Design methodologies for robust and low-overhead asynchronous quasi-delay-insensitive digital systems

Zhou, Rong

2015

Zhou, R. (2015). Design methodologies for robust and low-overhead asynchronous quasi-delay-insensitive digital systems. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/65411>

<https://doi.org/10.32657/10356/65411>

NANYANG
TECHNOLOGICAL
UNIVERSITY

DESIGN METHODOLOGIES
FOR
ROBUST AND LOW-OVERHEAD
ASYNCHRONOUS QUASI-DELAY-INSENSITIVE
DIGITAL SYSTEMS

ZHOU RONG

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2015

**DESIGN METHODOLOGIES
FOR
ROBUST AND LOW-OVERHEAD
ASYNCHRONOUS QUASI-DELAY-INSENSITIVE
DIGITAL SYSTEMS**

ZHOU RONG

School of Electrical and Electronic Engineering

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

2015

Acknowledgements

My sincerest gratitude goes to my supervisor Associate Prof. Gwee Bah Hwee and my senior colleague Dr. Chong Kwen Siong for their patient and professional guidance and encouragements throughout my doctoral studies.

I would like to thank Associate Prof. Joseph Chang and my colleagues Ne Kyaw Zwa Lwin, Sun Yin, Ho Weng Geng and Lin Tong for their kind help and support in my doctoral studies.

I would like to thank my family for their patience and support throughout my doctorate study.

Contents

Acknowledgements	I
Contents	II
List of Publications	V
Abstract.....	VI
List of Figures.....	IX
List of Tables.....	XII
Abbreviations	XIII
1. Introduction	1
1.1 Motivation	1
1.2 Objectives.....	4
1.3 Contributions of the Thesis	5
1.4 Organization of the Thesis	7
2. Literature Review.....	10
2.1 Asynchronous Communication Channel	10
2.2 Classifications of Asynchronous Digital Systems	13
2.3 Advantages and Limitations of Asynchronous Digital Systems	14
2.4 Synthesis Methods for Asynchronous Digital Systems	19
2.4.1 Syntax-directed Translation.....	19
2.4.2 Logic Synthesis	23
2.4.3 Desynchronization.....	27
2.4.4 Null Convention Logic.....	28
2.5 Design Methodologies for Quasi-Delay-Insensitive Data Path	29
2.5.1 Non-input-complete Microcell with Completion Detection Circuit	30
2.5.2 Input-complete Microcell.....	32

2.6 Dynamic Voltage Scaling Techniques.....	33
2.6.1 Conventional No-error Tolerable Dynamic Voltage Scaling.....	33
2.6.2 Razor-based Error Tolerable Dynamic Voltage Scaling.....	35
3. Proposed Synthesis Tool for Asynchronous Quasi-Delay-Insensitive Digital Systems.....	37
3.1 Introduction.....	37
3.2 Four-phase Dual-rail Channel for Proposed Synthesis Tool.....	38
3.3 Quasi-Delay-Insensitive Pipeline Structure.....	40
3.4 Specification Language - Verilog Hardware Description Language.....	43
3.5 Design Compilation.....	46
3.5.1 Design Flow of Proposed Quasi-Delay-Insensitive Synthesis Tool.....	46
3.5.2 Data Flow Extraction.....	48
3.5.3 Pipeline Partition.....	52
3.5.4 Insert Null Pipeline Stages.....	53
3.5.5 Low Overhead Initialization of Linear Pipelines.....	54
3.5.6 Compiled Circuit Patterns.....	55
3.6 Finite Impulse Response Filter Design Example.....	60
4. Proposed Quasi-Delay-Insensitive Data Path Optimization using Microcell-Interleaving Genetic Algorithm.....	64
4.1 Introduction.....	64
4.2 Proposed Microcell-Interleaving Approach.....	67
4.3 Performance Profile Analysis for the Proposed Microcell-Interleaving Approach.....	73
4.3.1 Power Dissipation.....	73
4.3.2 Transistor-Count.....	79
4.3.3 Delay.....	81
4.4 Proposed Microcell-Interleaving Genetic Algorithm.....	83
4.4.1 Genetic Coding of Combinational Circuit Netlist.....	86

4.4.2	Fitness Function Formulation.....	87
4.5	Experimental Results	89
4.5.1	The Performance Profile for C499	90
4.5.2	Comparison on ISCAS-85 Benchmark Circuits.....	94
4.5.3	Comparisons of the Proposed Microcell-Interleaving Genetic Algorithm with Other Algorithms	96
5.	Proposed Dynamic Voltage Scaling Technique on Asynchronous Quasi-Delay-Insensitive Digital Systems.....	99
5.1	Introduction.....	99
5.2	Proposed Dynamic Voltage Scaling Technique.....	101
5.3	Design Details of the Proposed Dynamic Voltage Scaling Pipeline	104
5.3.1	On-chip Pipeline-granular Power Supply Unit with Three Voltage Scales	104
5.3.2	Level Shifter.....	110
5.4	Operation Mode Investigations.....	111
5.4.1	Low Power Operation Mode.....	112
5.4.2	Single Charge Up of Current-preceding Mode	113
5.4.3	Single Charge Up of Current-successive Mode	115
5.4.4	Double Charge Up Mode	117
5.5	Power Supply Unit Size Determination	118
5.6	Proposed FIR Filter with Async QDI DVS Technique	119
6.	Conclusions and Recommendations for Future Work.....	126
6.1	Conclusions	126
6.2	Recommendations for Future Work	128
6.2.1	Co-synthesize Synchronous and Asynchronous Systems from One Specification.....	129
6.2.2	Dynamic Voltage Scaling Technique on Sync-async Systems.....	131
	Bibliography.....	132

List of Publications

R. Zhou, K. S. Chong, B. H. Gwee, and J. S. Chang, "A Low Overhead Quasi-Delay-Insensitive (QDI) Asynchronous Data Path Synthesis Based on Microcell-Interleaving Genetic Algorithm (MIGA)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 7, pp. 989–1002, July 2014.

R. Zhou, K.S. Chong, B. H. Gwee, and J. S. Chang, " A Single VDD Half-Clock-Tolerant Fine-Grained Dynamic Voltage Scaling Pipeline," *IEEE International Symposium on Circuits and Systems*, Lisbon, Portugal, May 2015.

R. Zhou, K.S. Chong, B. H. Gwee, and J. S. Chang, "Synthesis of Asynchronous QDI Circuits using Synchronous Coding Specifications," *IEEE International Symposium on Circuits and Systems*, May 2014, pp. 153-156.

R. Zhou, K.S. Chong, B. H. Gwee, and J. S. Chang, "Quasi-delay-insensitive compiler: Automatic synthesis of asynchronous circuits from Verilog specifications," *IEEE International Midwest Symposium on Circuits and Systems*, Aug 2011, pp. 1-4.

R. Zhou, K.S. Chong, B. H. Gwee, and J. S. Chang, "A Fine-Grained Asynchronous Quasi-Delay-Insensitive Dynamic Voltage Scaling Pipeline," *To be submitted to IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

Abstract

In the past decades of electronic circuit designs, the synchronous-logic (sync) is the main de-facto design approach for digital systems. This is largely due to its synchronization to a global clock signal (or its variants thereof) which simplifies the data transfer. Nevertheless, designing digital circuits and systems based on sync approach is becoming more challenging due to the increase in process, voltage and temperature (PVT) variations in the deep submicron fabrication process. An alternative design approach which is based on the asynchronous-logic (async), is potentially more robust against the PVT variations, lower power dissipation and higher speed due to the absence of the global clock signal. In async design approach, the Quasi-Delay-Insensitive (QDI) circuit is probably the most practical and robust design methodology. This thesis pertains to the investigation of high robustness, low design effort and low overhead async QDI digital design. Three contributions have been made in this thesis as follows.

First, a synthesis tool for designing the async QDI systems based on sync specifications is proposed. The proposed synthesis tool has the following salient features. The coding specifications are constructed based on the standard Verilog HDL language, to seamlessly integrate into the standard synchronous circuit design flow, giving it full leverages to design async QDI circuits by using the commercial Electronic Design Automation tools. The targeted synthesized async QDI circuits are robust against the PVT variations. To verify the effectiveness of the proposed

synthesis tool, an 8-bit 8-tap async QDI Finite Impulse Response (FIR) filter based on IBM 130nm CMOS process is synthesized and implemented; it dissipates 3.7mW @ 5MHz sampling rate and features 0.74nJ per operation, whereas the reported async design methodology dissipates 1.13nJ per operation ($1.53\times$ energy per operation as that of the proposed methodology).

Second, a novel microcell-interleaving approach is proposed to reduce the overhead of async QDI data path synthesized by the proposed synthesis tool. The performance of the async QDI data path designed by the microcell-interleaving approach is evaluated in terms of power dissipation, transistor-count, circuit delay and the upper / lower boundaries of these performance profiles. A Microcell-Interleaving Genetic Algorithm (MIGA) is further proposed and implemented to stochastically optimize the power dissipation, transistor-count, and delay of the benchmarking circuits based on the proposed microcell-interleaving approach. The async QDI circuits optimized by the MIGA, on average, dissipate 349 μ W @50MHz, feature 6917 transistors. Comparatively, the reported async QDI methodologies dissipate 617 μ W@50MHz ($1.77\times$ power dissipation as that of the proposed methodology), feature 9684 transistors ($1.4\times$ transistor count as that of the proposed methodology).

Third, a pipeline-granular async QDI dynamic voltage scaling (DVS) technique with low overhead is proposed to maintain minimum power dissipation of async QDI pipelines to perform necessary operations while accommodating the PVT variations. In the proposed async QDI DVS technique, the completion of the circuit operations is automatically detected by completion detection (CD) circuits, without additional

timing margin assumption. The power supply unit (PSU) is integrated on-chip with three voltage scales, 0.5V for low-speed operations, 0.8V for medium-speed operations and 1.2V for high-speed operations. The PSU adaptively adjusts the supply voltage of each pipeline stage according to their corresponding work-loads, thereby maintaining minimum power dissipation. To validate the proposed async QDI DVS technique, an async QDI FIR filter is implemented. The results show that the async QDI multiplier pipeline stage with the proposed async QDI DVS technique features a low power dissipation of $89.9\mu\text{W}@50\text{MHz}$, whereas the async QDI multiplier pipeline stage with the conventional design methodology dissipates $613.1\mu\text{W}@50\text{MHz}$ ($6.82\times$ power dissipation as that of the proposed methodology).

In summary, this thesis has proposed a synthesis tool, a microcell-interleaving optimization, and a DVS technique for low overhead and robust async QDI circuits with low design effort.

List of Figures

Fig. 1.1 Process and threshold voltage variation comparison.....	2
Fig. 2.1 (a) Two-phase handshake protocol, (b) Four-phase handshake protocol	11
Fig. 2.2 Synchronous vs asynchronous, (a) Synchronous approach, (b) Asynchronous approach.....	15
Fig. 2.3 (a) Combinational circuit with non-input-complete microcells and CD circuits, (b) Combinational circuit with input-complete microcells	31
Fig. 3.1 A delay-insensitive channel using the four-phase dual-rail protocol.....	39
Fig. 3.2 (a) QDI pipeline stage, (b) Completion detection tree and structure of C-Muller, (c) Implementation of register.....	41
Fig. 3.3 Supported synthesizable Verilog constructs	44
Fig. 3.4 Overall design flow of the proposed synthesis method.....	47
Fig. 3.5 (a) Details of combinational logic generation, (b) Details of pipeline generation	48
Fig. 3.6 (a) Directed graph for Example 1, (b) Directed graph for Example 2.....	49
Fig. 3.7 Pseudo-code of pipeline partition algorithm	51
Fig. 3.8 (a) A synchronous pipeline, (b) Compiled asynchronous pipelines with inserted Null pipeline stages	53
Fig. 3.9 (a) Linear pipelines with inserted Null pipeline stages, (b) Low overhead linear pipeline with loading initial values.....	54
Fig. 3.10 The schematic of C-Muller with load function.....	55
Fig. 3.11 Synthesized pipeline of complete ‘if-else’ clause with conditional operations embedded inside the combinational circuit.....	56

Fig. 3.12 Synthesized pipeline of incomplete ‘if-else’ clause with conditional operations embedded inside the combinational circuit.....	57
Fig. 3.13 Synthesized pipelines of one fashion of ‘case’ clause	58
Fig. 3.14 (a) Implementation of guard controller, (b) Implementation of default value cell, (c) Source code of the generated guard combinational circuit.....	59
Fig. 3.15 Block diagram of synthesized 8-bit 8-tap async QDI FIR filter.....	61
Fig. 3.16 The layout view of the synthesized Async QDI FIR Filter	63
Fig. 4.1 (a) The circuit schematic of a DCVSL async dual-rail OR/NOR microcell and estimated overhead for CD circuit, (b) The circuit schematic of a DSI dual-rail OR/NOR microcell	66
Fig. 4.2 The proposed asynchronous pipeline by adopting microcell-interleaving approach with DCVSL and DSI microcells	69
Fig. 4.3 Microcell interconnection: (a) Initial interconnection, (b) Interleaving interconnections, (c) Interleaving connection of zero CD circuits, and (d) Interleaving connection without redundant wire checking	71
Fig. 4.4 Overall power dissipation profile of all microcell combinations satisfying QDI attribute	75
Fig. 4.5 Overall delay profile of all microcell combinations satisfying QDI attribute	80
Fig. 4.6 Pseudo-code of the proposed Microcell-Interleaving Genetic Algorithm.....	85
Fig. 4.7 Performance of benchmark circuit C499 based on our proposed microcell-interleaved approach and genetic algorithm (a) Power profile, (b) Transistor-count profile, (c) Delay profile, and (d) The number of CD circuits in the searching for the lowest power	93
Fig. 4.8 Pseudo-code of greedy algorithm and dynamic programming.....	97
Fig. 5.1 Block diagram of the proposed async QDI DVS pipeline.....	102
Fig. 5.2 Control logic and power transistors of PSU in the proposed Async QDI DVS technique.....	109

Fig. 5.3 Transistor level implementation of level shifter	111
Fig. 5.4 The timing diagrams of low power operation mode.....	113
Fig. 5.5 The timing diagrams of single charge up of current-preceding mode.....	114
Fig. 5.6 The timing diagrams of single charge up of current-successive mode.....	116
Fig. 5.7 The timing diagrams of double charge up mode	117
Fig. 5.8 Block diagram of 8-bit 8-tap async QDI FIR filter with proposed DVS technique.....	120
Fig. 5.9 Waveforms for the low power operation mode in Async QDI FIR filter multiplier pipeline stage	121
Fig. 5.10 Waveforms for the single charge current-preceding mode in Async QDI FIR filter multiplier pipeline stage	122
Fig. 5.11 Waveforms for the single charge current-successive mode in Async QDI FIR filter multiplier pipeline stage	123
Fig. 5.12 Waveforms for the double charge up mode in Async QDI FIR filter multiplier pipeline stage	124
Fig. 6.1 Design flow of Sync-async Co-synthesis	130

List of Tables

Table 2.1	Dual-rail coding and explanations	12
Table 2.2	Comparisons between the reported DVS techniques and the proposed DVS technique	36
Table 3.1	Overhead (power dissipation (P , mW), transistor-count (TC) and delay (t , ns)) breakdown for the synthesized Asynchronous QDI FIR Filter @ 1.2V, IBM 130nm CMOS	62
Table 3.2	The power dissipation (P), transistor-count (TC), delay (t) and a composite metric of $P \times TC \times t$ of the lowest power point for Asynchronous QDI FIR Filter @ 1.2V, IBM 130nm CMOS	62
Table 4.1	The power, transistor-count and delay characteristics of several dual-rail microcells based on DCVSL and DSI @ 1.2V, IBM 130nm CMOS	68
Table 4.2	Power dissipation (P), transistor-count (TC), delay (t) and composite metric of $P \times TC \times t$ of the lowest power points for various benchmarks @ 1.2V, IBM 130nm CMOS	95
Table 4.3	The lowest power points (μW @ 50MHz) comparisons between the Greedy Algorithm, Dynamic Programming and the proposed MIGA for five ISCAS85 benchmark circuits @ 1.2V, IBM 130nm CMOS; normalized to the proposed MIGA	98
Table 5.1	Voltage scaling truth table of current pipeline stage with respect to control signals of current and successive pipelines ('0' indicates Null data, '1' indicates Valid data).....	107
Table 5.2	Voltage scaling truth table of current pipeline stage with respect to control signals of current and preceding pipelines ('0' indicates Null data, '1' indicates Valid data)	107
Table 5.3	Power dissipation comparisons (μW @ 1.2V) between the four operation modes with the proposed async QDI DVS technique and the nominal voltage (1.2V) operation	124

Abbreviations

C	C-Muller
CD	Completion Detection
CL	Combinational Logic
DCD	Data Completion Detection
DCVSL	Differential Cascode Voltage Swing Logic
DIMS	Delay-Insensitive Minterm Synthesis
DSI	Dynamic Strong Indicating
DVS	Dynamic Voltage Scaling
EDA	Electronic Design Automation
FFT	Fast Fourier Transformation
FIR	Finite Impulse Response
HDL	Hardware Description Language
ITRS	International Technology Roadmap for Semiconductors
LS	Level Shifter
MIGA	Microcell-Interleaving Genetic Algorithm
NCL	Null Convention Logic
NoC	Network-on-Chip
NRST	Negative Reset
OA/OAI	Or-And/ Or-And Inverted
PSU	Power Supply Unit
PVT	Process, Voltage, Temperature
QDI	Quasi-Delay-Insensitive
SoC	System-on-Chip
TTM	Time to Market
VLSI	Very Large Scale Integration

1.

Introduction

1.1 Motivation

The rapid advancement of the electronic technology has resulted in numerous applications, including space projects [1, 2], defense projects [3-5], mobile/portable devices [6, 7], wireless-sensor-networks [8, 9], Internet-of-Things [10, 11], multi-core processors [12, 13], etc. In these applications, digital communication and processing capabilities are collectively integrated in Systems-on-Chip (SoC) or Systems-in-Package using the advanced nano-scaled fabrication processes.

In recent decades, electronic circuits and systems are designed mainly based on synchronous-logic (sync) approach. This is mainly due to the following reasons.

