs. 5.2 and 5.3; other single-rail library cells (e.g. Muller C-elements, etc.) are also used.
Fig. 5.9: Simplified Architecture of a 64-bit Kogge-Stone Adder
Table 5.3 tabulates the realization of the SAHB pipeline blocks in Group PG Logic in terms of symbol view, cell view and SAHB design view. In general, the handshake signal to preceding pipeline stage \((\text{Ack}_{n-1})\) is asserted when the SAHB pipeline cells have evaluated their outputs, or de-asserted when the SAHB pipeline cells have reset the outputs to empty. The handshake signal from succeeding pipeline stage \((\text{Ack}_n)\) indicates whether the outputs \(G(i:k)_n\) and/or \(P(i:k)_n\) are accepted by the next connecting SAHB pipeline cells. A Muller C-element is used to ‘join’ the \(\text{Ack}_{n-1}\) generated from two parallel SAHB pipeline cells if the same input \(P(i:k)_{n-1}\) is accepted by them simultaneously (see last row in Table 5.3). The handshake connections for other SAHB pipeline cells can be constructed similarly.

<table>
<thead>
<tr>
<th>No.</th>
<th>Symbol View</th>
<th>Cell View</th>
<th>SAHB Design View</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(G(i:j)_{n-1})</td>
<td><img src="" alt="Buffer" /></td>
<td><img src="" alt="SAHB Buffer" /></td>
</tr>
<tr>
<td>2</td>
<td>{ (G(i:j)<em>{n-1}), (P(i:j)</em>{n-1}) }</td>
<td><img src="" alt="AO/AOI" /></td>
<td><img src="" alt="SAHB AO/AOI" /></td>
</tr>
<tr>
<td>3</td>
<td>{ (G(k-1:j)<em>{n-1}), (P(k-1:j)</em>{n-1}) }</td>
<td><img src="" alt="AO/AOI" /></td>
<td><img src="" alt="SAHB AO/AOI" /></td>
</tr>
</tbody>
</table>

TABLE 5.3
REALIZATION OF SAHB PIPELINE BLOCKS IN THE GROUP PG LOGIC

99
5.3.2 IC Chip Implementation and Verification

Figs. 5.10 (a) and (b) respectively depict the microphotograph and the layout of the SAHB adder @65nm CMOS process ($V_{tn} = 0.38V$, $V_{tp} = -0.45V$), with the test structure. We adopt a hybrid full-/semi-custom approach to implement the SAHB adder. The layouts of all the SAHB library cells and standard (single-rail) library cells, including Muller C-elements, were first hand-crafted using the Cadence layout tool. With the extracted LEF files of the library cells and the Verilog files of the SAHB adder, the layout of the SAHB adder is placed and routed using the Cadence First Encounter tool. The SAHB adder netlist is simulated/verified using the Synopsys Nanosim tool, and prototype ICs are physically tested/measured. The core area of the SAHB adder is $306\mu m \times 209\mu m$.

![SAHB KS Adder Test Structure](a) ![SAHB KS Adder Test Structure](b)

Fig. 5.10: The 64-bit SAHB KS Pipeline Adder: (a) Microphotograph and (b) Layout View

All 20 SAHB adder prototype ICs were measured and were fully functional. Of these 20 chips, five chips are functionally for $V_{DD} \geq 0.25V$, and the remaining
chips are functionally for $V_{DD} \geq 0.3\text{V}$.
Fig. 5.11(a) depicts a prototype chip operating at 0.25V. Put simply, our SAHB approach is applicable for full-range DVS (i.e. sub-threshold voltage (0.3V) $\rightarrow$ near-threshold voltage $\rightarrow$ nominal voltage (1.0V)) [37]. The reported QDI and QDI-like designs (PCHB, PS0, etc.), however, would likely be more applicable to half-range DVS (i.e. near-threshold voltage $\rightarrow$ nominal voltage). This is because those reported designs adopt dynamic-logic style, where the cross-coupled inverters (in the integrated-latch) are not functional-robust at sub-threshold voltage. Consider now the operational robustness of our SAHB adder against $V_{DD}$ variations for an in-situ self-adaptive $V_{DD}$ system [37] where $V_{DD}$ is automatically adjusted such the minimum $V_{DD}$ voltage is applied. The top and bottom traces of the Fig. 5.11 (b) respectively depict the real-time varying $V_{DD}$ (from 1.4V to 0.3V) and a generated output. It can be appreciated that even when $V_{DD}$ is varied widely, the associated operation is uninterrupted and error-free. Put simply, circuits embodying our SAHB cell design approach are advantageous for power/speed trade-off through voltage scaling with low transition/recovery time [55].
Fig. 5.11: Signal Waveforms of the SAHB Adder Operations: (a) Sub-threshold ($V_{DD} \sim 0.25V$) and (b) DVS ($V_{DD}$ from 1.4V to 0.3V)