First, the sync approach features low design effort. The synchronization of sync approach is largely based on a global clock or its variants. The correct operation is ensured if the clock cycle (period) of the global clock has sufficiently large delay margin to complete all the current batch of operations before the next batch of operations commences. As the data synchronization is straight forward by simply

observing the clock edge, the sync design approach is well accepted for digital design in the integrated circuit (IC) design industry.

Second, the design flow of the sync approach is well-developed and widely accepted. This is due to its low design effort, numerous research, investment and development have been made to the improvement of the design flow for the sync approach. Many software/tools have been commercialized and are widely accepted in the industry, which has reduced the time-to-market (TTM) requirement. This, in return, has secured the domination of the sync design approach in the IC design industry; otherwise, large investment would be required to switch to another design approach.

However, as the minimum-feature size of semiconductor fabrication processes continues to scale downwards; the process, voltage and temperature (PVT) variations have become increasingly large. For example, the International Technology Roadmap for Semiconductors (ITRS) [14] has projected that the process variations of the 45nm process will increase from 11% in 2009 to 32% for the impending 9nm process in

Fig. 1.1 Process and threshold voltage variation comparison

2024, and the threshold voltage variations will increase even more - from 42% to 112% respectively, depicted in Fig. 1.1. Consequently, designing digital circuits based on the ubiquitous sync approach is becoming increasingly challenging, particularly if the potential of higher operating speed is to be exploited. The usual approaches to accommodate the PVT variations include specifying a very conservative clock rate (the clock rate is slower than warranted when the conditions are more benign), adopting timing-tolerant techniques, etc. Nevertheless, it can be argued, perhaps somewhat contentiously, that unconditional error-free operation for synchronous circuits may be compromised because a complete profile of the PVT variations is intractable.

An alternative approach for robust operation is the esoteric asynchronous-logic (async) approach [15-20], is based on handshake signals for communication or synchronization, which is in contrast of using the global clock signal in synchronous design approach. Inter/intra-die processing and wire delay variation push the difficulty of distribution of clock tree properly to a higher level, especially with the increase of clock frequency, the size of die and the scaling down of silicon processes in the SoC era. The ITRS has further projected that the data synchronization between processor cores would be increasing async, and the percentage of async signaling in the overall chip will grow up to ~50% in 2024.

Async systems have several attractive features over their sync counterpart, such as potential lower power dissipation, more robust against timing issues, average case performance, good modularity, etc. In view of these advantages, the async designs

have been an active research area in the recent years. In async community, the Quasi-Delay-Insensitive (QDI) design is arguably the most practical and robust async design methodology to accommodate the PVT variations. Thus, this thesis pertains to the investigation of async QDI design methodology with low design effort and low overhead of designing async digital circuits and systems.

1.2 Objectives

The main objective of this thesis pertains to the investigation of async QDI circuit and system designs in terms of low implementation effort, high robustness and low-overhead. Specifically, three areas are explored as follows:

- 1) To investigate and develop a synthesis tool for async QDI circuits based on the prevalent Verilog Hardware Description Language (HDL) specifications. In this regard, the proposed synthesis tool is to seamlessly integrate with the standard commercial Electronic Design Automation (EDA) circuit design flow to speed-up the design process and reduce the design effort in async QDI circuit design.
- 2) To optimize the synthesized async QDI combinational circuits so as to reduce the power dissipation, area (transistor count) and delay overhead without compromising the async QDI attribute.
- 3) To investigate and implement an async QDI dynamic voltage scaling (DVS) technique based on data completion detection for low power or high speed applications by adaptively adjusting the supply voltage of the circuit without compromising the operations.

1.3 Contributions of the Thesis

Three main contributions are highlighted and summarized in this thesis as follows.

First, a synthesis tool of async QDI circuit is proposed, and it has the following salient features. To the best of our knowledge, no other reported async QDI synthesis tools have incorporated all these features collectively.

- 1) The targeted synthesized circuits abide by the async QDI attribute. Hence, they are inherently timing-robust, accommodating the foresaid PVT variations and are desirable for applications with high variation-space and wide operation-space (including defense/space applications).
- 2) The coding specifications accept Verilog HDL, and are largely similar to the standard coding for sync circuits, no special and/or ad-hoc design/coding rules are required. Therefore, the proposed synthesis tool can be seamlessly integrated with the standard sync design flow, and fully leverage on the commercial EDA tools.
- 3) The proposed synthesis tool is applicable to accept various async QDI library cells, hence enabling it to exploit the full benefit of different library cells.

Second, a microcell-interleaving optimization technique, based on genetic algorithm, to optimize the async QDI combinational circuits is proposed. It mitigates the hardware overhead of the Data Completion Detection (DCD) circuit in async QDI

circuits. In this proposed optimization, three novelties are highlighted as follows:

- 1) A novel microcell-interleaving approach is proposed to reduce the number of Completion Detection (CD) circuits whilst retaining the required async QDI attribute. The advantage of this approach is to combine the best features of different async QDI combinational circuit as to achieve the optimal performance in term of power dissipation, transistor-count and delay.
- 2) The performances of the async QDI circuits based on the proposed microcell-interleaving approach is analyzed in terms of power dissipation, transistor-count and delay, and evaluate/determine the upper and lower boundaries of these performance profiles. With the analyzed profiles, the performance contour of the async QDI combinational logic can be determined. It can then be employed to explore the combination/percentage of each kind of microcells according to the specific requirements.
- 3) A Microcell-Interleaving Genetic Algorithm (MIGA) is proposed to stochastically optimize the proposed microcell-interleaving approach on the power dissipation, transistor-count, and delay. The proposed MIGA converges to global optimal solutions rapidly with the formulated genetic coding and fitness functions. The proposed MIGA is compared with the greedy algorithm and dynamic programming, and the results show that the proposed MIGA has outperformed the rest.

Third, a pipeline-granular async QDI DVS technique is proposed to achieve low power dissipation and accommodate PVT variations. The novelties/features of the

proposed async QDI DVS technique are summarized as follows:

- 1) No additional timing margin is required to ensure the delay of the circuit operation due to PVT variations. This is realized by the dual-rail data coding with async QDI attribute to adaptively accommodate the PVT variations.
- 2) An on-chip power supply unit with three voltage scales, 0.5V, 0.8V and 1.2V, is integrated in each pipeline. The supply voltage of each pipeline stage can be adaptively adjusted according to the corresponding working loads, making the energy saving more efficient in this pipeline-granular manner.
- 3) The proposed async QDI DVS is free of error-data-vulnerability when compared with the reported DVS techniques, as robustness of the circuits is aptly ensured by async QDI attribute.

1.4 Organization of the Thesis

The remaining of this thesis is organized as follows.

In Chapter 2, a comprehensive literature review on async design is presented. Specifically, the async communication channel is reviewed. This includes the async handshake protocol, data coding and push/pull attributes. Then, the four async design methodologies are discussed. The advantages of async digital systems and the reported synthesis methodologies for async systems are reviewed. Next, the reported async QDI data path design methodologies are presented, followed by a discussion on the reported DVS techniques.

In Chapter 3, a synthesis tool for the async QDI systems is proposed. Specifically,

an introduction is given to serve as a preamble of the proposed async QDI synthesis tool. Thereafter, async QDI pipeline structure is presented. The Verilog HDL is adopted as the default description language, complying with standard synchronous design flow. The synthesis details include the proposed design flow, data flow extraction and pipeline partition are presented. In order to validate the effectiveness of the proposed async QDI synthesis tool, a Finite Impulse Response (FIR) filter is synthesized, implemented and benchmarked against the reported counterparts.

In Chapter 4, an async QDI Data Path Optimization based on the Microcell-Interleaving Genetic Algorithm is proposed. The proposed microcell-interleaving approach is delineated with an example circuit. Thereafter, the analysis of the proposed microcell-interleaving approach is presented in terms of power dissipation, transistor count, and delay. In order to find the optimal solution in the gargantuan magnitude of search space, the Microcell-Interleaving Genetic Algorithm is proposed, including the genetic coding of combinational logic and formulation of fitness functions. To show the efficiency of the microcell-interleaving approach and the Microcell-Interleaving Genetic Algorithm, experimental comparisons are provided, including the comparisons with other reported design methodologies and other reported searching algorithms.

In Chapter 5, a pipeline-granular async QDI DVS technique is proposed. Specifically, an overview of the proposed DVS technique is presented, and it provides the general information of the pipeline architecture and operations. Then, each component in the proposed DVS technique is described in detail, including the power

supply unit and the level shifter. It is followed by the investigation of the four operation scenarios. Then, the size of transistors in power supply unit is formulated. Thereafter, an async QDI FIR filter with the proposed async QDI DVS technique is presented.

In Chapter 6, the conclusions of this thesis are drawn, and two potential areas for future research are recommended.

2.

Literature Review

2.1 Asynchronous Communication Channel

Communications between two or more circuit modules in an async system are implemented in channels. The channels can be classified into different handshake protocols, data coding and channel type attributes. The handshake protocols consist of *request* and *acknowledge* signals and they can be implemented as two-phase or four-phase handshake styles. The two-phase and four-phase handshake protocols are also known as non-return-to-zero and return-to-zero [21] handshake protocols respectively. The data coding is the format to encode data into channels, and it is usually expressed in the form of *m-of-n* coding (n is not smaller than m), where each data has n wires, and m wires of the n wires change values to present a complete data change. The most common data coding is 1-of-1 coding, namely *bundled-data*, and 1-of-2 coding, namely *dual-rail coding*. The channel type attribute is to indicate which party (sender or receiver) will initialize the channel. This attribute includes *push channel* and *pull channel*. The sender initializes the communication refers to

Fig. 2.1 (a) Two-phase handshake protocol, (b) Four-phase handshake protocol

push channel whereas the receiver initializes the communication in the *pull channel*.

In short, all attributes of channels can be expressed as follows:

$$\{\text{two-phase, four-phase handshake protocol}\} \times \{m\text{-of-}n \text{ data coding}\} \times \{\text{push, pull}\}.$$

The transitions of the request and acknowledge signals for two-phase and four-phase protocols are depicted in Fig. 2.1. For two-phase handshake protocol (see Fig. 2.1(a)), each signal transition on the request line and acknowledge line represents a communication event. In other words, one transition in request line and one transition in acknowledge line will complete a communication, irrespective if it is a positive or negative edge transition. Whereas, for the four-phase protocol (see Fig. 2.1(b)), the request line initializes the channel when the data is ready for transmitting. The receiver sets the acknowledge signal to high after it has completed the operation. The sender will then reset the request signal to zero to prepare for new data. Once the request line is reset to zero, the data on the channel will be considered invalid. Finally, the receiver resets the acknowledge signal to zero to complete the data communication.

For the data coding, high radix coding m -of- n , where $m \geq 1$, $n > 3$, is more suitable if the emphasis of the coding is on the reliable communication instead of the computational efficiency. For high computational efficiency which emphasizes on lesser number of wires to represent one data signal, the bundled-data or dual-rail is the preferred coding styles. The bundled-data uses one wire to represent one bit of data, the same coding as the sync circuits, but associates one request and one acknowledge signals to form a communication channel. The bundled-data coding has the advantage of fewer wires. However, the bundled-data coding is often associated with timing assumption, hence it is less robust to PVT variations. The dual-rail coding uses two wires, named $\{d.t, d.f\}$, to represent one bit of data. The meaning of dual-rail coding is explained in Table 2.1. Dual-rail coding is delay-insensitive because one of the two wires must have a transition to indicate a valid value, where two wires are reset to zero before next valid value.

Two channel type attributes namely the push channel and the pull channel are used in a communication channel. A communication channel is a push channel if the data sender initiates the communication. Otherwise, the communication channel is considered a pull channel. The push channel is applicable to both control-driven systems and data-driven systems. Whereas, the pull channel is only applicable to the

Table 2.1 Dual-rail coding and explanations

$\{d.t, d.f\}$ Coding	$\{0,0\}$	$\{0,1\}$	$\{1,0\}$	$\{1,1\}$
Explanations	Null Data	Boolean Logic 0	Boolean Logic 1	Not Used

control-driven systems as the receiver takes control over the sender.

2.2 Classifications of Asynchronous Digital Systems

In async community, there are four general classes of async circuits: Delay-Insensitive [22-24], Bundled-Data [25-27], Speed-Independent [28-30] and Quasi-Delay-Insensitive [31-38].

The delay-Insensitive circuits are highly robust but impractical. They are robust because they assume all gates and wires can have unbounded delay, which means circuits can operate correctly in any speed under any condition. Although the attribute of the delay insensitive is very attractive, limited classes of circuits can be built with delay insensitive attribute because the current-art sync standard library cells cannot be easily implemented in this class of circuits [39]. Therefore, only a limited number of gates can be employed to build the delay insensitive circuits, such as the inverter, the buffer and the C-Muller whose output switches to zero when all input are zero and switches to one when all inputs are one. However, using an inverter alone is very difficult to maintain the delay-insensitive attributes if a deeper investigation is taken at the transistor level because the fork wires from input wire are not guaranteed to reach PMOS and NMOS transistors at the same time due to unbounded wire delay assumption. Therefore, delay-insensitive circuits are very limited from a published standpoint.

Bundled-Data circuits suffer from PVT variations as the gate and wire delays are assumed to be bounded. This is the same assumption as the sync circuits, and these

circuits are not robust if there are large PVT variations in current nano-scale fabrication process.

Speed-Independent circuits are often less robust. It assumes unbounded gate delay and negligible wire delay. With the assumption of negligible wire delay, multiple-input gates can be employed to construct complex digital systems with completion detection at the input and output sections to ensure delay insensitive attribute. However, with the scaling down of the silicon process and increasing chip size, wire delay will continue to increase. In current deep submicron process, wire delay can be longer than gate delay. Therefore, the negligible wire delay assumption may not be true.

Quasi-Delay-Insensitive circuits can tolerate arbitrary gate, wire delays and handle isochronic fork [40] (strong intra-operator fork and inter-operator fork timing assumptions [41]), a condition that can be satisfied in practice. It is generally accepted within the async community that the QDI approach probably offers the most practical option [42-45] to accommodate PVT variations. This is because it innately detects the computation delays according to different workloads and operating conditions. Therefore, this thesis targets QDI circuits to manage PVT variation.

2.3 Advantages and Limitations of Asynchronous Digital Systems

The design frameworks of sync approach and async approach are depicted in Fig. 2.2(a) and Fig. 2.2(b) respectively. In the framework of sync approach, all the

Fig. 2.2 Synchronous vs asynchronous, (a) Synchronous approach, (b) Asynchronous approach

operations are triggered by the global clock signal. The completion of the operations is estimated by a given period of time, the clock period. On the contrast, async framework communicates through handshake signals. Compared to the sync approach, the async approach is potentially advantageous in the following areas.

First, *potentially lower power dissipation*. In general, the async circuits dissipate only the dynamic power when there is an input event/data. In other words, the async circuit is in the sleep mode until it is activated. After completing its operation, it goes back into sleep mode where no power is dissipated (except sleep mode leakage power) until the next active operation. Due to this attribute, async circuits have potentially lower power dissipation compared to sync counterpart. In other words, a digital system which has high frequency of idle or standby mode can be potentially good candidates for applying async approach. For example, async Network-on-Chip (NoC) [46, 47], which transfers data between different microprocessors or intellectual property (IP) blocks, goes into sleep mode and dissipates only leakage power when no

data needs to be transferred. After a useful data is being received, the NoC is activated, and dissipates dynamic power during transfer process. If NoC is designed in sync logic style, it may dissipate a significant amount of power even if it is in the waiting mode due to long and complex clock distribution trees. A similar technique to achieve low power in sync circuit design community, known as clock gating, gates the clock switching in sub-circuits where no operation is performed may be similar to async approach. The clock gating might be useful in shutting down some idling circuits to save power. However, in order to fulfill a fine clock gating, additional scanning circuits are required to be implemented to detect the timings and the sub-circuits to be shut down, which makes the system more complex, thereby dissipating more power. In addition, clock gating may exacerbate the distribution of global clock, which will increase clock skew problem and degrade system performance.

Second, *average-case performance*. In sync systems, a clock period which satisfies the longest combinational circuit operation (worse-case) needs to be determined to ensure all the operations can be completed within the clock cycle. This worse-case clock period unnecessarily slows down the speed of the system when most of the time the system is not performing the longest combinational circuit operation. However, for an async circuit, the communications between different computational blocks are performed by handshake signals, so each computational block only relates to its predecessor and successor. In this case, every computational block can start its tasks immediately when it receives all inputs, thus, eliminating the need to wait for

the global signal. Therefore, the async system can potentially provide average-case performance over worst-case performance of its sync counterpart.

Third, *lower electromagnetic interference*. In sync circuits, all modules switch at the same clock frequency simultaneously, and this causes high current glitches in power supply line, which probably drives analog circuits, drawing current from the same power supply or connected to the same ground. As a result, the circuit malfunctions. Furthermore, when clock signal switches, it generates electromagnetic radiation proportionate to the frequency of clock, and the electromagnetic wave (interference) is probably wrongly received and recognized as useful signal by radio transceivers. High electromagnetic radiation can pose a security problem for many circuits due to side-channel attack or power analysis attack [48, 49]. In contrast, in async systems, no common clock signals are shared by global modules, no high current glitches appear in the power supply line, and analog circuits in the same system will not encounter the problem of power supply and/or ground noise. Moreover, the frequencies of electromagnetic radiation scatter over a wide range of frequency domain due to independent switches. This will prevent the radio frequency modules from being affected, allowing low electromagnetic interference to be observed from the async circuits.

Fourth, *absence of clock skew problem*. With the increase of system scales, sync systems face the challenge of distributing clock trees properly to prevent circuit malfunction from clock skew problem. While in async systems, they do not suffer

from clock skew problem as there is no global clock signal. This benefits all async systems.

Fifth, *good modularity*. In sync circuits, especially SoC, function blocks and IPs are designed for a certain clock frequency. Therefore, it is difficult to migrate these blocks to other systems and platforms due to different optimal clock frequencies, even if it is possible, it is also difficult to cater to all modules for optimal performance. In contrast, async modules can be easily migrated to any systems and platforms because they are performed independently, as long as handshake protocols are compatible with each other.

Despite the advantages mentioned above, async design approach is mainly explored in the field of academic research, and modern circuit systems are largely dominated by sync design approach. Several reasons from literatures have accounted for the predominance of sync approach over the async approach.