5.3.3 Fabricated IC Measurement Results

Based on 20 IC chip measurement, Fig. 5.12 depicts normalized energy/operation ($E_{per}$), test-speed ($1/t$), $E_{per}t$ and $E_{per}t^2$ of the SAHB adder on different supply voltages ($V_{DD} = 0.3V$ to 1.4V; $V_{DD,L} = 0.3V$) and at different temperatures (-40°C, 0°C, 27°C and 100°C). The results are normalized with the readings taken @1V, 27°C where $E_{per} = 76.5ps$, $1/t = 125MHz$, $E_{per}t = 610x10^{-21}$ J.s and $E_{per}t^2 = 4.86x10^{-27}$ J.s$^2$ respectively. The test-speed includes the test structure circuit overheads (for loading/synchronizing inputs) to test operations at different temperatures/voltages. The actual throughput of the SAHB adder is
expected to feature 10× faster than the test-speed. The test jig is placed into a temperature chamber (model: Binder MK53) and the chamber temperature is carefully controlled and is stable for at least one hour before the measurement readings are taken.
Fig. 5.12: Normalized Figure-of-merits of 64-bit Pipeline Adder: (a) Energy/operation ($E_{per}$), (b) Test-speed (1/t), (c) $E_{per}.t$ and (d) $E_{per}.t^2$; normalized to the results taken @ 1V, 27ºC
From Fig. 5.12 (a), we remark the following for the $E_{\text{per}}$ plot. First, as expected, $E_{\text{per}}$ reduces as the $V_{\text{DD}}$ reduces from 1.4V until to the minimum $E_{\text{per}}$ voltage point (0.3V, within the sub-threshold voltage regime). Second, from the sub-threshold to nominal voltage regimes (0.3V to 1.4V), $E_{\text{per}}$ increases when the temperature increases (due to increase in the static power consumption).

From Fig. 5.12 (b), we remark the following for the $1/t$ plot. First, as expected, the $1/t$ reduces as the $V_{\text{DD}}$ reduces. Second, within the near-threshold voltage to nominal voltage regimes (0.7V ↔ 1.4V), the $1/t$ reduces when the temperature increases (due to the slower electron mobilities at higher temperature). However, within the sub-threshold voltage to near-threshold voltage regimes (0.3V ↔ 0.5V), the $1/t$ conversely increases when the temperature increases (due to the sub-threshold operation effects [15]).

From Fig. 5.12 (c), we remark the following for the $E_{\text{per}}t$ plot. First, $E_{\text{per}}t$ reduces as the $V_{\text{DD}}$ reduces from 1.4V until to a minimum $E_{\text{per}}t$ voltage point (0.6V, within the near-threshold voltage regime), and further reducing $V_{\text{DD}}$ from that point causes $E_{\text{per}}t$ to be higher than the minimum $E_{\text{per}}t$. Perhaps, it is not unsurprising that there are some existing research efforts for near-threshold operation to strike a balance between low power and high speed operation [64]. Second, minimum $E_{\text{per}}t$ decreases when the temperature decreases. Third, $V_{\text{DD}}$ for the minimum $E_{\text{per}}t$ reduces when the temperature decreases. Fourth, within the near-threshold voltage to nominal voltage regimes (0.5V ↔ 1.4V), the $E_{\text{per}}t$ increases when the temperature increases. However, within the sub-threshold voltage to near-threshold
voltage regimes (0.3V ↔ 0.5V), the $E_{per\cdot t}$ conversely decreases when the temperature increases.

Lastly, from Fig. 5.12 (d), we remark the following for the $E_{per\cdot t^2}$ plot. First, $E_{per\cdot t^2}$ slightly reduces and remains relatively constant at the near-threshold voltage to nominal voltage regimes (0.6V ↔ 1.4V). However, it increases significantly in the sub-threshold voltage regime (0.3V ↔ 0.4V); this is expected as $t^2$ increases significantly. Second, within the near-threshold voltage to nominal voltage regimes (0.6V ↔ 1.4V), the $E_{per\cdot t^2}$ increases when the temperature increases. However, within the sub-threshold voltage to near-threshold voltage regimes (0.3V ↔ 0.6V), the $E_{per\cdot t^2}$ conversely decreases when the temperature increases.