First, *higher design effort*. Avoiding hazard makes async circuits more difficult to design than sync counterpart. In sync systems, it is easier to deal with the timing issue. Circuits are safe as long as clock period is set long enough to settle down all operation to prevent malfunction from hazard. However, in async system, circuits start to perform computation as long as the input signal changes and control signals are ready. Circuits will be driven into malfunction if glitches are introduced at the input of control circuit. Therefore, glitches should either not be introduced in the first place or be removed by circuits. Some strict constraints are imposed on async circuits, such as single-input changes or multiple-input changes with timing assumption, but those

constraints have severely impeded the speed performance of async controllers. In all, sync circuits use clock signal simply to deal with timing issue, while designers need to take more care to design hazard-free async circuits.

Second, *lacking mature design flow and automation*. Lacking a widely accepted design flow and automation is one of the most important reasons that hinder async circuits from becoming more popular in the industry. For modern complex and very large scale integration (VLSI) circuit design, it is necessary to have a complete design flow from high level of abstraction to low level circuit implementations, such as the sync circuit design flow. Although lacking a mature and widely accepted design flow for async circuits, many design flows in the field of academic research are proposed.

2.4 Synthesis Methods for Asynchronous Digital Systems

In this section, existing methodologies and tools for async design are reviewed and the advantages and limitations in them are identified. The existing synthesis methodologies for async systems can be broadly classified into four categories: syntax-directed translation, logic synthesis, desynchronization, and Null Convention Logic. These methods will be reviewed in turn.

2.4.1 Syntax-directed Translation

This methodology implements an async circuit system through a high level language that is based on the concept of Communicating Sequential Processes (CSP) [50], whose key characteristics include the support of concurrent processes, sequential and concurrent composition of statements within a process, and synchronized

message passing over channels. Syntax-directed translation is used to convert high level programs into hardware by replacing each of the program construct into corresponding hardware handshake component.

The syntax-directed translation methodology forms the basis of many async circuit synthesis systems. It originated from Tangram group at Philips Research [51]. Tangram design flow is supported by a proprietary CSP-based language and private tool set from Philips Research Lab. This methodology was available to the public until the tool sets of Balsa [52, 53] from Manchester Amulet Group were released. Syntax-directed translation was also used in OCCAM-based design system [54], Communicating Hardware Processes (CHP) [55], SHILPA [56], Rainbow [57, 58].

System specification of syntax-directed translation methodology is first described in high level CSP-like language. Then, the specification is compiled to a parsing tree, which is subsequently mapped to a network of handshake components. The network of handshake components can be used for functional verification, and is also used in system placement and routing. Thereafter, timing information can be extracted to facilitate its function verification.

There are several advantages of using this syntax-directed translation.

First, a dedicated circuit construction flow which appropriately models the concurrency of async circuits is provided. In this case, the designers will require lesser effort to understand the operation of the async circuits. Second, the CSP-like language has a high level abstraction; it masks the complexity of circuits from system level; designers can focus on the development of algorithm and architecture of

systems. Third, syntax-directed translation is transparent to designers; it maps the operation of programs to corresponding handshake modules, thus, correct circuits can be obtained through structure, making it feasible and easy for designers to check the functional correctness in small designs.

However, this syntax-directed translation methodology also has several disadvantages.

First, high-level abstraction language and directed-translation techniques imply that the circuit systems are highly un-optimized. This is because the high-level abstraction masks the details of circuit architecture. In this way, different program constructs may introduce significantly different logic complexity with same functionality. In order to alleviate this drawback, designers must have a superb understanding of async circuit implementation, and be proficient in coding. Directed-syntax translation has inherently high overhead due to plenty of redundant handshake channels and modules built into circuits, and the switching on handshake lines may dissipate significant amount of power and degrade the advantage of low power in async circuit design over its sync counterpart.

In order to alleviate the problem of redundant handshake logic, some optimization techniques have been reported. One of the optimization techniques applied is peephole optimization [59, 60]. This technique reduces overhead by looking at several local handshake controllers that meet predefined requirements, e.g. they have common input channel(s) or common output channel(s), and replacing them with a predefined more efficient controller that performs the same function. This

optimization has a significant impact if the targeted circuit systems are highly redundant, however, it will not make great difference if the targeted systems have been carefully designed. Another more aggressive optimization technique is the clustering optimization. This technique groups handshake circuit together to re-synthesize into one controller that performs the same function as before by determining the optimal optimization, which seems to make a great impact on the optimization and also looks promising. However, clustering optimization faces the difficulty of determining the optimal clustering, in other words, what are the optimal clustering groups. One method is to automatically determine optimal clustering through software algorithm, but this method is computational expensive, which makes it impossible for application with large systems. Another method is to cluster handshake circuit by fixed rules or by designers. The most serious disadvantage for this method is that handshake circuits are computational expensive and not fully optimized, e.g. clustering all handshake modules into one cluster, could lead to state explosion problem for large systems. When they are re-synthesize into one controller, or some specified clustering, this could lead to tedious work if systems are large and complex. Even with those optimization techniques mentioned above, there are still substantial evidence [59, 61] that shows directed-syntax translation methodology suffers high controller overhead.

Second, the directed-syntax translation methodology is a stand-alone design flow. Although the CSP-like language is appropriate to describe the concurrent behavior of async circuits, it is still esoteric to some hardware designers, so retraining is required

to alleviate the redundant logic problem due to inappropriate coding style as discussed before, which is potentially expensive and time-consuming. Moreover, directed-syntax translation methodology does not leverage on the commercial sync design flow and tools. For example, when implementing a system using directed-syntax translation, a library that includes handshake modules and macros that support all logic computations need to be provided. For handshake modules, it is reasonable to be predefined because they are customized for async circuits, whereas, the pre-design of the latter excludes the use of powerful commercial logic synthesis tools, e.g. Synopsys, which is disadvantageous for directed-syntax translation. If industries migrate from sync circuit design flow to directed-syntax translation methodology or merge directed-syntax translation methodology into sync design flow, a lot of investment is required outside the research lab, which is undesirable.

In summary, directed-syntax translation employs CSP or CSP-like language to describe the async circuits from high level abstraction. The language is appropriate to model the concurrency of async circuits and masks the complexity of circuits, but it results in redundant handshake logic overhead. Moreover, the directed-syntax translation is a stand-alone design flow, it cannot leverage on the commercial EDA tools, and plenty investments are required to make it standardized. Therefore, directed-syntax translation design methodology is not widely accepted.

2.4.2 Logic Synthesis

Logic synthesis methodology mainly employs Petri Net (PN) [62] to describe async circuits, also known as the graph-based methodology. The system specification

is usually specified in Signal Transition Graph (STG) [63]. Graph specification is not straightforward to recognize the functionality of system, error prone and difficult to debug, so for complex system, it is too large and tedious to manage. Thereafter, high-level language based description approaches are proposed [64, 65]. Employing existing HDL language to specify async circuits from the system level not only mask the complexity of systems, but also enable off-the-shelf commercial EDA tools to verify functionality, which is definitely a promising interface with current sync design flow. Thereafter, system specifications are specified in high level abstraction, and some tools, for example, Pipefitter [66], are reported to transform high level system specification into graph-based specification, and existing graph synthesis tools, such as Petrify [67], can be employed to synthesize the PN or STG into speed independent circuits. Basically the flow of logic synthesis contains data path implementation and controller implementation, but the main attention is focused on the controller part, thus, this methodology is a controller-based approach. Data path can be implemented by commercial logic synthesis tools, such as Design Compiler. For the implementation of controller part, two methods are introduced. One of which is direct mapping, while the other is logic synthesis. Direct mapping maps the places of graphs into circuit cells, and graph arcs map into corresponding connections. The logic synthesis uses Petrify tool to algorithmically synthesize the Petrify graphs into circuits. Then, the data path and controller will be merged together to obtain an entire digital system.

The most significant advantage of logic synthesis can be the partial compatibility to current sync design flow. However, it has some shortcomings as follows.

First, logic synthesis methodology adheres to speed-independent assumption. The wire delay in current processes cannot be completely neglected, so the assumption of speed-independent is weakly violated. In order to make SI circuits work correctly, high-quality layout is required to align gates as close as possible. However, with the scaling down of processes, and increasing chip size, the delay on wire increases significantly, even larger than that of gates, making the speed independent assumption impractical.

Second, the high-level descriptions and graphs are overly complex and not straightforward for large and complex systems which makes debugging tedious or impossible.

Third, the graph synthesis process is highly complex and prohibitively time-consuming as the size of graph increases exponentially with the size and complexity of circuits. This problem is termed as state-explosion problem [68, 69]. For graph-based methods, a fundamental challenge is to fully encode every reachable state of the controller, through which there is a need to insert additional internal signals in order to resolve state-encoding contentions, which in turn leads to a more complex graph. This is a tangible and difficult problem for large designs. The synthesis time for large and complex systems tend to be intolerable [70], even with advanced synthesis tool Petrify, this poses a serious disadvantage for commercial design environments because TTM is crucial consideration for creating commercial

products. One way to solve this problem is to partition a large and complex graph into several smaller parts, and synthesize these subparts separately. However, currently no tool has been designed to perform this job. Designers can partition it manually, but the criterion to follow is unclear, so more research needs to be carried out.

Fourth, the logic synthesis flow is not fully compatible with sync design flow. Most of the tools that support HDL high level description and transform system specification into graph only support a subset of HDL constructs. For example, pipefitter, which transform Verilog code into STG, only support *module*, *function*, *initial*, *always*, *wait*, *if*, *case*, *fork*, *join* constructs, however, some of these constructs are not synthesizable, and is only meant for function modeling.

Fifth, as of yet, there has not been a mature method to partition controller and data path from graph automatically. Usually, the partition is performed manually by designers, but for large system, this work is tedious and error prone, because the graph is intolerably large when systems become very complex. Another way of partitioning is automatically performed by tools, however, this technique is still immature, and still has a long way towards maturity.

In summary, logic synthesis methodology has partially leveraged on standard sync design flow, however, it targets SI circuits, which is increasingly impractical, and it suffers high complex graph synthesis, which is also not practical for big and complex systems. Therefore, logic synthesis methodology is, to some degree, not suitable for modern and complex system design.

2.4.3 Desynchronization

Desynchronization [71] was first proposed by J. Cortadella. This methodology is a direct conversion from sync system into async system.

Systems are firstly specified in sync code, using Verilog or VHDL languages, and functionally verified. Thereafter, the sync systems are synthesized into netlist using the commercial synthesis tools, i.e. the Design Compiler tool. In this context, the design flow is completely sync and it served as a significant advantage for desynchronization as it is completely compatible to the standard sync design approach. After a sync circuit netlist is generated, the desynchronize transformations are employed to convert it into an async circuit list. This is done by first replacing all the global clock signals and flip-flops with corresponding handshaking signals and latches while the combinational logics remain unchanged. Secondly, a matched-delay chain corresponding to the longest delay of the combinational logics is generated. This is to ensure the control signals from the controllers communicate with the preceding and following stages after its combinational operations are completed. Thereafter, all the async handshake controllers and the data path are connected into a complete pipeline. Finally, the async circuit netlist is ready for place and route and functional verification.

The most advantageous point for desynchronization is that it starts from the netlist of sync systems, which is completely compatible to current sync circuit design flow. Transformation from sync netlist into async netlist can also be completely transparent to designers, which masks the complexity of async design approach.

However, the resulting async circuits belong to the class of bundled-data, which is not robust and suffers PVT variations. It also encounters area overhead problem when compared to sync circuit design [72].

In summary, desynchronization is a promising methodology of converting a sync circuit into async circuit. It is compatible to sync design flow as it starts from a sync netlist, which leverages on the commercial EDA tools. The transformation process from the sync netlist to async netlist is transparent to the circuit designer. However, it has several shortcomings; the resulting circuit is classified as bundled-data and requires a delay margin assumption, which is not robust to tolerate large PVT variations. Similar to logic synthesis methodology, the final desynchronization circuit also suffered a large area overhead problem when compared to its sync counterpart.

2.4.4 Null Convention Logic

Null Convention Logic (NCL) [73] is another methodology of designing async circuit from Register-Transfer-Level (RTL) code. Systems are specified in Verilog or VHDL code. The code must only contain RTL description. RTL code is then synthesized into 3NCL circuits that are the same as Boolean prototype. 3NCL circuits are translated into delay-insensitive implementation using Delay-Insensitive Minterm Synthesis (DIMS) [21]. At last, the DIMS-based circuits are transformed into netlist customized for NCL library cells. In order to simulate NCL circuits in RTL, tools are required to support NULL value and hysteresis behavior of threshold gates, which is problematic as it is not standard in sync design flow; further modifications of tools or special packages are required.

Accepting RTL Verilog or VHDL code is one of the advantages which enable the description of circuits from high level abstraction and commercial EDA tools can be employed for function verification and synthesis. Furthermore, it targets QDI circuits, which is robust and practical. However, from the design flow, some disadvantages can be observed. First, description code is somewhat limited. It requires tools to support NULL and hysteresis behavior of threshold gates, which is not normal in sync design flow, and therefore, problematic. Second, the design flow is customized for NCL library cells, thus, lacking an option to use other kinds of library. In order to maintain delay insensitive attributes, a lot of logics are included for checking the completion of all internal signal states. If combinational logic blocks are large enough, the added-in checking logic will increase significantly, which will exacerbate the power and speed attributes.

In summary, Null Convention Logic is a practical and robust but has high overhead and is not a standardized async QDI circuit design methodology.

2.5 Design Methodologies for Quasi-Delay-Insensitive Data Path

Circuits preserve QDI attribute if the design issue of ‘observability’ [74] is satisfied, where all the gates and wires therein are detectable/acknowledgeable. The ‘observability’ will avoid ‘gate orphan’ and ‘wire orphan’. The gate/wire orphan refers to an internal gate/wire that is able to switch its output but the switching is masked from observable outputs of the entire circuit. In async community, there are generally

two reported methods to address the issue ‘observability’ of circuits.

2.5.1 Non-input-complete Microcell with Completion Detection Circuit

Microcells whose outputs can be all valid/null before all inputs are all valid/null are non-input-complete microcells, whereas, microcells whose outputs can be all valid/null only after all inputs are all valid/null are input-complete microcells.

Non-input-complete microcells are generally only implemented with logic function with dual-rail coding style in themselves. In order to maintain the robustness of the circuit, the validity/nullity (addresses the ‘observability’ issue) of internal wires are detected by additional completion detection circuits with the isochronic forks timing assumption, which leads to practical realizations of digital QDI circuits. The combinational circuit example with non-input-complete microcells and CD circuits is depicted in Fig. 2.3(a). The states (validity/nullity) of intra-wire (the output of microcell *NIC1*) cannot be observed through the output of microcell *NIC2*, as *NIC2* is a non-input-complete microcell. In order to observe the states of wire inside of the circuits, some completion detection circuits (usually single-rail OR gates and C-Muller [21]) are intentionally integrated into the combinational logic. With the isochronic fork assumption, the states of intra-wires can be observed at the output of C-Muller microcells (*done* signal).

The non-input-complete microcell with completion detection circuit method features high operation speed but high hardware overhead. The non-input-complete microcells are implemented only for logic operations, and with no consideration of input-complete attribute. Hence, they have smaller number of transistors in series

compared to the corresponding input-complete microcells, resulting in higher charge and discharge speed. However, additional completion circuits and C-Muller tree are integrated to maintain QDI attribute, which incur heavy hardware burden in power dissipation and area.

In light of high hardware overhead drawback of this method, there is a substantial on-going research effort to design QDI circuits with reduced hardware overhead.

Fig. 2.3 (a) Combinational circuit with non-input-complete microcells and CD circuits, (b) Combinational circuit with input-complete microcells

Reported techniques include adopting a dynamic logic implementation (instead of a static logic implementation to reduce the number of transistors [21]), CD circuits with a high fan-in structure [74], 1-of-4-rail logic (but with more complex circuitry), optimization by relative-timing analysis [75], etc. These reported researches mitigate the overhead of QDI circuit in some aspects, but some of them have introduced new timing assumption, resulting in the so-called QDI circuits being less robust.

2.5.2 Input-complete Microcell

An alternative method to address the ‘observability’ issue is to employ input-complete microcell to implement the circuits. The circuit example with input-complete microcells is depicted in Fig. 2.3(b). Microcell *IC2* automatically detects the validity/nullity of the intra-wire, and the states of intra-wire can be observed through the output wire of microcell *IC2*. The DIMS is the most widely accepted method for implementing the input-complete microcells.

In this method, the input-complete microcells are realized to not only perform logic computations, but also indirectly infer the validity/nullity of signals. Due to the latter functionality, input-complete microcells are generally much more complex than the corresponding non-input-complete microcells. Hence, the input-complete microcell method features relatively low hardware overhead but significantly increased delay. In view of the drawback of this method, many optimization methods are reported, including local input completeness relaxation [76] coping combinational logic delay [77], synthesis based on partial acknowledge [78], etc.

The above two QDI circuit realizations have their own advantages and limitations.

The former non-input-complete microcells with completion detection circuits feature lower realization complexity but faster speeds. In contrast, input-complete microcell realization does not require additional completion detection circuits to observe the validity/nullity of intra wires, resulting in potentially lower overhead. However, the realization of input-complete microcells is more complex than that of non-input-completion microcells, and also features much slower operation speed. In view of the advantages and limitations of these two QDI circuit realizations, potential overhead optimizations can be explored by mixing them together in QDI circuit implementations.

2.6 Dynamic Voltage Scaling Techniques

A critical issue for the DVS technique is to maximize the energy reduction by dynamically scaling the supply voltage according to the corresponding working load, and simultaneously ensure the correct operation. The challenge of ensuring the correct operation arises from the different timing variations due to fabricating process variation, supply voltage change, and environment temperature fluctuation, etc. In order to cope with all these possible variations to ensure correct operations, supply voltage must be tuned sufficiently to cover the worst case. There are substantial research work on DVS, and these reported DVS techniques can be generally classified into conventional no-error tolerable DVS and razor-based error tolerable DVS.

2.6.1 Conventional No-error Tolerable Dynamic Voltage Scaling

In the conventional no-error tolerable DVS technique, voltage and/or timing

margins are pre-added to circuits, catering for the timing uncertainty caused by PVT variations. Among the reported research, lookup table (LUT) and critical path replica [79-81] are two widely used approach.