For comparison, we further benchmark, on the basis of simulations, our SAHB adder against its PCHB and sync counterparts. The PCHB and sync designs are designed/simulated using the same process. The sync design is synthesized to its fastest possible speed. Fig. 5.13 depicts the normalized energy/operation versus the throughput of the three designs. The results are normalized to the readings of the SAHB adder @ 1GHz throughput. The throughput of the SAHB and PCHB designs are adjusted through voltage scaling from $V_{DD} = 1V$ to lower $V_{DD}$ until these designs fail. On the other hand, the throughput of the sync design is adjusted through frequency scaling @ $V_{DD} = 1V$; voltage scaling is not considered due to a need of timing matching.
From Fig. 5.13, we remark the following. First, the sync adder leverages on the clock timing assumption, hence would not be error-free if the clock timings were violated (e.g. due to PVT variations). As a result, the sync design often operates slower than warranted for accommodating PVT variations. In contrast, the SAHB and PCHB designs having extremely good operational robustness. The maximum throughputs of the SAHB and PCHB designs are 1.23GHz and 1.02GHz respectively. This is expected as both the SAHB and PCHB designs satisfy the QDI async protocol and require some delay (transition) overheads to acknowledge their operation sequence. Further, the transistors of the SAHB design (and PCHB design) are not sized for maximum speed but for low power dissipation.

Second, the sync design is less energy-efficient than the SAHB design. The high energy dissipation of the sync design is due to a large number of registers used (for high speed gate-level pipelining) and to some extent to the energy dissipated in
the high-speed clock buffers. Although it is may be argued that the sync design
could be re-designed to have a different architecture (e.g. having a block-level
pipeline) to reduce the energy dissipation (at the cost of having a slower
throughput), such analysis and other permutations are being investigated and will
be reported elsewhere. Third, at the fixed throughput rate of 1GHz, the sync and
PCHB designs dissipate 1.65× and 2.29× higher energy respectively than our
SAHB design. Fourth, of the SAHB and PCHB designs, SAHB is indeed more
energy- and speed-efficient than PCHB. Further, the PCHB design is also less area-
efficient, occupying 1.31× more transistors than our SAHB design.

For completeness, Table 5.4 tabulates a comparison of several reported 64-
bite adders. Although the comparison is somewhat contentious due to large
variations of the designs, architectures, pipelining and parameters therein, it is
nonetheless worthwhile to note that our SAHB adder described in this chapter is
robust, insensitive to PVT variations and energy-efficient.
### Table 5.4
**Comparison of Various 64-bit Adders**

<table>
<thead>
<tr>
<th>Design Approach</th>
<th>Async</th>
<th>Sync</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit Adder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAHB</td>
<td>P</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>PCHB</td>
<td>P</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>CMOS (nm)</td>
<td>65</td>
<td>65</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>65</td>
<td>180</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>65</td>
<td>180</td>
</tr>
<tr>
<td>V_{DD}(V)</td>
<td>1.0</td>
<td>1.0</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>2.5</td>
<td>2.5</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.1</td>
<td>1.0</td>
</tr>
<tr>
<td>Pipeline Structure</td>
<td>P</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>KS</td>
<td>KS</td>
<td>CLA</td>
</tr>
<tr>
<td></td>
<td>CLA</td>
<td>CLA</td>
<td>△</td>
</tr>
<tr>
<td></td>
<td>CLA</td>
<td>△</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLA</td>
<td>△</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLA</td>
<td>△</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLA</td>
<td>△</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLA</td>
<td>△</td>
<td></td>
</tr>
<tr>
<td>Logic Coding</td>
<td>QDI &amp;</td>
<td>QDI &amp;</td>
<td>DR</td>
</tr>
<tr>
<td></td>
<td>DR</td>
<td>DR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>QDI</td>
<td>QDI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&amp;</td>
<td>&amp;</td>
<td></td>
</tr>
<tr>
<td>Logic Family</td>
<td>SAHB logic</td>
<td>PCHB logic</td>
<td>STFB logic</td>
</tr>
<tr>
<td></td>
<td>Static logic</td>
<td>Domino logic</td>
<td>Race logic</td>
</tr>
<tr>
<td></td>
<td>Domino logic</td>
<td>Domino logic</td>
<td>Domino logic</td>
</tr>
<tr>
<td></td>
<td>Dynamic/domino logic</td>
<td>Domino logic</td>
<td>Boosted differential logic</td>
</tr>
<tr>
<td>PVT Immunity</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Good</td>
</tr>
<tr>
<td></td>
<td>Good</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td></td>
<td>Poor</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td></td>
<td>Poor</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td>Throughput (GHz)</td>
<td>1.23</td>
<td>1.02</td>
<td>1.45</td>
</tr>
<tr>
<td></td>
<td>4.00</td>
<td>2.27</td>
<td>1.10</td>
</tr>
<tr>
<td></td>
<td>1.28</td>
<td>4.16</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td>5.26</td>
<td>6.26</td>
<td>5.26</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>0.06</td>
<td>0.07</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>NA</td>
<td>0.36</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>0.03</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Async E*\text{t} (x10⁻²)</td>
<td>2.7</td>
<td>5.7</td>
<td>13.1</td>
</tr>
<tr>
<td></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>P – Pipeline; *NP – Non-pipeline; ^CS – Carry-Select; △CLA – Carry Look-ahead; RC – Ripple Carry; #SR – Single-rail; *DR – Dual-rail; aBased on Simulation Results</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 5.4 Summary

We propose a high speed SAHB QDI cell template with emphasis on high operational robustness and yet low power dissipation. On the basis of six library cells (i.e., buffer, 2-input AND/NAND, 2-input OR/NOR, 2-input XOR/XNOR and 2-input MUX/IMUX and 3-input AO/AOI) @ 1V, 65nm CMOS process, the proposed SAHB approach outperforms the reported competing PCHB approach; the PCHB library cells, on average, dissipate 2.8× more power, suffer 1.27× slower delay and occupy 1.06× larger area than that of the SAHB library cells.
We further describe a 64-bit KS pipeline adder embodying the proposed SAHB approach for a power management application. Our SAHB pipeline adder is experimentally verified to be operational robust within wide $V_{DD}$ voltage range (0.3V to 1.4V) and wide temperature range (-40°C to 100°C). When benchmarked against its competing async PCHB and sync equivalents (@ 1GHz throughput), our SAHB pipeline adder is more energy-efficient; the PCHB and sync counterpart dissipates 2.29× and 1.65× higher energy respectively. The PCHB pipeline adder also occupies 1.31× more transistor-count. When further benchmarked against the reported 64-bit adders [65], [66], [67], [68], [69], [70], our SAHB adder is robust, insensitive to PVT variations and energy-efficient.
Chapter 6: Proposed High-Robustness Async Network-on-Chip (ANoC) based on SAHB

6.1 Introduction

Designing a SoC [71] realized by NoC [72]-[78] is a promising research area applicable to many applications, including defense and security applications. The basic premises for NoC-based SoCs include good scalability/routability (due to duplication of µP and NoC), improved speed (due to parallel processing) and high energy-efficiency (due to effective routing and computations). For our security application, we illustrate in Fig. 6.1 our targeted NoC-based SoC, comprising 9 processing units which are connected through their respective NoC routers. Among the 9 processing units, 6 are programmable microprocessors (µP0 to µP5) that can be programmed for processing various algorithms. The remaining processing units are a crypto-µP (to encrypt/decrypt data) [79], a secured non-imprinting SRAM (to store sensitive data) [80] and a globally shared SRAM. Each processing unit embodies its individual power supply. The supply voltage can be adjusted for DVS.

Fig. 6.1: Proposed NoC-based SoC for Our Security Application
To design such low power yet reliable NoC-based SoC, one of the main issues is the design of NoC routers (and the associated power management controller). To-date, there are many NoC routers reported, including those based on fully sync and those in part based on async. The design of sync NoC routers, however, becomes increasingly more challenging due to the timing issues for operation correctness. The timing issues are further compounded in view of wider PVT variations in deep-submicron fabrication processes. Conversely, the design of async NoC (ANoC) routers [72], [73], [74], [75], [76], [77], [78] has increasingly become attractive in accommodating timing and data synchronization issues because their operations are essentially self-timed. Their self-timed operation can often be utilized for low power dissipation such as with the techniques associated to DVS and/or leakage power reduction [78].

Of many reported ANoC routers, they can be generally realized using the bundled-data approach or the QDI approach. The ANoC routers using the bundled-data approach include MANGO [73], SCAFFI [74], ASPIN [75] and QNoC [76]. These bundled-data ANoC routers are undesirably relied on timing assumptions, and are somewhat not robust towards PVT variations. In contrast, the ANoC routers using the QDI approach include ALPIN [72], QoS [77] and FAUST [78]. These QDI ANoC routers were reported to be operationally robust and power-efficient. Of particular interest, the ALPIN ANoC router adopts quad-rail data encoding where each rail matches one direction for data propagation, hence simplifying its implementation. However, the reported ALPIN ANoC router [72] adopts the conventional QDI WCHB realization approach which requires a huge amount of
Muller C-element circuit overheads, hence somewhat compromising the energy-efficiency of the overall NoC router.