The LUT approach is to store the pre-characterized voltage-frequency relationship into chips, and the predetermined supply voltage value is read out and automatically scaled according to the working frequency in operations. In this approach, the voltage-frequency relationship must be conservatively characterized to ensure correct operations. In other words, an additional voltage margin should be added for a fixed frequency, or a conservative frequency is determined at a given supply voltage. Due to the conservative prediction of computation completion, the energy reduction impact of DVS technique is reduced. Another drawback of the LUT approach is that it does not guarantee correct operations in view of the different variations despite a delay margin is intentionally included.

Another widely reported approach to ensure correct operation is the critical path replica [79-81]. In this approach, additional delay chains are designed to replicate the delay of critical path. Usually, an additional timing margin is added to the critical path replica, expecting that the critical path replica fails before the real critical path. Although the critical path replica can cope with the global PVT variations, it is still vulnerable to intra-die/local variations, where the critical path replica probably does not replicate the exact timing of the real critical path.

In summary, the conventional no-error tolerable DVS technique intentionally provides a delay margin in the circuits to accommodate the local and/or global PVT

variations, to ensure a correction operation. However, the circuit employs this DVS technique is vulnerable to any error. In other word, this circuit can be malfunctioned with a single error. In order to alleviate this risk, an alternative error tolerable DVS technique is reported, and reviewed in the following section.

2.6.2 Razor-based Error Tolerable Dynamic Voltage Scaling

The idea of Razor-based error tolerable DVS technique [82-86] is to scale the supply voltage to a point where systems begin to fail, and design additional error-correction circuits to replay the operations if errors are detected. The Razor-based error tolerable DVS technique reduces energy more significantly than the conventional DVS as the supply voltage of the first failure point is lower than that of the latter (which adds a voltage margin to ensure the correct operations).

With the supply voltage of Razor-based systems scaled to the first failure point, the malfunction possibility of the systems increases significantly due to the variations in voltage supply, intra-die process and the environment temperature. In order to ensure the correct operation, Razor Flip-Flop was proposed [87, 88]. Within the Razor Flip-Flop, input data is stored in DFF at the positive edge of clock signal, like conventional sync circuits. Besides that, an additional latch, called Shadow Latch, is designed to latch input data at the positive level of clock signal. Therefore, the latch has half more clock cycle to receive and capture data, expecting the correct data is latched in Shadow Latch. Thereafter, the comparison of the two data in DFF and Shadow Latches are performed. If these data are different, an error flag will be asserted; the correct data in Shadow Latch will be reloaded at the next clock cycle.

Through the working process of Razor-based error tolerable DVS technique, it can be observed that the Razor allows the error data to propagate, causing the subsequent pipelines to generate wrong data as well. The invalid operations can reduce the efficiency of energy saving by scaling down voltage, especially when many pipeline stages are involved and/or the error rate is high.

Table 2. 2 Comparisons between the reported DVS techniques and the proposed DVS technique

	Advantages	Limitations
Conventional DVS	1. Easy implementations	1. No error data toleration 2. Suffer PVT variations 3. Require additional timing margin
Razor-based DVS	1. Error data toleration for half cycle time 2. No timing margin is required.	1. Additional overhead due to error-tolerant DFF 2. Potential error data propagation in speculation window 3. Cannot tolerate timing variation more than half cycle time
Proposed DVS	1. No timing margin is required 2. Accommodate PVT variations due to QDI circuits 3. Voltage can be adaptively adjusted in each pipeline	1. Additional speed overhead due to the reset phase of QDI circuits. 2. Additional overhead due to dual-rail circuit

In order to highlight the advantages and limitations of the reported DVS techniques and our proposed async QDI DVS technique in Chapter 5, their respective advantages and limitations are summarized in Table 2.2.

3.

Proposed Synthesis Tool for Asynchronous Quasi-Delay-Insensitive Digital Systems

3.1 Introduction

As discussed earlier, with the continuous scaling down of semiconductor process, PVT variations will become more significant, making it a challenge to synchronize the data signal through a global clock signal. This is especially so with high frequency operations ($\sim$ GHz) and large digital systems (SoC and multi-cores processors). Hence, many researchers have shifted their interest to other design approaches which include the Globally-Asynchronous-Locally-Synchronous (GALS) and async approaches. The async circuit design is an alternative to address the varying timing issues, and has proven to be functionally robust [89, 90] against timing variations. Other potential advantages of these async circuits (over their sync counterparts) include no clock skew, less electromagnetic interference, potentially high speed (due to average-delay over the worse-case delay) operation [91, 92]. Despite all these advantages, the async approach is still not widely accepted, mainly due to the lack of appropriate async design flow or EDA tools to alleviate the design effort. Although several async EDA

tools such as Pipefitter, Balsa, etc. have been reported, they do have several limitations.

This chapter will review some of the limitations of the reported synthesis methods for async circuit. In addition, a synthesis tool of robust async QDI systems is proposed. It has the following prominent features. The targeted synthesized circuits, abiding by the QDI protocol, are inherently timing-robust, accommodating the foresaid PVT variations and therefore are desirable for applications with high variation-space and wide operation-space (including defense and space applications). The coding specifications accept Verilog HDL language. They are similar to the standard coding for sync circuits. Hence, no special and/or ad-hoc design/coding rules are required. This allows the proposed synthesis tool to be seamlessly integrated with the standard sync design flow and fully leverage on the commercial EDA tools. The synthesis tool can accept various QDI library cells, hence enabling the full merit of different library cells to be explored.

3.2 Four-phase Dual-rail Channel for Proposed Synthesis Tool

The async communication channel has been reviewed in Chapter 2. Intuitively, two-phase handshake signals are more suitable for low power design and high performance since there are less signal transitions for each communication as it has no reset phase. However, four-phase circuits are typically smaller than two-phase circuits because two-phase circuits have to capture signal transitions instead of the value level

Fig. 3.1 A delay-insensitive channel using the four-phase dual-rail protocol

of signals. Moreover, the implementation of edge transition circuits is often larger and more complex too. Taking into account the power dissipated by the additional complex circuit for the two-phase communication, the four-phase handshake protocol is usually more power efficient than the two-phase protocol. In terms of speed performance, the four-phase handshake protocol is usually slower than the two-phase protocol but more reliable. This is due the data path in four-phase protocol has a null and a valid reset phase to ensure data are completely stable during the transmission.

The dual-rail coding is employed for its delay-insensitive attribute. The data validity/nullity can be detected through the coding wires without request signal, as it is separated by null value in dual-rail coding. In other words, the request signal is automatically encoded into data wires. Hence for n bit Boolean logic, $2n+1$ wires are required. In the context of dual-rail coding, active controllers are not required to generate request signal as dual-rail coding is data-driven style.

In summary, the proposed synthesis tool adopts async communication channels in four-phase handshake protocol and dual-rail data coding with push channel (due to its

lower power dissipation and QDI attributes). Fig. 3.1 depicts the four-phase dual-rail push channel operation.

3.3 Quasi-Delay-Insensitive Pipeline Structure

The design styles of QDI pipelines can be categorized into two paradigms, the integrated pipeline and the data-control decomposition. The former paradigm (such as pre-charged half-buffer [40]) is a fine-grain pipeline where its QDI circuits are optimized at the transistor-level by collectively integrating a functional logic and a controller (which includes a CD circuit and a latch). This paradigm yields high speed designs but at the cost of high synthesis effort and high IC area overhead. Conversely, the latter paradigm (such as NCL [74], DIMS, Pre-charge-Static Logic [8], etc.) is a coarse-grain pipeline that considers the functional logic and the controller independently. This paradigm, conceptually similar to a sync pipeline, features design simplicity (in part by leveraging on commercial tools and occupying a smaller IC area) but at the cost of reduced speed largely due to its coarse-grain structure. In this thesis, the latter paradigm is of more interest, as the IC area is smaller and power dissipation is lower.

Of the various data-control decomposition QDI methodologies, the NCL_X design methodology [45] is reported to have an efficient design flow and yields designs with relatively low area overhead (compared to the NCL_D design methodology). Its design flow involves synthesizing a standard single-rail netlist and a transformation of an NCL (dual-rail) netlist. To preserve the QDI attribute, a DCD

embodying local CD circuits is required to detect the outputs of microcells therein. The DCD circuit does not only fully acknowledge the output validity/nullity but also to avoid gate orphans [45]. The gate orphan indicates that a gate is activated but not acknowledged by its output.

Fig. 3.2(a) depicts the async QDI pipeline structure, comprising $Register_i$, $Controller_i$, $QDI\ Combinational\ Circuit_i$, Register Completion Detection (RCD_i) and DCD_i . Particularly, RCD_i and DCD_i are essential in addressing ‘input completeness’ and ‘gate orphan’ to ensure the QDI attribute of the pipeline. The ‘input

Fig. 3.2 (a) QDI pipeline stage, (b) Completion detection tree and structure of C-Muller, (c) Implementation of register

completeness' indicates that all inputs have to be acknowledged before the next operation commences. This is to prevent the error that the output is ready before the input is ready. The elimination of gate orphan ensures that all gates in one block are acknowledged before the next operation commences.

To better delineate the QDI attribute, the detailed operation of the async QDI pipeline is investigated. The *Data* is first passed via *Register_i*, and *RCD_i* (i^{th} Register Completion Detection) will check the 'input completeness' attribute by triggering *ACK_i* if the inputs are all valid or empty. *Data* will also trigger *QDI Combinational Circuit_i* for computation. *DCD_i* will check all intermediate outputs of cells to ensure that no gate-orphan occurs within *QDI Combinational Circuit_i*. *Controller_i* controls *Register_i* according to the validity/nullity of inputs between the current and next pipeline stages and also the error-free robust data transfer. It is observed that the reset phase (which does not have any computational operation) takes an equal or longer time than the set phase (true computational phase). Therefore, the output signal *Req_i* is designed to fast reset *QDI combinational logic*, in this case, some fast reset library cells, such as the Differential Cascode Voltage Swing Logic (DCVSL), can be incorporated to reduce the cycle time of pipelines and improve the speed performance. Fig. 3.2(b) depicts the circuit implementation of *DCD_i* comprising of OR and C-Muller gates. Since the C-Muller gates are considered a strong-indicating logic gate and the isochronic fork assumption is applied at the input of OR gates, QDI attribute is maintained. Fig. 3.2(c) shows the implementation of *Register_i*, C-Muller gates are also employed to ensure the pipeline's QDI attribute.

3.4 Specification Language - Verilog Hardware Description Language

A common drawback of many async design methodologies, such as syntax-directed translation, is the difficulty of using the commercial tools in the IC design industry. Adoption of an async design flow therefore requires considerable investments in time, effort and financial resources to set up the design environment and educate the IC designers on async designs and the usage of the respective tools. The lack of integration also means that the strengths of, and support provided for, existing tools cannot be leveraged. Our approach places a strong emphasis on fully leveraging on the off-the-shelf design flow and tools, so that our flow can be seamlessly integrated into industry design flow.

With the emphasis of fully leveraging on commercial design flow in mind, a hardware-driven specification language is believed to be natural and more suitable for circuit system description instead of programming language, although programming languages, such as CSP or CSP-like language that supports the concurrent behavior of async communication. HDLs that have gained widespread acceptance within the VLSI design community, such as VHDL and Verilog HDL – both of which are formally defined by IEEE standards [93, 94].

The use of standard HDL as specification language can bring significant benefits to IC designers and electronic industries. Firstly, (sync circuit) designers are more familiar with standard HDLs. No retraining efforts and investments are required to be carried out. Secondly, the adoption of standard HDLs enables async circuits design

flow to fully enjoy the availability of a large amount of powerful sync circuit EDA design tools, such as behavior simulator, logic synthesis, etc. Whereas for stand-alone async circuit design flow, all tools have to be built to make design flow complete, which definitely requires a lot of investments and it is also time-consuming. Thirdly, the adoption of the standard HDLs in async EDA design flow will simply the design of mixed sync and async systems. For modern SoC systems, it may not be feasible to construct a complex system which is completely sync or async. In most case, it is in part sync and in part async, such as the GALS systems. The sync part is usually designed by the current design flow, since a significant number of IP blocks and EDA tools are available. It will definitely simplify and speed up the efficiency of the whole flow if the async portion is also designed under the same environment.

Adoption of standard HDLs to design async circuits is not a new idea. Some methods [95] are already using HDL to help the function verification. They convert system specification, such as CHP, STG into corresponding HDL, and use

Fig. 3.3 Supported synthesizable Verilog constructs

commercial simulators to verify the functionality of systems. A common limitation is that HDL is used in one branch of these design flows. The role of HDL is to facilitate their esoteric flow. However, the HDL is the main stream, and is therefore used from the system specification to the back-end activities in our approach. Some other methods [96] use HDL as their specification language, but extra modeling rules are required. In some reported designs, the description of systems using VHDL is only limited to RTL level [73]. In [96], although system specification can be described in both behavior and RTL levels, the coding rules do not support sync clock description style, instead, Verilog is just used to describe the data flow relation and the location of inserting async controllers. This approach is hard to verify functionality if system is complex because appropriate delay is required to be added in manually, which is problematic. Some methods start from standard sync code. In [71], the author introduces async concept at the netlist level, which is probably too low a level to have full optimization, whereas, our approach also adopt a complete standard sync code, and introduce async concept at code level, thus, potentially resulting in an increase in optimization.

Verilog HDL is adopted as the default specification language for our proposed synthesis tool, and the code style is completely compatible to sync system code, no other special code rules need to be considered. Based on the standard model/code style, a subset of Verilog synthesizable constructs are supported, which is shown in Fig. 3.3. They are *Declarations*, *Sequential Process*, *Conditions*, *Operations* and

Assignments. It is interesting to note that these constructs are synthesizable and virtually sufficient to describe any complex system.

3.5 Design Compilation

In this section, the details of the proposed synthesis tool will be described which include the design flow, data flow extraction, pipeline partition, combinational logic and pipeline generation, etc.

3.5.1 Design Flow of Proposed Quasi-Delay-Insensitive Synthesis Tool

In this section, based on the QDI pipeline structure and the Verilog constructed, the design flow of our synthesis tool is presented as shown in Fig. 3.4. The flow includes three parts, parts I, II, and III. Parts I and III are the routine front-end and back-end design flows respectively, and Part II pertains to our proposed async design flow.

First consider the standard design flows of Parts I and III. In Part I, the digital circuit is modeled in the standard format, simulated and then verified. This flow is essentially the same as a sync circuit design. On the other hand, in Part III, a final netlist (from Part II, see later) is used for layout implementation (i.e. place and route). Commercial EDA tools can be leveraged on for Parts I and III.

Consider now Part II – our proposed async QDI design flow that bridges between Parts I and III. The design flow essentially involves first extracting data flow, pipeline partition from intermediate presentation-data flow chart, thereafter generating QDI

Fig. 3.4 Overall design flow of the proposed synthesis method

Fig. 3.5 (a) Details of combinational logic generation, (b) Details of pipeline generation

pipeline framework (with handshakes and controllers), generating, optimizing dual-rail combinational logic (CL) and finally obtaining async dual rail netlist by merging CL into the QDI pipeline. Fig. 3.5(a) further delineates how a dual-rail CL is generated simply by a single-rail to a dual-rail expansion. The single-rail circuit can be obtained easily by using the commercial synthesis tool (e.g. Design Compiler), and its corresponding dual-rail expansion is a direct mapping (based on an async library). Fig. 3.5(b) depicts in detail a generated handshake configuration. This step mainly maps the handshake configuration information generated in pipeline partition into corresponding handshake components, framing the backbone of the whole system.

3.5.2 Data Flow Extraction

This step is to extract the intermediate representation- data flow, which is also the base of pipeline partition. The data flow and the relations between all variables are modeled with *directed graph* [97].

Only the causalities between variables are considered instead of the exact operations, and the exact operations are processed in the phase of combinational logic generation.

A directed graph D_1 consists of vertices $V(D_1)$ and arcs $A(D_1)$. In the context of Verilog source code, vertices $V(D_1)$ represent variables, such as primary inputs, primary outputs, registers, wires, etc; and arcs $A(D_1)$ represents the causal relation between different vertices. Suppose a causal relation (a, b) exists, where a, b are variables defined in Verilog HDL specification, it is then considered that b is dependent on a . Two dependencies are defined: one is data-dependent, where the value a is used in the computation of b ; another is control-dependent, where the value of a is used in the conditional branches that determine the value of b . The causal relations from Verilog HDL specification are inferred under the following rules:

- 1) Continuous assignments. The causal relations are established from the variables on the right-hand side to the variables on the left-hand side. For example, the continuous assignment in Example 1 establishes arc ba from vertex b to vertex a , and arc ca from vertex c to vertex a , ignoring the operation of addition the

Fig. 3.6 (a) Directed graph for Example 1, (b) Directed graph for Example 2

corresponding directed graph is shown in Fig. 3.6(a).

Example 1

assign $a=b+c$;

2) Procedural assignments. The rules of extracting causal relations in procedural assignments are the same as that in continuous assignments.

3) Procedural assignments in conditional branches. The rules of extracting this kind of causal relations must include the conditional expression other than the procedural assignments, in other words, create directed graph with the variables of continuous assignments on the right-hand side and the ones in the conditional

Example 2

```
1  case (a)
2      0:
3          if (b==1)
4              x=d;
5          else if (c==2)
6              y=e;
7          else
8              z=f;
9      1:
10         if (b==2)
11             x=d;
12         else if (c==3)
13             y=e;
14         else
15             z=f;
16  endcase
```


intermediate representation contains only the information of causal relations between variables but without operation types. However, it is currently informative enough for the next step-pipeline partition.

3.5.3 Pipeline Partition

This section describes the principle of partitioning pipelines based on the directed graph obtained from the data flow extraction. A directed graph D_2 is derived from D_1 , such that the arcs $A(D_2)$ represent the communication *channels* between the various components, and vertices $V(D_2)$ represent the variables that need handshake, namely registers, and $V(D_2) \subseteq V(D_1)$. The primary inputs, primary outputs and the variables whose assignments are triggered by the clock signal in sync Verilog code are included into the handshake variable list for the following considerations:

- a) Maintain the equivalence of a data flow from sync pipelines to synthesized async pipelines. This is realized by including clocked variables as part of the handshake list. Any data synchronizations specified in the sync circuits will be maintained in the async circuits. Hence, the equivalence of data flow between sync circuits and async circuits is valid.
- b) Keep pipeline alive to guarantee that a pipeline ring/loop can alternate between Valid and Null data. This is done by including primary input and output variables in the handshake list to avoid the deadlock caused by the external environment.