In this chapter, we propose an 18-bit ANoC router with 5 dual-ports by emphases on high robustness and low energy dissipation for our NoC-based SoC security application. Our proposed ANoC router features several significances. First, we leverage on our SAHB realization approach to design novel quad-rail async cells structurally appropriate for the realization of our proposed ANoC router. Second, our proposed ANoC satisfy the QDI, making it robust in accommodating the timing and PVT variations. Third, our proposed ANoC router features full DVS (nominal voltage ↔ near-threshold voltage ↔ sub-threshold); reported ANoC routers had only limited DVS (e.g. nominal voltage ↔ near-threshold voltage). Viewed differently, our proposed ANoC router will be more advantageous for low power management in the NoC-based SoC. Lastly, we apply the distributed-based XY-algorithm routing [74] for our ANoC router, minimizing the routing overhead to only 4-bit header flit for routing up to 4×4 cluster.

6.2 ANoC Interface Structure

In the 2D-mesh NoC topology, an NoC router, as depicted in Fig. 6.1 earlier, comprises five interfaces, each having an input port and an output port. Four interfaces are used to transfer data to/from four directions (i.e. north (N), east (E), south (S) and west (W)) between two neighboring NoC routers/IOs, and the last interface to transfer data between the NoC router and its respective processing unit.
Fig. 6.2 depicts the block diagram of one router interface where the input and output ports are depicted in (a) and (b) respectively. In Fig. 6.2, the bold dark lines represent the data packet, the straight lines represent the control signals and the dotted lines represent the acknowledge signals. In the input and output ports, every data packet and control signal is accompanied by an acknowledge signal for one complete async operation.
Fig. 6.2: ANoC Router Interface (a) Input port and (b) Output port
The input port consists of a virtual channel (VC) Demux and two identical input VCs (i.e. IVC0 and IVC1). The VC Demux first selects either IVC0 or IVC1 for propagating the data packet, depending on the select signals \textit{Send0} and \textit{Send1}. The select signals are triggered by the traffic conditions in the NoC router according to the routing mechanism [76]. In each input VC, the Path Generator decodes the destination information (in the header data packet) to produce the routing direction. Next, the Signal Packet triggers $Pkt_i$ ($i = \text{N, E, S or W directions}$) and allows the data packet to propagate to the designated output port (one of the 4 directions). As the control signals are required to transmit over two pipeline stages (i.e. the Path Generator and the Signal Packet), two Buffers, forming another two pipeline stages, are used to synchronize the data packet. Once the designated output port processes the sent data packet, $Acc_i$ is triggered and the Gather Accept combines $Acc_i$ from all directions to generate $Acc0/Acc1$ to acknowledge handshake operation.

The output port consists of a Signal Accept, a VC Arbiter, a VC switch and two identical output VCs (i.e. OVC0 and OVC1). In each output VC, the Direction Switch selects the prioritized data packet sent from the input port. The Direction Arbiter runs the arbitration among the various incoming $Pkt_i$ and decides the direction for prioritized data packet. This is to allow the first comer to continuously occupy the traffic channel such that the late comers have to wait until all flits of the first comer are all propagated. The Buffers synchronize and propagate the data packet and the control signal.
The Signal Accept collects the output control signals from the Direction Switch and triggers Acc to be sent to the input port. The VC Arbiter runs the arbitration among OVC0 and OVC1, and generates the VC prioritizing signal. The VC Switch selects the prioritized data packet to be sent out to the succeeding NoC router, and triggers Send0/Send1, depending on the input control signals Acc0/Acc1.

6.3 SAHB Quad-rail Cell Design

The functions [76] described in the input and output ports can be easily realized by buffering, multiplexing and de-multiplexing. Hence, for our async ANoC router, the most critical cells are the buffers, multiplexers and de-multiplexers. We leverage on our SAHB realization approach, and design async cells for quad-rail encoding; the original SAHB cells are based on the dual-rail encoding. The SAHB quad-rail cells are designed at transistor-level. For illustration, Fig. 6.3 depicts the SAHB quad-rail buffer cell, which comprises 4 functional blocks, two control-logic-cum-sense-amplifier (CLSA) blocks and a completion detection. The functional blocks take in a set of the quad-rail input signals (L.0 to L.3) and generate a set of quad-rail output signals (R.0 to R.3); the acknowledgement signal from the succeeding pipeline (Rack) is integrated for on/off operation. The CLSA blocks control the sequence of async operation, and help amplifying the output and thereafter latching the output. The completion detection generates the acknowledgement signal to the preceding pipeline (Lack) from the quad-rail output signals through the NAND gate. The supply voltages

117
$V_{DDA}$ (for functional blocks) and $V_{DD}$ (for remaining blocks) can be different. For proper operation, $V_{DD} \geq V_{DDA}$.