On the base of handshake variables of primary input, output and clocked variables, the pipeline partition algorithm is proposed and depicted in Fig. 3.7, which is in PERL language to derive the exact boundaries of pipelines.

3.5.4 Insert Null Pipeline Stages

The purpose of inserting null pipeline stages is to provide space for data alternating between valid and null. To ensure that the equivalent data flows between compiled QDI circuits and sync circuits, the initialized values of the two should be the same. However, the compiled QDI circuits have set and reset phases for evaluating and clearing data path respectively so an additional null stage is required between two initialized stages. The condition of inserting a null stage is that both the current and the preceding stages are initialized to valid data. The circuits depicted in Fig. 3.8(a)-(b) is an example of the Null pipeline stage insert. Fig. 3.8(a) is the standard sync pipeline stages, and the three stages are initialized to V1, V2, and V3 respectively. The three compiled pipeline stages are reset to the corresponding valid data in the dual-rail

Fig. 3.8 (a) A synchronous pipeline, (b) Compiled asynchronous pipelines with inserted Null pipeline stages

format in Fig. 3.8(b). Thereafter, two pipeline stages are inserted in between them, and the inserted pipeline stages are reset to Null, separating adjacent two valid data.

3.5.5 Low Overhead Initialization of Linear Pipelines

From the previous section, it can be observed that the overhead of the inserted Null pipeline stages is relatively large. Now a lower overhead initialization of linear pipeline stages is proposed. The general idea is to load initial data by the latch with load function. One low overhead initialization of linear pipeline circuit is depicted in Fig. 3.9(b). There is one load token pass from the right side to the left side through the

Fig. 3.9 (a) Linear pipelines with inserted Null pipeline stages, (b) Low overhead linear pipeline with loading initial values

3.5.6 Compiled Circuit Patterns

The complete ‘if-else’ clause is synthesized as a single async pipeline. For easy view of the circuit structure, the coding style of separation of the sequential circuits and combinational circuits is preferred (see the source code in *Example 3*). Any other coding style (such as the mixing code of sequential circuits and combinational circuits together in *always* blocks) will be automatically transformed into the preferred coding style in *Example 3*. During synthesis, sequential circuits are mapped to async registers

Example 3:

// sequential circuit

always @ (posedge CLK or negedge NRST)

 if(!NRST) A<=Constant;

 else A <=A_temp;

// combinational circuit

always @ (A1 or A2 or ...)

 if (Condition1) A_temp= f(A1, A2, ...);

 else if (Condition2) A_temp= f(B1, B2, ...);

 else A_temp = f(C1, C2, ...);

Fig. 3.11 Synthesized pipeline of complete 'if-else' clause with conditional operations embedded inside the combinational circuit

with handshake circuits, and the combinational circuits will be synthesized, converted to dual-rail netlist. The synthesized pipeline structure is depicted in Fig. 3.11.

2) Synthesis of Incomplete 'if-else' Clause

The incomplete 'if-else' clause is viewed as the state-holding or feedback functionality (see the original source code in *Example 4*). Similar to the synthesis of the complete 'if-else' clauses, the separate codes of the sequential and combinational circuits are generated. However, the complete 'if-else' branch is different as it is

Example 4:

//original source code

```
always @(posedge CLK or negedge NRST)
  if(!NRST)      A<=Constant;
  else if (Condition1) A<= f(A1, A2, ...);
  else if (Condition2) A<= f(B1, B2, ...);
```

//processed code

//generated sequential circuit

```
always @(posedge CLK or negedge NRST)
  if(!NRST)  A<=Constant;
  else      A<=A_temp;
```

//generated sequential circuit

```
always @( A_feedback or A1 or A2 or ...)
  if      (Condition1)  A_temp = f(A1, A2, ...);
  else if (Condition2)  A_temp = f(B1, B2, ...);
  else                A_temp = A_feedback;
```

Fig. 3.12 Synthesized pipeline of incomplete ‘if-else’ clause with conditional operations embedded inside the combinational circuit

completed by adding the omitted assignment (e.g. $A_temp = A_feedback$ in *Example*

```
always @(posedge CLK or negedge NRST)
    if(!NRST)          state<=Constant;
    else                state<=next_state;
```

```
always @(state or Condition1)
    case(state)
        Branch1: next_state= f(state);
        Branch2: if (Condition1) next_state= f1(state);
                  else          next_state= f2(state);
        Branch3: next_state= f3(state);
        ...
        Default: next_state= fn(state);
    endcase
```

4). Thus, the generated combinational circuits will be similar to that of complete ‘if-else’ clause, but the pipelines are different since a feedback path is added to maintain the previous data. The synthesized circuit structure is depicted in Fig. 3.12.

(a)

(b)

```
// generated guard combinational circuit
Always @ (state_feedback)
    if (state_feedback == Branch2)
        guard =1;
    else
        guard =0;
```

(c)

Fig. 3.14 (a) Implementation of guard controller, (b) Implementation of default value cell, (c) Source code of the generated guard combinational circuit

3) Synthesis of 'Case' Clause

'Case' is employed to model state machine in the proposed synthesis, such as the

source code in *Example 5*. From the previous synthesis of complete and incomplete ‘if-else’, all the primary input variables are assumed to be updated to a useful/meaningful data for every operation. If this happens to the ‘case’ clause, the code in *Example 5* will be synthesized to the pipelines as depicted in Fig. 3.12. However, the situation in ‘case’ clause may be encountered that some variables are conditionally required (such as *Condition1* in *Example 5* is explicitly required only in *Branch2*). This is not a problem for the sync circuits as the variables/signals are either logic 1 or logic 0. However, it is problematic in the async dual-rail circuits if the variable needs manual triggering, as the variables/signals in the dual-rail async circuits alternate between valid and null. In this case, the Verilog code in *Example 5* is synthesized to the pipelines in Fig. 3.13, where the guard controller is to pass the primary input when ‘guard’ is true, otherwise, it will generate a default value to the combinational logic. The implementation of the guard controller and the source code of ‘*generated guard combinational circuit*’ are depicted in Fig. 3.14.

3.6 Finite Impulse Response Filter Design Example

In this section, the proposed synthesis is validated by synthesizing an 8-bit 8-tap async QDI FIR filter from the sync specification - one of the critical building blocks for our targeted wireless sensor network. For completeness, the circuits are implemented in IBM 130nm CMOS and simulated in Synopsys Nanosim using spice models, and at the nominal $V_{DD} = 1.2V$. The transistor sizing of all library cells are 560nm/120nm for PMOS and 280nm/120nm for NMOS, except for the weak keepers

Fig. 3.15 Block diagram of synthesized 8-bit 8-tap async QDI FIR filter

where 160nm/500nm for PMOS and 160nm/1000nm for NMOS. Fig. 3.15 depicts the

block diagram of the synthesized QDI FIR filter. The synthesized FIR filter operates as follows. State Machine will generate the write address and write enable signal on *SRAM_Addr* and *R/W* respectively to accept input *Data_In*, thereafter, generating *SRAM_Addr*, *ROM_Addr*, *R/W* signals to read out data and coefficient which they are held by Latch 3 and Latch 2 respectively, and are fed to the Multiplier. The output of the Multiplier is held in Latch 5 and is accumulated by the Adder. The layout view of the synthesized FIR filter is shown in Fig. 3.16.

The performance of the synthesized QDI FIR filter is evaluated and the result is tabulated in Table 3.1. The filter comprises of 39,181 transistors and it can dissipate 3.7mW at the sample rate of 5MHz (200ns per operation). From the breakdown of the

Table 3.1 Overhead (power dissipation (P, mW), transistor-count (TC) and delay (t, ns)) breakdown for the synthesized Asynchronous QDI FIR Filter @ 1.2V, IBM 130nm CMOS

	Handshake Circuits	Combinational Circuits	Total
<i>P</i>	44.5%	55.5%	100% (3.7)
<i>TC</i>	34.5%	65.5%	100% (39181)
<i>t</i>	200 ns per operation		

Table 3.2 The power dissipation (P), transistor-count (TC), delay (t) and a composite metric of $P \times TC \times t$ of the lowest power point for Asynchronous QDI FIR Filter @ 1.2V, IBM 130nm CMOS

	Power, <i>P</i>	<i>TC</i>	Delay, <i>t</i>	$P \times TC \times t$
NCL	2.51×	1.39×	0.61×	2.13×
Proposed	1×	1×	1×	1×

overhead, it can be observed that the combinational circuits take up a large portion of the overall overhead ($\sim 60\%$), ensuing potential optimizations.

The async QDI FIR filter synthesized by the proposed synthesis tool is benchmarked with NCL design methodology. The results, tabulated in Table 3.2, show the proposed async QDI FIR filter is of $2.51\times$ power efficiency, $1.39\times$ transistor count efficient, and reasonable speed overhead; but $1.53\times$ efficiency in energy ($P\times t$) consumption per operation (not shown in Table 3.2), $2.13\times$ efficiency in composite metric of $P\times TC\times t$.

Fig. 3.16 The layout view of the synthesized Async QDI FIR Filter

4.

Proposed Quasi-Delay-Insensitive Data Path Optimization using Microcell-Interleaving Genetic Algorithm

4.1 Introduction

The overhead of async QDI combinational circuits (i.e. the data path) consist of two main aspects. First, a near doubling of the hardware compared to single-rail designs comes from its dual-rail modality, and overhead for this part usually cannot be mitigated due to the QDI attribute requirement. Second, a larger additional overhead arises from its DCD circuits. From a broad perspective, the power dissipation and IC area overheads of the DCD circuits are the main obstacles of employing dual-rail combinational circuits. If this overhead is mitigated, the efficacy of the data-control decomposition QDI realizations could be significantly improved.

In this chapter, a design approach to reduce the aforesaid overhead is proposed by means of a novel microcell-interleaving approach and a Microcell-Interleaving Genetic Algorithm. Here a “microcell” is defined as an async dual-rail gate, capable of executing basic operations such as AND/NAND, OR/NOR, etc. Our proposed

design approach complements the NCL_X design methodology, and there are several novelties and contributions in this chapter. First, our proposed design approach essentially involves interleaving input-complete microcells with non-input-complete microcells to enable hardware to be shared without compromising the async QDI attribute. This microcell-interleaving approach is beneficial to reduce the number of CD circuits, thereby reducing the overall hardware overhead. Second, unlike the NCL_X design methodology that is targeted for NCL microcells, our proposed design approach is applicable to many implementations of microcells, including non-input-complete microcells (NCL, DCVSL, etc.), fully and partially input-complete microcells (DIMS, Dynamic Strong Indicating (DSI), etc.). This attribute would widen the scope of circuit implementation and its ensuing optimization. Third, the performance limit (upper/lower bounds) of async QDI circuits is comprehensively analyzed in terms of power dissipation, transistor-count and delay, and the optimal point is derived based on the given metrics. Fourth, the proposed MIGA, leveraging on the well-established genetic algorithm and its associated fitness functions, enables a faster optimization process to achieve the possible optimal metrics and provides a practical approach to the actual circuit realization.

Based on the DCVSL and DSI microcells as examples, the proposed design approach is validated, including a full exploration of the performance limit/profile of an async QDI circuit. Based on several well-accepted ISCAS-85 benchmark circuits, the proposed design approach is shown highly competitive when benchmarked against

the competing DCVSL, NCL and DSI ($1.77\times$, $1.4\times$, $1.58\times$ better in power dissipation, area, a composite metric of power $\times$ area $\times$ delay respectively, and with a reasonable

Fig. 4.1 (a) The circuit schematic of a DCVSL async dual-rail OR/NOR microcell and estimated overhead for CD circuit, (b) The circuit schematic of a DSI dual-rail OR/NOR microcell

delay increase). Finally, the efficiency of the proposed MIGA is demonstrated by comparing with the greedy algorithm and dynamic programming.

4.2 Proposed Microcell-Interleaving Approach

The efficacy of async QDI pipelines strongly depends on the types of the microcells adopted. In general, non-input-complete microcells are widely adopted (over input-complete microcells) in async QDI pipelines due to their higher speed, lower power dissipation and smaller IC area. Despite this, the area and energy overhead of DCD are considerably larger and cannot be ignored especially if the functional circuits in the async QDI circuit are highly complex. In Table 4.1, the power dissipation, transistor-count and delay of 6 basic non-input-complete and input-complete microcells are characterized; the former is based on DCVSL and the latter is based on DSI (with transistor-level optimization). For illustration purposes, the schematics of dual-rail OR/NOR microcells embodying DCVSL and DSI are depicted in Fig. 4.1(a) and Fig. 4.1(b) respectively. The readings (in Table 4.1) in DCVSL microcells include a local CD circuit to maintain the QDI attribute. Particularly, the local CD circuit is a 2-input OR microcell and a quarter of a 4-input C-Muller microcell (which is shared by 4 DCVSL microcells to mitigate the power/area overhead). When the DCVSL microcells, embodying a local CD circuit, are compared against the DSI microcells, on average, the latter appear highly uncompetitive in speed (by $\sim 3.1\times$ worse) but comparable in terms of power dissipation (by $\sim 0.04\times$ better) and transistor-count (by $\sim 0.16\times$ worse). Nonetheless,

these comparisons allude to possible overhead optimization and such optimization will be in turn described based on the proposed interleaved microcell approach. Please note that DCVSL and DSI microcells tabulated in Table 4.1 are necessary for efficacious applications of the proposed approach and algorithm (see later).

The fundamental advantage of the proposed microcell-interleaving approach is to reduce the number of local CD circuits and yet maintain the QDI attribute. Fig. 4.2 depicts a pipeline structure where a DCVSL microcell is interleaved with a DSI microcell; the DSI microcell not only computes its logic operation but also detects the output validity/nullity of its interleaved DCVSL microcell. In this interleaving structure, only a local CD circuit is required to integrate into the DSI microcell (but not to the DCVSL microcell), thereby reducing the overall hardware overhead in the RCD_i and DCD_i . The amount of power dissipation and area reduction is largely in line with the number of reduced local CD circuits. For example, the power dissipation overhead due to the local CD circuit (for each DCVSL microcell) is $1.1 \mu\text{W}$ @

Table 4.1 The power, transistor-count and delay characteristics of several dual-rail microcells based on DCVSL and DSI @ 1.2V, IBM 130nm CMOS

	Power, $\mu\text{W}@100\text{MHz}$		Transistor-count		Delay, ns	
	DCVSL*	DSI	DCVSL*	DSI	DCVSL*	DSI
2-input OR/NOR	2.2	1.5	23	21	0.14	0.32
2-input AND/NAND	2.2	1.5	23	21	0.15	0.32
2-input XOR/XNOR	2.3	1.6	27	24	0.17	0.33
3-input OA/OAI	2.3	2.9	25	34	0.15	0.93
3-input AO/AOI	2.3	2.9	25	34	0.15	0.94
3-input Carry/ICarry	2.3	2.9	27	38	0.16	0.87
Average	2.3	2.2	25	29	0.15	0.62

*: include a CD circuit

100MHz. By using a 3-input OA/OAI DSI microcell (connected to three DCVSL microcells), three local CD circuits can be reduced, namely, $3.3\mu\text{W}$ ($3 \times 1.1\mu\text{W}$) power dissipation is reduced. Although the DSI microcell dissipates $1.7\mu\text{W}$ ($2.9\mu\text{W} - (2.3\mu\text{W} - 1.1\mu\text{W})$) of power dissipation more than the corresponding DCVSL microcell, the overall power dissipation saving is still significantly large at $1.6\mu\text{W}$ ($3.3\mu\text{W} - 1.7\mu\text{W}$). Similarly, the transistor-count can be minimized by reducing the local CD circuits. With respect to the speed of the optimized circuits, microcell-interleaving circuits usually features a lower speed attribute when compared to the fully non-input-complete circuits. This is mainly due to the already slow speed attribute of input-complete microcells.

To appreciate our proposed microcell-interleaving approach and the effects of the different interleaving strategies, a circuit depicted in Fig. 4.3(a)-(d) is considered. This circuit has three paths: $Path_1$, $Path_2$ and $Path_3$ where $Path_1$ consists of microcell

Fig. 4.2 The proposed asynchronous pipeline by adopting microcell-interleaving approach with DCVSL and DSI microcells

1 to 9, $Path_2$ consists of microcell 10 to 15, and $Path_3$ consists of microcell 16 to 20.

The rectangle comprises only microcells (DCVSL or DSI), while the oval comprises microcells (DCVSL or DSI) with a local CD circuit. All the microcells are assumed to be 2-input microcells. For simplicity, not all the inputs or wires are shown. The objective here is assumed to have achieved the lowest power dissipation by using our proposed approach.

In order to minimize the power dissipation (largely due to fewer CD circuits), the microcell-interleaving optimizations are intuitively arranged into four stages, in Fig. 4.3(a), Fig. 4.3(b), Fig. 4.3(c) and Fig. 4.3(d) respectively. Fig. 4.3(a) is the initial circuit where the default microcells are the DCVSL microcells. All the microcells in Fig. 3.4(a) are oval shape except the primary output microcells. This is because the DCVSL microcells are non-input-complete as the local CD circuits are required to be integrated to detect the validity/nullity of internal wires, hence fulfilling the QDI attribute. The primary output microcells do not require local CD circuits as their primary outputs, which are checked by the RCD of the following register. Hence, a total of 17 CD circuits are required for the circuit in Fig. 4.3(a). Next microcell-interleaving approach is applied to the circuits in Fig. 4.3(a) with DSI microcells and integrate appropriate CD circuits are integrated to obtain a new circuit configuration as depicted in Fig. 4.3(b). In this example, the DCVSL microcells $DCVSL_1$, $DCVSL_3$, $DCVSL_5$, $DCVSL_7$, $DCVSL_{10}$, $DCVSL_{12}$, $DCVSL_{14}$, $DCVSL_{16}$, $DCVSL_{18}$ do not require any local CD circuit. While all the output wires of DSI microcells are connected to DCVSL microcells, local CD circuits are integrated into

Fig. 4.3 Microcell interconnection: (a) Initial interconnection, (b) Interleaving interconnections, (c) Interleaving connection of zero CD circuits, and (d) Interleaving connection without redundant wire checking

DSI microcells to preserve the QDI attribute. In this case, there are only 8 CD circuits

and a reduction of 9 CD circuits as compared to the initial configuration in Fig. 4.3(a).

For further power dissipation reduction, all the CD circuits are eliminated to achieve a new updated configuration as depicted in Fig. 4.3(c). In this updated configuration, all the validity/nullity of the internal wires are checked by the DSI microcells and no CD circuits are required.

Note that even without the CD circuits, the combination of DCVSL and DSI microcells may not be the lowest power dissipation, as redundant wire checking will also lead to power dissipation. In Fig. 4.3(c), it is observed that the output wire of $DCVSL_{10}$ is checked twice by the DSI_{11} and DSI_{18} . In this case, the redundant checking can be easily removed by replacing DSI_{11} with $DCVSL_{11}$, or by replacing DSI_{18} with $DCVSL_{18}$. However, for the latter replacement, the output wire of DSI_{17} is not internally checked, and a CD circuit should be integrated into DSI_{17} . Therefore, replacing DSI_{11} with $DCVSL_{11}$ is a better choice. The same approach can be applied to the output wire of $DCVSL_{16}$, DSI_4 , and DSI_5 ; and DSI_{17} , DSI_5 , DSI_6 can be replaced with $DCVSL_{17}$, $DCVSL_5$, $DCVSL_6$ respectively. The final optimized configuration is depicted in Fig. 4.3(d), which probably has the lowest power dissipation among different DCVSL and DSI combinations.

The above example demonstrates the optimization process of the proposed microcell-interleaving approach. With the increase in complexity of the circuits (number of microcells), the number of combinations between non-input-complete microcells (such as DCVSL, NCL, etc.) and input-complete microcells (such as DSI), will increase exponentially. In such case, it is impossible to manually arrange the

locations of different microcells to achieve the lowest power dissipation combination. Hence, an optimization algorithm by employing the genetic algorithm is proposed to stochastically search for a combination which yield the best result. The details of the implementation will be explained later but a formal evaluation for the microcell-interleaving approach will first be investigated.

4.3 Performance Profile Analysis for the Proposed Microcell-Interleaving Approach

In this section, the complete performance profiles are investigated in terms of power dissipation, IC area (largely based on the number of transistors) and delay of async QDI circuits. Other attributes, such as the dependency of power and delay profiles, can be explored by the proposed MIGA. In order to study the trend of profiles, the size of circuits (number of microcells) is assumed to be so large that the impact of a single microcell on the trend is negligible.

4.3.1 Power Dissipation

For a QDI circuit, the power analysis is classified on the basis of two categories: individual-microcell analysis and group-microcell analysis

1) *Individual-microcell Analysis*. The individual-microcell analysis is first considered by using the i^{th} non-input-complete microcell $G_{NIC,i}$ and its corresponding input-complete microcell $G_{IC,i}$. The number of primary inputs (a dual-rail input is considered as one primary input) of the microcell is defined as I_i . The power dissipation of $G_{NIC,i}$ and $G_{IC,i}$ are denoted as $P_{NIC,i}$ and $P_{IC,i}$ respectively, and their

power difference is ΔP_i . The power dissipation of one CD circuit for detecting/acknowledging a primary input (of $G_{NIC,i}$) is further assumed to be P_{CD} .

Thus, an input threshold n_i of the microcell can be found in (4.1).

$$n_i = \frac{P_{IC,i} - P_{NIC,i}}{P_{CD}} = \frac{\Delta P_i}{P_{CD}} \quad (4.1)$$

The input threshold integer n_i will be grouped in one of the three domains in (4.2a) to (4.2c), depending on the value of the primary input I_i .

$$\left\{ \begin{array}{l} n_i > I_i \\ 0 \leq n_i \leq I_i \\ n_i < 0 \end{array} \right. \quad \begin{array}{l} (4.2a) \\ (4.2b) \\ (4.2c) \end{array}$$

In the domain (4.2a), ΔP_i is always > 0 , as $G_{NIC,i}$, which integrates with CD circuits, will be more power-efficient than $G_{IC,i}$. Conversely, in the domain (4.2c), ΔP_i is always < 0 , this implies that $G_{IC,i}$ is more power-efficient than $G_{NIC,i}$ (although this scenario is unlikely to be true due to the additional overhead for input-completeness in $G_{NIC,i}$). In the domain (4.2b), it is of interest to find an optimal combination of microcells as some of the CD circuits can be shared in the actual circuit realization (at the circuit-level), those yet unchecked primary inputs/wires, denoted as $N_{WUC,i}$, will then need to be acknowledged by other CD circuits. If $N_{WUC,i}$ is $<$ input threshold n_i in $G_{NIC,i}$, this scenario indicates that a replacement of their corresponding $G_{IC,i}$ would not be power-efficient. If $N_{WUC,i}$ is $>$ input threshold n_i in $G_{NIC,i}$, this scenario indicates that a replacement of its $G_{IC,i}$ would be power-efficient. In a complex circuit where many microcells are connected and some primary inputs could be shared, the

replacement of the microcells and their associated CD circuits will affect the number of yet undetectable primary inputs of other microcells. In other words, it would be possible that a replacement of the i^{th} microcell with $N_{WUC,i} > n_i$ could cause $N_{WUC,k} < n_k$ in k^{th} microcell in a complex circuit.

The analysis of the individual microcell in a completion detection circuits is performed at the inputs, as it is easier to appreciate the optimization space in this fashion. Nevertheless, from an overall circuit perspective, the analysis is performed on output instead

2) *Group-microcell Analysis.* Now the group-microcell analysis of the overall power dissipation are considered, where microcells located in the domain (4.2b) will be of main interest. The power dissipation profile of a QDI circuit embodying many microcells is first depicted in Fig. 4.4. The abscissa represents the percentage of the

Fig. 4.4 Overall power dissipation profile of all microcell combinations satisfying QDI attribute

number of input-complete microcells over the total number of microcells (including input-complete and non-input-complete microcells), whereas the ordinate represents power dissipation. The shaded region indicates the possible microcell combinations satisfying the QDI attribute. The power $P_{0\%}$ is the power dissipated in the QDI circuit embodying fully non-input-complete microcells, and $P_{100\%}$ is the converse; $P_{0\%} > P_{100\%}$. The lower contour (line $P_{0\%}$ to $P_{j\%}$ to $P_{m\%}$ to $P_{100\%}$) indicates the lower power boundary for their respective percentage of using the input-complete microcells. Similarly, the upper contour (line $P_{0\%}$ to $P_{100\%}$) indicates the upper power boundary.

The abovementioned power profile is now explained in detail as follows. The power dissipation of the QDI circuit is first modeled as:

$$P_{j\%}(\alpha) \cong M_{CD,\alpha} \cdot P_{CD} + \sum_{l=1}^{M_{IC}} P_{IC,l}(\alpha) + \sum_{o=1}^{M_{NIC}} P_{NIC,o}(\alpha) \quad (4.3)$$

where α is the arbitrary point at the $j\%$ (within the lower and upper power dissipation boundaries); $M_{CD,\alpha}$ is the total number of CD circuits used at the point α ; M_{IC} is the total number of input-complete microcells at $j\%$; M_{NIC} is the total number of non-input-complete microcells at $j\%$;

$$j\% = \frac{M_{IC}}{M_{IC} + M_{NIC}} \times 100\% . \quad (4.4)$$

Within the same percentage of input-complete microcells, e.g. at $j\%$ in Fig. 4.4, it is possible to have many microcell combinations (with different α points), the minimum point is defined as $P_{j\%}$ as expressed in (4.5).

$$P_{j\%} = \min[P_{j\%}(0), P_{j\%}(1), P_{j\%}(2), \dots] \quad (4.5)$$

The minimum point $P_{j\%}$ can be achieved by exhaustively evaluating the different

microcell combinations (at different α points). When $j = m$, $P_{m\%}$ is the lowest power point among all the combinations, which is the main focus in our analysis. Considering the lower power boundary, the power dissipation is initially reduced from $P_{0\%}$ to $P_{m\%}$, and the rate of power reduction satisfies:

$$\frac{P_{j\%} - P_{(j-\Delta)\%}}{\Delta\%} \leq \frac{P_{(j+\Delta)\%} - P_{j\%}}{\Delta\%} < 0 \quad \text{where} \quad 0 < j < m \quad (4.6)$$

This situation is easy to appreciate as the replacement of non-input-complete microcells by the input-complete microcells reduces the number of the CD circuits. The power reduction per input-complete microcell is more significant initially because each replacement could reduce as many CD circuits as possible. After $m\%$, the overall power dissipation will increase if the non-input-complete microcells continued to be replaced with the input-complete microcells, and the rate of power increased can be expressed as follows:

$$\frac{P_{(j+\Delta)\%} - P_{j\%}}{\Delta\%} \geq \frac{P_{j\%} - P_{(j-\Delta)\%}}{\Delta\%} > 0 \quad \text{where} \quad m < j < 100 \quad (4.7)$$

In this case, the lowest power point $P_{m\%}$ in Fig. 4.4 will satisfy the condition in (4.8) as follows:

$$\frac{P_{(m-\Delta)\%} - P_{(m)\%}}{-\Delta\%} < 0 < \frac{P_{(m+\Delta)\%} - P_{(m)\%}}{\Delta\%} \quad (4.8)$$

For the number of CD circuits, M_{CD} , at the lowest power dissipation point $P_{m\%}$, the following theorem is derived:

Theorem 1: The number of CD circuits M_{CD} of the lowest power dissipation point $P_{m\%}$ satisfies:

$$M_{CD} \leq \sum_{i=1}^{M_{NIC}} \min(I_i, \max(\lfloor n_i \rfloor, 0)) \quad (4.9)$$

Proof: If $P_{m\%}$ is the lowest power dissipation point, it implies that further power dissipation reduction is impossible through replacing non-input-microcell and CD circuits with its corresponding input-complete microcell. Then, for each non-input-complete microcell, it is found that the number of unchecked wires is smaller than or equal to the threshold n_i , namely, $N_{WUC,i} \leq \max(\lfloor n_i \rfloor, 0)$, where $1 \leq i \leq M_{NIC}$. Obviously, the number of unchecked wires $N_{WUC,i}$ is smaller than or equal to the number of primary input, $N_{WUC,i} \leq I_i$, where $1 \leq i \leq M_{NIC}$, hence, $N_{WUC,i} \leq \min(I_i, \max(\lfloor n_i \rfloor, 0))$ can be obtained, where $1 \leq i \leq M_{NIC}$. Therefore, the number of CD circuits required for ensuring QDI attribute is:

$$M_{CD} = \sum_{i=1}^{M_{NIC}} N_{WUC,i} \leq \sum_{i=1}^{M_{NIC}} \min(I_i, \max(\lfloor n_i \rfloor, 0)). \quad \blacksquare$$

Furthermore, if the 2-input microcells in Table 4.1 are taken into consideration, for example AND, OR, XOR, it can be observed that the n_i for each microcell satisfies $0 < n_i < 1$. Thus, the following corollary can be obtained, which can be employed to accelerate the convergence of the lowest power dissipation point searching process.

Corollary 1: If $0 < n_i < 1$, where $1 \leq i \leq M_{NIC}$, the lowest power dissipation point $P_{m\%}$ has the attribute: $M_{CD} = 0$.

Proof: According to the Theorem 1, $M_{CD} \leq 0$ can be obtained if $0 < n_i < 1$. Since M_{CD} cannot be negative, so $M_{CD} = 0$ ■

For completeness, the upper power dissipation boundary is also presented, which connects $P_{0\%}$ and $P_{100\%}$. The upper power dissipation boundary is a near-straight, concave-down or concave-up line, depending on the circuit complexity and structure. Since only the region $0 \leq n_i \leq I_i$ is initially considered for threshold n_i , the replacement of any combination of non-input-complete microcells and CD circuits with the corresponding input-complete microcells will slightly reduce the power dissipation. After some of the non-input-complete microcells have been replaced, which reduce power dissipation marginally, a certain combination that satisfies $N_{WUC,i} < n_i$ will appear. In the next replacement attempt, these combinations will first be replaced and the overall power dissipation will be marginally increased. The amount of increased/decreased power dissipation will determine the contour of the upper power dissipation boundary.

4.3.2 Transistor-Count

With respect to the profile of IC area/transistor-count in the proposed microcell-interleaving approach, the analysis is very similar to that of the power dissipation. For clarity, some key equations are formulated as follows.

$$q_i = \frac{TC_{IC,i} - TC_{NIC,i}}{TC_{CD}} \quad (4.10)$$

where q_i is transistor-count threshold for i th microcell;

$TC_{IC,i}$ is the transistor-count of $G_{IC,i}$;

$TC_{NIC,i}$ is the transistor-count of $G_{NIC,i}$;

TC_{CD} is the transistor-count of a single CD circuit.

The threshold, q_i , also allocates in the three domains as follows:

$$\left\{ \begin{array}{l} q_i > I_i \end{array} \right. \quad (4.11a)$$

$$\left\{ \begin{array}{l} 0 \leq q_i \leq I_i \end{array} \right. \quad (4.11b)$$

$$\left\{ \begin{array}{l} q_i < 0 \end{array} \right. \quad (4.11c)$$

The transistor-count at percentage $r\%$ is modeled as:

$$TC_{r\%}(\beta) \cong M_{CD,\beta} \cdot TC_{CD} + \sum_{s=1}^{M_{IC}} TC_{IC,s}(\beta) + \sum_{t=1}^{M_{NIC}} TC_{NIC,t}(\beta) \quad (4.12)$$

where β is the arbitrary point at the $r\%$ (within the lower and upper transistor-count boundaries); $M_{CD,\beta}$ is the total number of CD circuits used at the point β ; M_{IC} is the total number of input-complete microcells at $r\%$; M_{NIC} is the total number of non-input-complete microcells at $r\%$;

$$r\% = \frac{M_{IC}}{M_{IC} + M_{NIC}} \times 100\% . \quad (4.13)$$

The lowest transistor-count point, $TC_{u\%}$, will satisfy the condition as follows:

Fig. 4.5 Overall delay profile of all microcell combinations satisfying QDI attribute

$$\frac{TC_{(u-\Delta)\%} - TC_{(u)\%}}{-\Delta\%} < 0 < \frac{TC_{(u+\Delta)\%} - TC_{(u)\%}}{\Delta\%} \quad (4.14)$$

According to our analysis, the profile of IC area/transistor-count is (almost) the same as that of power dissipation, because the reduction of power dissipation and transistor-count is by reducing the number of CD circuits.

4.3.3 Delay

Next the delay performance profile is analyzed. For clarity and consistency, the delay is also measured by the percentage of replacement, similar to the power dissipation profile.

The delay profile is depicted in Fig. 4.5, and it is a region surrounded by the lower delay boundary and upper delay boundary. The lower delay boundary begins from $D_{0\%}$, and passes through $DL_{v\%}$ and $D_{z\%}$, and finally reaches $D_{100\%}$. The line connects the points $(D_{0\%}, DU_{v\%}, D_{y\%}, D_{100\%})$ constitutes the upper delay boundary. The shaded region contains the delay of all combinations satisfying QDI attribute. $D_{0\%}$ is the delay of fully non-input-complete microcells whereas $D_{100\%}$ is the delay of fully input-complete microcells, and $D_{0\%} < D_{100\%}$.

The delay of the QDI circuit in a single pipeline can be computed as the sum of forward and backward delays (local cycle time) in (4.15), and considered as a reasonable measure of performance for a pipeline stage. However, this measure may be of limited value to complex sequential (multiple pipeline) async circuits, whose analysis could be extended by using reported approaches [98-100].

$$D_{v\%}(\delta) \cong D_{F,v\%}(\delta) + D_{B,v\%}(\delta) \quad (4.15)$$

where δ is the arbitrary point at the $v\%$ (within the lower and upper delay boundaries); $D_{F,v\%}$ is the forward delay at point δ of $v\%$; $D_{B,v\%}$ is the backward delay at point δ of $v\%$;

$$v\% = \frac{M_{IC}}{M_{IC} + M_{NIC}} \times 100\% .$$

The forward delay is defined as the delay from Req_i to $Input_{i+1}$, through $Register_i$ and combinational circuit, provided $Input_i$ is ready. The backward delay is defined as the delay from $Input_{i+1}$ to Req_i , through $Register_{i+1}$ and RCD_{i+1} .

$$\text{Hence} \quad DL_{v\%} = \min[D_{v\%}(0), D_{v\%}(1), D_{v\%}(2), \dots] \quad (4.16)$$

$$DU_{v\%} = \max[D_{v\%}(0), D_{v\%}(1), D_{v\%}(2), \dots] \quad (4.17)$$

On the lower delay boundary, $D_{z\%}$ is the turning point of delay increasing, in other words, the lower delay boundary satisfy the condition as follows:

$$\begin{cases} \frac{D_{v\%} - D_{(v-\Delta)\%}}{\Delta\%} = \frac{D_{(v+\Delta)\%} - D_{v\%}}{\Delta\%} = 0 & \text{where } 0 < v < z \\ \frac{D_{(v+\Delta)\%} - D_{v\%}}{\Delta\%} \geq \frac{D_{v\%} - D_{(v-\Delta)\%}}{\Delta\%} > 0 & \text{where } z < v < 100 \end{cases} \quad (4.18)$$

This situation can be appreciated as the replacement of non-input-complete microcells located at non-critical paths, then critical path.

Similar analysis can be applied to the upper delay boundary, where $D_{y\%}$ is the turning point and the upper boundary as follows:

$$\begin{cases} \frac{D_{v\%} - D_{(v-\Delta)\%}}{\Delta\%} \geq \frac{D_{(v+\Delta)\%} - D_{v\%}}{\Delta\%} > 0 & \text{where } 0 < v < y \\ \frac{D_{v\%} - D_{(v-\Delta)\%}}{\Delta\%} = \frac{D_{(v+\Delta)\%} - D_{v\%}}{\Delta\%} = 0 & \text{where } y < v < 100 \end{cases} \quad (4.19)$$

The reason is the converse of that for lower delay boundary, in other words, the replacement of non-input-complete microcell is initially located at critical path, then

non-critical paths.

4.4 Proposed Microcell-Interleaving Genetic Algorithm

The objective of this section is to search for the optimal combination of input-complete and non-input-complete microcells with the aim of achieving lowest power dissipation or the shortest delay, equivalent to point $P_{m\%}$ in Fig. 4.4 or point $DL_{v\%}$ in Fig. 4.5 respectively. In general, the lowest power dissipation point, $P_{m\%}$, is near to the smallest number of transistor-count point.