![Diagram of proposed SAHB Buffer Cell]

Fig. 6.3: Proposed SAHB Buffer Cell

Our proposed quad-rail SAHB cells are designed to be QDI, hence inherently accommodating timing and PVT variations. For ease of comparison, Fig. 6.4 depicts the reported async quad-rail QDI WCHB cell used in the ALPIN router [72]. From operational modality viewpoint, the SAHB and WCHB quad-rail cells are functional equivalent. However, the reported WCHB realization approach is not efficient in terms of speed and power since it utilizes high-overhead Muller C-element circuits to validate the signal completion. The number of Muller C-element circuits increase significantly when the number of input sets $\geq 2$ and thus further worsen the performance.
Table 6.1 compares the three most critical cells embodying SAHB and WCHB. The results are normalized to the cells embodying SAHB. The WCHB cells, on average, dissipate 1.7× more power, suffer 1.3× longer delay and occupy 1.15× more transistor-count than the SAHB cells.

<table>
<thead>
<tr>
<th></th>
<th>Power @ 1V, 1GHz</th>
<th>Delay @ 1V</th>
<th>Transistor-Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SAHB</td>
<td>WCHB</td>
<td>SAHB</td>
</tr>
<tr>
<td>Buffer</td>
<td>1x (9.2µW)</td>
<td>1.21x</td>
<td>1x (174ps)</td>
</tr>
<tr>
<td>1:2 De-Mux</td>
<td>1x (10.6µW)</td>
<td>2.31x</td>
<td>1x (235ps)</td>
</tr>
<tr>
<td>4:1 Mux</td>
<td>1x (25.8µW)</td>
<td>1.59x</td>
<td>1x (420ps)</td>
</tr>
<tr>
<td>Average</td>
<td>1x</td>
<td>1.70x</td>
<td>1x</td>
</tr>
</tbody>
</table>

6.4 Design Implementation

The proposed ANoC router is implemented based on the 65nm standard-threshold-voltage CMOS process ($V_t = 0.38V$). We adopt a full-custom approach to implement the ANoC router. The layouts of all library cells, including the SAHB cells and standard single-rail library cells, are first hand-drafted using the Cadence
Virtuoso Layout tool. For SAHB cells, many building blocks (i.e. function block, CLSA and completion detection blocks) are similar, hence we standardize those similar building blocks for layout implementation. Some building blocks of the SAHB buffer cell are depicted in Fig. 6.5. The extracted library-exchange-format (LEF) files of all library cells are generated using the Cadence Abstract Generator tool. From the Verilog files, the SAHB ANoC router layout is placed and routed using the Cadence First Encounter tool. Eventually, the SAHB ANoC router layout is simulated and verified using the Synopsys Nanosim tool. The layout view and the microphotograph of the proposed ANoC router are shown in Fig. 6.6(a) and (b) respectively.

![Fig. 6.5: Layout View of the Proposed SAHB Buffer Cell](image-url)
6.5 Measurement Results and Comparison

Fig. 6.7 (a) depicts the measurement result of the proposed ANoC router @ the sub-threshold voltage of 0.3V. From the operation perspective, the ANoC router takes 840ns to complete 8 operations with 7.5µW. Hence the energy dissipation per operation per bit for the ANoC router is about 44fJ/bit. To demonstrate full-scale DVS, Fig. 6.7(b) depicts the operation of the proposed ANoC router under supply voltage scaling within the nominal (1.2V) to sub-threshold (0.3V) regions. We show that our ANoC router is functionally robust against the voltage variation and hence is appropriate for applications which can be easily trade-off between high speed or low power operations through DVS.
Fig. 6.7: Measurement Results (a) $V_{DD} = 0.3\text{V}$ and (b) DVS ($0.3\text{V} \leftrightarrow 1.2\text{V}$)

Fig. 6.8 depicts the measured energy dissipation per operation per bit of the proposed ANoC router when $V_{DD}$ is scaled from 1.2V to 0.3V. Its lowest energy point occurs at 0.4V. Viewed differently, when the system throughput is not critical, we can enable DVS to save up to 89% of energy dissipation.
Table 6.2 tabulates the measurement result of the proposed ANoC router and its benchmarking against the reported WCHB counterpart. Particularly, our design occupies 21% smaller chip area than the reported counterpart when they are designed for 18-bit operation. Furthermore, our design is 41% more energy-efficient than the reported counterpart @1.2V. This is mainly due to the lesser spurious switchings and lower leakage energy in the proposed novel SAHB quad-rail realization approach. In addition, the lower routing overheads in the proposed ANoC router also minimize the overall energy dissipation.