The most obvious way is to search for all possible combinations. However, as the number of combinations increases exponentially with the number of microcells in one circuit, this renders an exhaustive search virtually impossible for a circuit with a large number of microcells. Specifically, the search space (complexity) is N^M , where N and M are respectively the number of types of library cells and microcells in a given circuit. ($N=2$ in the experiments and $M\approx 400$ in the benchmark circuits herein). The search space in the experiment herein is hence $\sim 2^{400}$. Due to the gargantuan magnitude of the search space, some search algorithms, such as genetic algorithm [101-106], greedy algorithm [107-112], and dynamic programming [113-117], should be employed to search for the optimal solution.

The principle of genetic algorithm is to mimic the process of natural selection, where a population of candidate solutions to an optimization problem is evolved toward better solutions by selection, crossover and mutation. The genetic algorithm has the advantage of low possibility of being trapped at local optimal solutions

compared to the greedy algorithm. However computationally-wise, it is somewhat inefficient. The principle of greedy algorithm is to make the locally optimal choice at each stage with the objective of finding a global optimum. Although the greedy algorithm is computationally more efficiently than the genetic algorithm, it suffers from the high possibility of being trapped at a local optimal solution as the decisions made are not changeable. Dynamic programming breaks down the complex problem into a number of sub-problems and thereafter combines the sub-solutions to obtain an optimal solution and it usually not applicable to complex problems with optimal substructure. In view of the disadvantage and requirement of greedy algorithm and dynamic programming, the genetic algorithm is adopted in this chapter. The comparisons of the results (of the proposed approach) generated by genetic algorithm, greedy algorithm and dynamic programming will be presented later.

The pseudo-code of the proposed MIGA, which is largely based on the reported genetic algorithms, is depicted in Fig. 4.6. The input to the MIGA is a synthesized combinational circuit netlist whereas the output is the optimized microcell-interleaving QDI netlist. The **Preprocessing** operation will encode the (input-complete and non-input-complete) microcells of the input combinational circuit netlist into chromosome on the basis of their interconnections. First, the **Initial_population_generation** will randomly generate an initial population (300 in our simulations) of chromosomes. Then, the **Evaluation** process evaluates the fitness value of each chromosome in the population in terms of their power dissipation, transistor-count and delay. With higher fitness value proportional to higher selection

probability based on the roulette-wheel selection scheme [118-120], the **Selection** process selects a group of chromosomes to be the parent chromosomes. The purpose of the **Selection** process in MIGA is to exploit the optimal regions in the search space.

One-point **Crossover** operation [121-123] applies to two selected parent chromosomes which generate (produce) two offspring chromosomes with a crossover rate of 0.8. The **Crossover** operation is to exchange the information between two fit chromosomes in order to explore/combine the characteristics of different optimal regions in the search space. Bit-flip **Mutation** [124-125] applies to every gene of the

The proposed microcell-interleaving genetic algorithm (MIGA)

Input:

Netlist_input: standard single-rail combinational circuit netlist

01. Preprocessing:
02. formulate netlist *Connection*(Gate, Wire);
03. generate *Chromosome*(Gate) ;
04. Initial_population_generation:
05. for $i=1,2,\dots,number_of_population$
06. generate_initial_population(*Chromosome*(Gate));
07. end for
08. MIGA_main_loop: for $i=1,2,\dots,prescribed_generation$
09. Evaluation: for $j=1,2,\dots,number_of_population$
10. case(*optimization_target*);
11. Power: *fitness_value*= calculation_Power();
12. transistor_count: *fitness_value*= calculation_TC();
13. Delay: *fitness_value*= calculation_Delay();
14. end case
15. *Optimal_chromosome*=update_optimal_chromosome();
16. end for
17. Selection:
18. select(*population*);
19. New_population_generation:
20. for $j=1,2,\dots,number_of_new_children$
21. crossover();
22. end for
23. mutation();
24. end for
25. Mapping: *Netlist_optimized*=MapChromosomeToNetlist(*Optimal_chromosome*);

Output:

Netlist_optimized: optimal microcell-interleaving dual-rail netlist

Fig. 4.6 Pseudo-code of the proposed Microcell-Interleaving Genetic Algorithm

offspring chromosomes with a small probability of 0.01. This is to prevent the MIGA searching process from being trapped in the local optimal regions. After a new generation of population is produced, the processes of **Evaluation**, **Selection**, **Crossover**, and **Mutation** will repeat until the termination condition (500 generations in our simulations) is reached. Finally, the **Mapping** decodes the optimal chromosome with the highest fitness value to an output dual-rail QDI netlist.

In view of the pseudo code depicted in Fig. 4.6, the complexity (number of evaluation of fitness function) of the proposed MIGA can be obtained as the number of generations multiplied by the population (respectively 500 and 300 in our experiments, where the number of microcells is around 400). Hence, the efficiency of the proposed MIGA can be appreciated by obtaining the (near) global optimum in the large searching space of $\sim 2^{400}$ only with the computational complexity of 1.5×10^5 .

The combinational circuit netlist to chromosome encoding process and the fitness function formulation of the proposed MIGA will be elaborated in details in the following sections.

4.4.1 Genetic Coding of Combinational Circuit Netlist

The general chromosome is represented as:

$$Chromosome = \langle G_1 \ G_2 \ \dots \ G_M \rangle \quad (4.20)$$

where M is the number of microcells in the given circuit. In the string chromosome, all microcells are put into a string inconsequentially. For example, the combination of the microcells in Fig. 4.3(a) can be formulated as $Chro_A$ as follows:

$$Chro_A = \langle DCVSL_1 \ DCVSL_2 \ DCVSL_3 \ \dots \ DCVSL_{20} \rangle \quad (4.21)$$

Binary coding is adopted where a non-input-complete microcell will be represented by a gene value of ‘0’ and an input-complete microcell by a gene value of ‘1’. As a point of illustration, the configuration of the circuit in Fig. 4.3(b) is shown below:

$$Chro_A_b = <01010101001010101010> \quad (4.22)$$

With the proposed genetic coding, the circuit characteristics of a combinational circuit netlist can be effectively encoded in the chromosome in order for the proposed MIGA to search for an optimal combination of QDI netlist.

4.4.2 Fitness Function Formulation

For each chromosome, the power dissipation fitness function is the reciprocal of the total power dissipation of the CD circuits, input-complete microcells and non-input-complete microcells, expressed as follows:

$$Fitness(P) = \frac{1}{\sum_{i=1}^{M_{CD}} P_{CD,i} + \sum_{i=1}^{M_{IC}} P_{IC,i} + \sum_{i=1}^{M_{NIC}} P_{NIC,i}} \quad (4.23)$$

The transistor-count fitness function is the reciprocal of the total transistor-count of the CD circuits, input-complete microcells and non-input-complete microcells, shown as follows:

$$Fitness(TC) = \frac{1}{\sum_{i=1}^{M_{CD}} TC_{CD,i} + \sum_{i=1}^{M_{IC}} TC_{IC,i} + \sum_{i=1}^{M_{NIC}} TC_{NIC,i}} \quad (4.24)$$

where $M_{IC} = \sum_{i=1}^M G_i$, $M_{NIC} = \sum_{i=1}^M (1 - G_i)$, $M_{CD} = \sum_{i=1}^M H_i$ and H_i is obtained as follows:

$$H_i = \begin{cases} 0 & \text{if } \exists G_j, G_j \in (G_i \bullet), \text{ such as } G_j \text{ is an input-complete} \\ & \text{microcell or if } (G_i -) \text{ is conneted to a primary output} \\ 1 & \text{otherwise} \end{cases} \quad (4.25)$$

where $(G_i \bullet)$ are the microcells driven by G_i , and $(G_i -)$ is the output wire of microcell G_i .

The delay fitness function is the reciprocal of the sum of forward delay (D_F) and backward delay (D_B), expressed in the following (for fast calculation, the output of a microcell is assumed to drive the fan-out of 4 inverters):

$$Fitness(D) = \frac{1}{D_F + D_B} \quad (4.26)$$

On the basis of the three basic fitness functions in (4.21), (4.22), (4.24), some composite metrics can be further optimized. For example, energy can be optimized through the proposed MIGA with the fitness function in (4.25):

$$Fitness(E) = \frac{1}{(\sum_{i=1}^{M_{CD}} P_{CD,i} + \sum_{i=1}^{M_{IC}} P_{IC,i} + \sum_{i=1}^{M_{NIC}} P_{NIC,i})(D_F + D_B)} \quad (4.27)$$

Note that higher fitness values in (4.23), (4.24), (4.26) and (4.27) imply lower power dissipation, smaller transistor-count, smaller delay and lower energy respectively. This allows the proposed MIGA to employ the fitness values to guide and converge the search process to the fittest region, which is equivalent to the best combination of QDI netlist.

As an example, the circuits in Fig. 4.3(a)-(d) are used to calculate the fitness values of power dissipation. The fitness values of transistor-count and delay can be calculated accordingly but are omitted here for simplicity. The power dissipation of

the CD circuit, input-complete microcell and non-input-complete microcell (based on the average values of 2-input microcells given in Table 4.1) are 1.1, 1.5, and 1.1 respectively. From the circuit in Fig. 4.3(a), $M_{CD} = 17$, $M_{IC} = 0$, $M_{NIC} = 20$ can be obtained. Therefore, the power dissipation fitness, $Fitness(P_a) = 1/(1.1 \times 17 + 1.5 \times 0 + 1.1 \times 20) \approx 0.025$. Similarly, $Fitness(P_b) \approx 0.029 > Fitness(P_a)$; $Fitness(P_c) \approx 0.035 > Fitness(P_b)$; $Fitness(P_d) \approx 0.037 > Fitness(P_c)$. From the above four fitness values, it can be observed that higher fitness value reflects lower power dissipation.

4.5 Experimental Results

The performance profile analysis is first validated through implementation of an ISCAS-85 benchmark circuit [126], C499, embodying DCVSL and/or DSI microcells. Thereafter, the proposed design approach is compared against reported design methodologies based on five ISCAS-85 circuits (C432, C499, C880, C1355 and C1908) using different microcells. Finally, the proposed MIGA is compared against the greedy algorithm to demonstrate the efficiency of our algorithm. For completeness, the circuits are implemented in IBM 130nm CMOS and simulated in Synopsys Nanosim using spice models, and at the nominal $V_{DD} = 1.2V$. The transistor sizings of all microcells are 560nm/120nm for PMOS and 280nm/120nm for NMOS, except for the weak keepers where 160nm/500nm for PMOS, 160nm/1000nm for NMOS. The delay of ISCAS-85 benchmark circuits is computed as the combined delay of an evaluation phase ($Req_i^+ \rightarrow Req_i^-$) and a reset phase ($Req_i^- \rightarrow Req_i^+$). The computation

time of the proposed MIGA optimization for each circuit is computationally efficient (e.g. 5 minutes for C499, 13 minutes for C1908 with 300 population of chromosomes and 500 generations of genetic operations in this thesis) on the HP Pro 3130 Microtower PC computer, with an Intel i7 CPU @2.8GHz, and 4GB RAM. The proposed MIGA requires only 500 generations to obtain the optimal solution.

4.5.1 The Performance Profile for C499

Fig. 4.7(a) - (c) respectively depict the performance profiles of power dissipation, transistor-count and delay for C499, ranging from 0% DSI microcells (i.e. all DCVSL microcells) to 100% DSI microcells along the abscissa. In these profiles, the regions are quantified by the lower and upper boundaries.

From Fig. 4.7(a), C499 embodying all DCVSL microcells (at 0%) dissipates $460\mu\text{W}$ while that embodying all DSI microcells (at 100%) dissipates $284\mu\text{W}$. Nonetheless, with our proposed approach, C499 can be redesigned to achieve the lowest power dissipation of $262\mu\text{W}$ (at 57%), 43% and 8% lower than that of embodying all DCVSL and all DSI microcells respectively.

Fig. 4.7(b) depicted that C499, embodying all DCVSL microcells (at 0%), requires 6,600 transistors while C499 embodying all DSI microcells (at 100%) requires 4,914 transistors. With our proposed approach, C499 can be redesigned to achieve a lesser number of transistor-count of 4,482 (at 57%). These are 32% and 9% lesser than C499 embodying all DCVSL and all DSI microcells respectively. From Fig. 4.7(c), it can be observed that the smallest delay of C499 is 4.07ns (at 0%) and the longest delay of C499 is 9.3ns (at 100%). The upper delay boundary increases steeply along the

abscissa, and then becomes flat at the turning point of around 20%. The initial steep increase is caused by the replacements of DSI microcells located on the critical path(s). Conversely, the lower boundary remains almost flat at low percentage and increases steeply at high percentage, and the turning-up point is around 60%. In the plateau (<60%), almost all the DSI microcells are allocated at non-critical paths, hence constitute a negligible delay penalty.

As shown in Fig. 4.7(a) to Fig. 4.8(c), in this C499 example, it would be highly desirable to interleave 57% of DSI microcells (with 43% of DCVSL microcells) to enjoy low power dissipation, small transistor-count and with some delay penalty (see the delay of the lowest power point in Fig. 4.7(c)).

Fig. 4.7(d) depicts the evolving progress of the number of CD circuits in the searching for the lowest power point. With more generations evolved, the number of CD circuits is finally decreased to zero, which is near the global optimal point. The evolving progress is significant (i.e. fast optimization time) for large and complex circuits.

(a)

(b)

Fig. 4.7 Performance of benchmark circuit C499 based on our proposed microcell-interleaved approach and genetic algorithm (a) Power profile, (b) Transistor-count profile, (c) Delay profile, and (d) The number of CD circuits in the searching for the lowest power

4.5.2 Comparison on ISCAS-85 Benchmark Circuits

The ISCAS-85 benchmark circuits based on DCVSL, NCL and DSI microcells are considered. For circuits based on our proposed MIGA, two options are considered, Proposed_1 and Proposed_2. The circuits based on the Proposed_1 are interleaved with DCVSL/DSI microcells, and those based on Proposed_2 are interleaved with NCL/DSI microcells. For sake of fair benchmarking in terms of power and energy, the implementation of each circuit (such as DCVSL, NCL, etc.) is simulated at the same input rate and for the same simulation duration. Table 4.2 tabulates the performance values having the lowest power dissipation points. For ease of interpretation, these parameters are normalized to the circuits based on the Proposed_2 (with the lowest power dissipation objective), and the actual numerical values are shown in parentheses.

In terms of power dissipation, the circuits of DCVSL, NCL, DSI are respectively, on average, $2.17\times$, $1.94\times$ and $1.20\times$ worse than those of the Proposed_2. The circuits of the Proposed_1 are better than those of DCVSL, NCL and DSI, but are $1.11\times$ worse than those of the Proposed_2. In short, the type of microcells plays an important role in the overall circuit optimization.

From the transistor-count perspective, the circuits of the Proposed_2 feature the lowest number. The circuits of DCVSL, NCL, DSI and the Proposed_1 are $1.55\times$, $1.53\times$, $1.12\times$ and $1.01\times$ worse than those of the Proposed_2 respectively.

For delay parameter, the circuits of DCVSL, NCL, and the Proposed_1 feature shorter delay than those of the Proposed_2, where their average delays are $0.41\times$,

0.62 $\times$ and 0.87 $\times$ better. As expected, the DSI circuits feature the laiest delay.

Table 4.2 Power dissipation (P), transistor-count (TC), delay (t) and composite metric of $P \times TC \times t$ of the lowest power points for various benchmarks@ 1.2V, IBM 130nm CMOS

Circuits	Power, P (μ W) @ 50MHz				
	DCVSL	NCL	DSI	Proposed_1	Proposed_2
ISCAS C432	2.12 $\times$	1.91 $\times$	1.11 $\times$	1.05 $\times$	1 $\times$ (209)
ISCAS C499	1.82 $\times$	1.77 $\times$	1.13 $\times$	1.04 $\times$	1 $\times$ (253)
ISCAS C880	2.22 $\times$	1.97 $\times$	1.20 $\times$	1.12 $\times$	1 $\times$ (359)
ISCAS C1355	2.33 $\times$	1.99 $\times$	1.28 $\times$	1.17 $\times$	1 $\times$ (473)
ISCAS C1908	2.39 $\times$	2.07 $\times$	1.28 $\times$	1.18 $\times$	1 $\times$ (453)
Average	2.17 $\times$	1.94 $\times$	1.20 $\times$	1.11 $\times$	1 $\times$ (349)

Circuits	Transistor-Count, TC				
	DCVSL	NCL	DSI	Proposed_1	Proposed_2
ISCAS C432	1.51 $\times$	1.49 $\times$	1.06 $\times$	0.99 $\times$	1 $\times$ (4114)
ISCAS C499	1.49 $\times$	1.59 $\times$	1.11 $\times$	1.01 $\times$	1 $\times$ (4440)
ISCAS C880	1.56 $\times$	1.52 $\times$	1.13 $\times$	1.02 $\times$	1 $\times$ (7072)
ISCAS C1355	1.58 $\times$	1.53 $\times$	1.15 $\times$	1.02 $\times$	1 $\times$ (9628)
ISCAS C1908	1.60 $\times$	1.55 $\times$	1.17 $\times$	1.02 $\times$	1 $\times$ (9330)
Average	1.55 $\times$	1.53 $\times$	1.12 $\times$	1.01 $\times$	1 $\times$ (6917)

Circuits	Delay, t (ns)				
	DCVSL	NCL	DSI	Proposed_1	Proposed_2
ISCAS C432	0.32 $\times$	0.46 $\times$	1.03 $\times$	0.74 $\times$	1 $\times$ (12.66)
ISCAS C499	0.48 $\times$	0.77 $\times$	1.11 $\times$	0.91 $\times$	1 $\times$ (8.41)
ISCAS C880	0.41 $\times$	0.66 $\times$	1.17 $\times$	0.89 $\times$	1 $\times$ (10.04)
ISCAS C1355	0.41 $\times$	0.57 $\times$	1.15 $\times$	0.92 $\times$	1 $\times$ (11.74)
ISCAS C1908	0.43 $\times$	0.64 $\times$	1.19 $\times$	0.88 $\times$	1 $\times$ (11.86)
Average	0.41 $\times$	0.62 $\times$	1.13 $\times$	0.87 $\times$	1 $\times$ (10.94)

Circuits	Normalized Overall ($P \times TC \times t$)				
	DCVSL	NCL	DSI	Proposed_1	Proposed_2
ISCAS C432	1.03 $\times$	1.31 $\times$	1.20 $\times$	0.76 $\times$	1 $\times$
ISCAS C499	1.31 $\times$	2.16 $\times$	1.38 $\times$	0.95 $\times$	1 $\times$
ISCAS C880	1.42 $\times$	1.98 $\times$	1.57 $\times$	1.01 $\times$	1 $\times$
ISCAS C1355	1.51 $\times$	1.73 $\times$	1.70 $\times$	1.10 $\times$	1 $\times$
ISCAS C1908	1.65 $\times$	2.05 $\times$	1.78 $\times$	1.06 $\times$	1 $\times$
Average	1.39 $\times$	1.85 $\times$	1.52 $\times$	0.98 $\times$	1 $\times$

In the composite power $\times$ transistor-count $\times$ delay, on average, the circuits of the Proposed_1 feature the best (0.98 $\times$ marginally better than those of the Proposed_2), and the circuits of DCVSL, NCL, DSI have 1.39 $\times$, 1.85 $\times$ and 1.52 $\times$ worse overall performance than those of the Proposed_2.