<table>
<thead>
<tr>
<th>Table 6.2</th>
<th>MEASUREMENT COMPARISON OF THE PROPOSED AND REPORTED ANOCs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NoC topology</strong></td>
<td>Proposed</td>
</tr>
<tr>
<td>2D mesh</td>
<td></td>
</tr>
<tr>
<td><strong>Router port</strong></td>
<td>5 dual-ports router</td>
</tr>
<tr>
<td><strong>Flowing protocol</strong></td>
<td>Async handshake four-phase/quad-rail QDI</td>
</tr>
<tr>
<td><strong>Flit size</strong></td>
<td>18-bit</td>
</tr>
<tr>
<td><strong>Library cell</strong></td>
<td>SAHB</td>
</tr>
<tr>
<td><strong>Area (mm²)</strong></td>
<td>0.105 (-21%)</td>
</tr>
<tr>
<td><strong>Throughput (MHz) @ 1.2V</strong></td>
<td>258</td>
</tr>
<tr>
<td><strong>Throughput (MHz) @ 0.3V</strong></td>
<td>9.5</td>
</tr>
<tr>
<td><strong>Energy/op*bit (fJ/bit) @ 1.2V</strong></td>
<td>390 (-41%)</td>
</tr>
<tr>
<td><strong>Energy/op*bit (fJ/bit) @ 0.3V</strong></td>
<td>44</td>
</tr>
</tbody>
</table>

*results after scaling to 18-bit operation based on transistor-count ratio
6.5 **Summary**

The proposed ANoC router is implemented based on the 65nm CMOS process. We benchmark it against the reported ANoC router embodying WCHB. We show that our proposed ANoC is 41% more energy-efficient and 21% more area-efficient than its reported counterpart. The prototype of ANoC router occupies 0.105mm² and can operate as low voltage as 0.3V. At $V_{DD}=0.3V$, it dissipates 44fJ per bit and operates 105ns per flit.
Chapter 7: Conclusions and Recommendations for Future Work

7.1 Conclusions

This thesis pertains to the three proposed async QDI templates, namely low power iSAPTL, sub-threshold ASVHB and high-speed SAHB for high robustness, low power and high speed applications. Besides, an ANoC based on SAHB template has been further proposed for multi-core SoC platform, which is targeted for highly secured cryptography application.

First, we have designed an energy-delay efficient async 16×16-bit pipeline multiplier based on our proposed iSAPTL circuit template, which consists of an optimized Decision-Making Muller C-element and optimized Pass Transistor Stack. In the library cell level, the proposed design has reduced the number of transistors, hence the number of transistor switchings, as well as shortened transistor series of the pull-up and pull-down networks in the Output Sense-Amplifier for output evaluation. Based on the simulation results, our proposed 16×16-bit multiplier features on average ~31% shorter delay, ~21% lower energy/operation, thus resulting in a total of 46% better energy-delay product. Our design also features 16% lesser transistor-count.

Second, we have proposed a novel async QDI ASVHB realization approach for the sub-threshold low power operation. The comparison is benchmarked against the competitive reported WCHB and PCHB realization approaches. Our ASVHB realization approach is shown to be the best from the circuit and pipeline
perspective. The ASVHB library cells, on average, features ~52% and ~47% lesser transistors than the reported WCHB and PCHB library cells, whereas the ASVHB pipeline, on average, features ~44% and ~33% lesser transitions per cycle than the reported WCHB and PCHB pipelines respectively. We have further implemented our ASVHB realization approach in a 32-bit ALU and compared it to the WCHB and PCHB counterparts. Our design features 0.092mm² chip area, i.e. ~41% and ~29% lesser transistors respectively than the WCHB and PCHB counterparts. At $V_{DD} = 0.2V$, our design is the best in overall performance. In terms of the energy dissipation, our design dissipates ~41% and ~62% lower energy respectively than the WCHB and PCHB counterparts. In terms of the data throughput, our design is ~5% and ~37% faster respectively than the WCHB and PCHB counterparts. Finally, when compared to the various reported ALUs, our proposed design has demonstrated the competitive advantages in terms of energy-efficiency and robustness towards PVT variations.

Third, we have proposed a novel SAHB realization approach with emphases on high operational robustness, high speed and low energy dissipation. These attributes are collectively achieved by several novel circuit designs, including the async QDI approach, cross-coupled latch with a positive feedback mechanism in the SA block, reduced switching nodes in the evaluation and SA blocks, minimum sizing of the NMOS pull-up network in the evaluation block, and the static logic operation. The basic library cells embodying SAHB have been shown to feature higher speed, low energy dissipation and lower transistor-count than those embodying the reported competing PCHB. The 64-bit SAHB adder has been
prototyped for a power management application with full-range DVS. To
demonstrate its energy-efficiency, the proposed SAHB adder has been
benchmarking against its competing PCHB and sync equivalents. For 1GHz
throughput and at nominal $V_{DD}$ of 1.2V, the proposed SAHB adder features
simultaneously ~56% lower energy and ~24% lower transistor-count than the
PCHB approach. When benchmarked against the sync equivalents, the proposed
SAHB adder dissipates ~39% lower energy at 1GHz throughput but at the expense
of ~2× more transistor-count.

Fourth, we have proposed an energy-efficient high robustness ANoC router
based on QDI high-speed low-power SAHB approach. We have designed and
implemented the proposed ANoC router (@ 65nm CMOS), and benchmarked it
against the reported ANoC featuring WCHB approach. Although both designs are
highly robust against PVT variations, our design is ~41% more energy efficient and
~21% smaller area than the WCHB approach.

7.2 Recommendations for Future Work

7.2.1 Async DVS Microprocessor

The proposed low power iSAPTL, sub-threshold ASVHB and high-speed
SAHB QDI gate-level approaches can be applied to realize an async microprocessor
with DVS power management, as depicted in Fig. 7.1. This is because the proposed
approaches satisfy the QDI feature, hence is highly robust towards PVT variations.
The supply voltage is adjustable from nominal voltage (1V), to near-threshold
voltage (0.47V) and further to sub-threshold voltage (<0.38V). This mechanism is
known as full-range DVS, which dynamically reduce/increase the voltage supply for the microprocessor, switching between low power slow speed non-critical operation and high power fast speed critical operation.

Specifically, the proposed iSAPTL cell template can be used to construct the microprocessor parts targeted for power-saving operation. The proposed ASVHB cell template can be applied in the microprocessor parts targeted for low voltage low power operation. The proposed SAHB Adder can be applied in designing the counter block for power management application.

Fig. 7.1: Async DVS Microprocessor
7.2.2 ANoC-based Multi-core Platform

Our energy-efficient high robustness ANoC router based on QDI high-speed SAHB approach can be applied to construct an ANoC-based multi-core platform for highly secured cryptography application. Fig. 7.2 depicts an example of 9-core platform, which connects nine processing cores to each other through the proposed ANoC. The recommended multi-core platform is scalable, routable, programmable, and is able to run the parallel computation. Hence, the platform can be designed to feature distribution of encryption tasks in each core with individual voltage/frequency to randomize the power dissipation, increasing the difficulty to leak the information under side channel power analysis attack.

Fig. 7.2: 9-core Platform with Multiple Voltages/Frequencies
Bibliography


Appendix I: Improved SAPTL2

We present an improved SAPTL2 approach [57] for async dual-rail circuits with emphases on higher speed and lower power operation (compared to the reported SAPTL approach [52]). Fig. A1 depicts our SAPTL2 circuit template that is in part modified based on the reported approach [52]. Similar to the latter, our SAPTL2 template comprises a Stack Driver, a NMOS Pass Transistor Stack, an Output Sense Amplifier, and a Completion Circuit. Nonetheless, our SAPTL2 approach has the following differences.

First, the Stack Driver of our approach is an NAND-based circuit (instead of an OR-based circuit) and features 8 transistors (i.e. 2 transistors lesser than the reported Stack Driver). Second, we implement the decision-making Muller C-element by using the fully static-logic circuit (as opposed to the reported design which is embodying a pass logic controller); see Fig. A2 for the schematic of the decision-making Muller C-element. The removal of the pass-logic controller...
enhances the speed especially when the NMOS Pass Transistor Stack has a long series transistors. Third, the charged restoring pull-up keeper is shared by two decision-making Muller C-elements (as opposed to a separate keeper in each Muller C-element in the reported design). Fourth, a pull-down keeper is included to conditionally maintain the floating node (either S.T or S.F during evaluation). The pull-down keeper minimizes the leakage current (if any) as no floating node exists. Finally, the Completion Circuit is an NOR gate (as opposed an AND gate in the reported design), saving an inverter.

![Decision-Making Muller C-element Circuit in SAPTL2](image)

Fig. A2: Decision-Making Muller C-element Circuit in SAPTL2

The operation concept of the SAPTL2 is indeed the same as the reported SAPTL but our design reduces floating nodes problem. The idea is as follows: once the dual-rail outputs are evaluated to be valid, the pull-up keeper will restore the incomplete voltage swing node (either S.T or S.F) and the pull-down keeper will maintain the logic ‘0’ of the other node (i.e. no floating node exists). Although the floating node still temporarily exists when the pull-down keeper is turned off (when Rreq is asserted), Lack (= Rreq) will acknowledge the preceding pipeline stage to
reset Data Input which eventually will reset both S.T or S.F to ‘0’ (i.e. no floating node exists again). The complete signal transition graph of the SAPTL2 template is shown in Fig. A3.

Fig. A3: SAPTL2 Signal Transition Graph