In Table 4.2, the circuits of the Proposed_1 and the Proposed_2 are optimized based on the lowest power dissipation. Although not shown, our proposed design approach could be applied to optimize other parameters (including delay and transistor-count). The multiple-objective optimization is also possible by integrating different fitness functions into a composite fitness function, or implementing a decision maker to subjectively conduct the evolution, but it is beyond the scope of this work.

4.5.3 Comparisons of the Proposed Microcell-Interleaving Genetic Algorithm with Other Algorithms

In order to demonstrate the efficiency of the proposed MIGA, five ISCAS benchmark circuits based on the greedy algorithm and dynamic programming have been implemented, and benchmarked against our proposed MIGA. For the implementation of greedy algorithm, the microcells that have the largest power dissipation reduction in each evaluation will be replaced. For the implementation of dynamic programming, each microcell is considered as a stage, it optimizes the global minimum power dissipation solution on the basis of finding a lowest power dissipation solution at each stage (sub-solution). The pseudo code of the implemented greedy algorithm and dynamic programming is depicted in Fig. 4.8. The results are

Greedy Algorithm

Input: all microcell initial types (input-complete or non-input- complete).

Define: @Determine: the microcells whose types are determined
 @Non-determine: the microcells whose types are not determined

Main loop:
 update @Determine, @Non-determine and CD circuits required;
 for $i=1,2,\dots, \text{Number of Microcells}$
 Calculate the power reduction of each microcell in @Non-determine;
 Rank the above power reduction in descending order;
 Replace the number 1 microcell to input-complete type;
 update @Determine, @Non-determine and CD circuits required;
 End for

Output: @Determine records the optimal solutions.

Dynamic Programming

Input: all microcell types (input-complete or non-input- complete) in connection.

Define: stage i : the types of i microcells are determined
 $F_i(p)$: the power reduction of stage i ;

Main loop:
 for $i=1,2,\dots, \text{Number of Microcells}$
 Calculate the power reduction (ΔP) of i th microcell if change the type;
 if ($\Delta P > 0$) $F_i(p) = F_{i-1}(p) + \Delta P$;
 else $F_i(p) = F_{i-1}(p)$;
 update the microcells in stage i ;
 End for

Output: the last stage is the optimal solution.

Fig. 4.8 Pseudo-code of greedy algorithm and dynamic programming

tabulated in Table 4.3. The proposed MIGA out performs the greedy algorithm and dynamic programming by ~10% and ~11% respectively. The reason is because the proposed MIGA makes use of multiple search paths and global information to explore the multimodal search space. It is interesting to note that dynamic programming might not perform well in some cases as the sub-solutions generated in the optimization process may not effectively lead to the optimal solution, probably due to the lack of optimal substructures [127] in this problem.

Table 4.3 The lowest power points (μW @50MHz) comparisons between the Greedy Algorithm, Dynamic Programming and the proposed MIGA for five ISCAS85 benchmark circuits @ 1.2V, IBM 130nm CMOS; normalized to the proposed MIGA

	C432	C499	C880	C1355	C1908	Average
DSI	1.11×	1.13×	1.20×	1.28×	1.28×	1.20×
Greedy Algorithm	1.10×	1.12×	1.06×	1.12×	1.10×	1.10×
Dynamic Programming	1.09×	1.10×	1.13×	1.12×	1.11×	1.11×
Proposed MIGA	1× (209)	1× (337)	1× (478)	1× (473)	1× (453)	1×

5.

Proposed Dynamic Voltage Scaling Technique on Asynchronous Quasi-Delay-Insensitive Digital Systems

5.1 Introduction

Generally, in most, if not all, of the portable electronic applications such as mobile phones, personal multimedia devices and notebook PCs are powered by batteries with limited energy storage. It is therefore desirable if the power dissipation of these devices can be maintained at the minimum level to perform necessary operations in order to maximize the length of their battery life. In view of the conflicting objectives of high operating speed and low power dissipation, numerous studies have been carried out to achieve an optimal tradeoff between these two objectives. Among these studies, DVS is one of the most widely applied techniques [128-135]. With DVS, the energy dissipation can be reduced substantially by scaling down the supply voltage when the operating workload is low. In applying DVS techniques, one of the critical issues is to effectively detect the completion of operations as the operating speed may need to be varied due to changes in input data

and PVT variations. In this context, the self-detecting attribute in the async QDI circuits presented in Chapter 3 and Chapter 4 will be expedient to the proposed async QDI DVS technique.

In this chapter, a FIR filter is employed as a test circuit to demonstrate the features of the proposed pipeline-granular async QDI DVS technique. The proposed DVS technique can significantly reduce the power dissipation by adaptively scaling down the supply voltage when the filter is not operating at full speed and has the following novelties/features. First, no additional timing margin is required to ensure the delay of the circuit operation due to input data and PVT variations. This is realized by the async QDI dual-rail circuits which allow the completion of operations to be automatically detected and the PVT variations to be adaptively accommodated. Second, an on-chip Power Supply Unit (PSU), with three supply voltage scales (0.5V, 0.8V and 1.2V), is integrated into each pipeline. Hence, the supply voltage of each pipeline stage can be scaled according to their corresponding workloads in preceding, current and successive pipeline stages to maintain minimum power dissipation. Third, the proposed async QDI DVS is free of error-data-vulnerability as robustness of the circuit operations is safeguarded by the QDI attribute. In the FIR filter, the operations of the critical multiplier pipeline stage are analyzed. The results validate that the proposed async QDI DVS technique is able to adaptively adjust the supply voltages based on the operating load. The three proposed scaling modes for the multiplier pipeline stage dissipate $3.53\times$, $2\times$ and $1\times$ of $89.9\mu\text{W}$ minimum power dissipation with 0.5V of supply voltage.

The pipeline structure, circuit modules and operation modes of the proposed async QDI DVS technique are now presented. The design details of the FIR filter will then be described.

5.2 Proposed Dynamic Voltage Scaling Technique

The proposed async QDI DVS technique is applied to each pipeline stage, thereby enabling voltage control in the pipeline to maintain minimum power dissipation effectively. Fig. 5.1 depicts the block diagram of the proposed async QDI DVS technique. In this figure, each pipeline stage (e.g. *Pipeline_i*) consists of *PSU_i*, *Register_i*, *RCD_i*, *Dual-rail Combinational Logic_i*, *DCD_i*, *Level Shifter (LS)_i* and handshake controller *C_i*. *PSU_i* is the unit controls the supply voltage of each pipeline stage according to their operating speed. Three voltage levels (0.5V, 0.8V and 1.2V) are generated by *PSU_i*. *PSU_i* is designed to be on-chip and incorporated into each pipeline stage, making power dissipation reduction more efficiently in granular manner. *Register_i* is employed to latch and hold the dual-rail input data for computation in *Pipeline_i*. It is integrated at the input of *i* pipeline stage from the preceding pipeline stage. *RCD_i* is the Register Completion Detection circuit that detects the nullity/validity of the input and it ensures the arrival of all input signals for the pipeline stage. *Dual-rail Combinational Logic_i* is the async QDI dual-rail computational block such as multiplier and adder blocks in the proposed FIR filter. *DCD_i* is the Data Completion Detection circuit that checks the validity/nullity of all the internal wires within *Dual-rail Combinational Logic_i*. *LS_i* converts signal of

different virtual V_{DD} (VV_{DD}) (0.5V or 0.8V) to nominal voltage supply (1.2V) to ensure proper data integrity between the operations in different pipeline stages. The handshake controller C_i (with the functionality of inverted C-Muller gate) is to indicate the data in the current pipeline stage are correctly transferred to the following pipeline stage. It also generates the request signal to initiate a new operation by latching the data to $Register_i$.

Now consider the power supply domains of the proposed async QDI DVS technique. Each pipeline stage (e.g. $Pipeline_i$ stage) has two voltage supply domains. One is the nominal voltage supply domain, consisting of PSU_i and LS_i . The other is the VV_{DD} supply domain (depicted by the left shaded box for $Pipeline_i$ in Fig. 5.1)

Fig. 5.1 Block diagram of the proposed async QDI DVS pipeline

and the VV_{DD} is generated by PSU_i . The VV_{DD} supply domain comprises of $Register_i$, RCD_i , *Dual-rail Combinational Logic_i*, DCD_i , LS_i and handshake controller C_i . The VV_{DD} supply domain is the main DVS domain to adaptively scale the supply voltage according to the operating speed of *Dual-rail Combinational Logic_i*. The conversion of signal voltage between two power supply domains is done by LS_i . Three signals, RCD_i , $Done_i$ and $Input_{i+1}$, require voltage conversion between two pipeline stages. The signal voltages of RCD_i and $Done_i$ are converted to nominal voltage before transmitting to preceding and successive pipeline stages. The LS_i will also convert the voltage of the *Pipeline_i* output data to the appropriate voltage of the successive pipeline stage.

The operation of the proposed async QDI DVS technique is now discussed. Basically, there are two main types of operations: pipeline handshake operations and DVS operations. For pipeline handshake operations, the handshake signal Req_i is initially asserted high by handshake controller C_i to indicate the completion of operation in *Pipeline_i* and is ready to accept a new input data for the next operation. The RCD_i is used to detect the arrival of input signal. After the input data are processed by *Dual-rail Combinational Logic_i*, the output of DCD_i will be asserted high to indicate the completion of operation. The output data will then be converted to nominal voltage by LS_i and passed to the next pipeline stage for processing. After the RCD_{i+1} detects the completion of latching $Input_{i+1}$, it will trigger the handshake controller C_i in *Pipeline_i* to commence the next operation.

For voltage scaling operations, PSU_i is employed to adaptively scale the voltage

supplies to the VV_{DD} domain. At the outset, PSU_i is triggered by the input completion and operation completion signals of the preceding, current and successive pipeline stages, namely $Input_Done_{i-1}$, $Done_{i-1}$, $Input_Done_i$, $Done_i$, $Input_Done_{i+1}$ and $Done_{i+1}$ respectively as depicted in Fig. 5.1. LS_i will then convert the voltage of these input signals to nominal voltage as these signals are generated by different voltage supply domains. Thereafter, the PSU_i will adaptively scale the VV_{DD} according to the timing information of these input signals. The details of the PSU_i operations will be presented later.

In summary, the async QDI dual-rail handshake operations innately detect the completion time of each pipeline, ensures the successful propagations and provide operating timing information of each pipeline to PSU to adaptively scale the operating voltage of the dual-rail combinational logic.

5.3 Design Details of the Proposed Dynamic Voltage Scaling Pipeline

In this section, the details of the circuit modules involved in the voltage scaling operation, including the PSU_i and LS_i , of the proposed async QDI DVS Technique will be presented.

5.3.1 On-chip Pipeline-granular Power Supply Unit with Three Voltage Scales

The PSU is integrated on-chip in each pipeline stage (pipeline-granularity) and designed to adaptively scale the VV_{DD} according to the operating speed of preceding,

current and successive pipeline stages. The supply is scaled to the minimum voltage to achieve low power dissipation if current pipeline is faster than the preceding and successive pipelines. The minimum voltage is set to near threshold ($\sim 0.5\text{V}$) to avoid the circuit going to sub-threshold region. The current pipeline requires the maximum supply voltage (1.2V) to speed up operation if current pipeline is slower than the preceding and successive pipelines. The current pipeline will be the potential speed bottleneck of the three pipelines if it is faster than one of the preceding and successive pipelines but slower than the other; hence the supply voltage is scaled to the medium level ($\sim 0.8\text{V}$) to tradeoff between the speed and power dissipation. Specifically, the PSU innately scales the V_{DD} in the following three cases:

- 1) PSU scales the V_{DD} to low voltage level (0.5V) from high or medium voltage (1.2V or 0.8V) if the operation of current pipeline stage completes before both of the successive and preceding pipeline stages.
- 2) PSU scales the V_{DD} to medium voltage level (0.8V) from low voltage level (0.5V) if the operation of current pipeline stage completes after one of the preceding and successive pipeline stages, but before the other.
- 3) PSU scales the V_{DD} to high voltage level (1.2V) from medium voltage level (0.8V) if the operation of current pipeline stage completes after both of the preceding and successive pipeline stages.

The control logic and power transistors of PSU are depicted in Fig. 5.2. It comprises of two parts. Part I is the control logic to generate control signals to enable the voltage scaling. Part II is the power transistor circuit. The two parts are described

next.

The control logic in Part I takes the status signals from the previous, current and successive pipeline stages as inputs, namely $Input_Done_{i-1}$, $Done_{i-1}$, $Input_Done_i$, $Done_i$, $Input_Done_{i+1}$ and $Done_{i+1}$ respectively. It generates control signals A and B to Part II.

The voltage scaling truth table of the current pipeline stage with respect to the control signals of current and successive, current and preceding pipelines are depicted in Table 5.1 and Table 5.2 respectively. From the tables, it is observed that there are two cases represented by 'YES' in the 'Scaling' column in each table. The two 'YES' cases in each table represent the reset phase (Null) and evaluation phase (Valid) of the current pipeline stage respectively.

Table 5.1 Voltage scaling truth table of current pipeline stage with respect to control signals of current and successive pipelines ('0' indicates Null data, '1' indicates Valid data, 'X' indicate 'don't care states')

$Input_Done_i$	$Input_Done_{i+1}$	$Done_i$	$Done_{i+1}$	Scaling
0	0	X	X	NO
1	1	X	X	NO
0	1	0	X	NO
0	1	1	0	NO
0	1	1	1	YES
1	0	1	X	NO
1	0	0	1	NO
1	0	0	0	YES

Table 5.2 Voltage scaling truth table of current pipeline stage with respect to control signals of current and preceding pipelines ('0' indicates Null data, '1' indicates Valid data, 'X' indicate 'don't care states')

$Input_Done_i$	$Input_Done_{i-1}$	$Done_i$	$Done_{i-1}$	Scaling
0	0	X	X	NO
1	1	X	X	NO
0	1	0	X	NO
0	1	1	0	NO
0	1	1	1	YES
1	0	1	X	NO
1	0	0	1	NO
1	0	0	0	YES

It can be observed from Table 5.1 and Table 5.2, the control signals A and B are generated by the circuits depicted in Part I of Fig. 5.2. $Node_1$ resets to 0 to indicate the current pipeline is slower than the preceding pipeline. Similarly, $Node_2$ resets to 0 to indicate the current pipeline is slower than the successive pipeline. Signal A resets to logic 0 to conduct First Charge Path (will be described later) if the current pipeline is slower than either the preceding or successive pipelines. Similarly, Signal B will reset to logic 0 to conduct Second Charge Path (will be described later) if the

current pipeline is slower than both of the preceding and successive pipelines.

Part II in Fig. 5.2 depicts the schematic of the power transistors in PSU to realize the three voltage scales, i.e. 0.5V for low speed operation, 0.8V for medium speed operation and 1.2V for high speed operation respectively. It consists of three power supply paths: Clamp Path, First Charge Path and Second Charge Path. The three power supply paths correspond to low, medium and high voltages respectively. In the Clamp Path, NMOS transistors are connected in series; Drain and Gate of each transistor are connected together, forcing the transistors to operate in saturation region. M NMOS transistors are assumed to be stacked in the Clamp Path and the voltage of V_{DD} (low voltage) is modeled as:

$$V_{DD} = V_{DD} - M \cdot V_{TH} \quad (5.1)$$

In order to significantly reduce the power dissipation, V_{DD} is scaled to near threshold voltage. The corresponding number of NMOS transistors is shown in Eq. (5.1). For the proposed FIR filter, $M = 3$ is adopted such that the supply voltage is clamped at (or scaled to) 0.5V which corresponds to low power operation mode.

Fig. 5.2 Control logic and power transistors of PSU in the proposed Async QDI DVS technique

First Charge Path increases the VV_{DD} for the case where the supply voltage

provided by the Clamp Path is insufficient to complete the operation. First Charge Path comprises of N NMOS transistors connected in series and one PMOS transistor connected to V_{DD} . The N NMOS transistors are operating in saturation region and the PMOS transistor is controlled by signal A . The V_{DD} (medium voltage) of First Charge Path is modeled as follow:

$$V_{DD} = V_{DD} - N \cdot V_{TH} \quad (5.2)$$

where $N < M$ and $N = \lfloor M / 2 \rfloor$.

In order to accommodate a higher speed operation, Second Charge Path can be activated where V_{DD} is scaled to V_{DD} through a POMS transistor controlled by signal B , since V_{DD} can pass through POMS transistor without degrading signal voltage, i.e. V_{TH} loss.

In summary, the PSU detects the operation speed of the preceding, current and successive pipeline stages and adaptively adjust the V_{DD} domain of each pipeline. It generates three supply voltages ($V_{DD} - M \cdot V_{TH}$, $V_{DD} - N \cdot V_{TH}$ and V_{DD} respectively) to tradeoff between low power dissipation and high operating speed operations. The PSU is integrated on-chip in each pipeline stage.

5.3.2 Level Shifter

The LS is employed to convert the signals from one voltage level to another. Specifically in the proposed async QDI DVS technique, the output signal from V_{DD} domain is converted to nominal voltage to ensure correct operations and prevent data error caused by different power supply domains.

The transistor level implementation of the LS is depicted in Fig. 5.3. The

complementary signal of *Input* is first generated. The input signal *Input* is in V_{VDD} domain and the two inverters employed to generate the complementary signals also operate in V_{VDD} domain. The two NMOS transistors are driven by the two complementary signals. The Drains of these two NMOS transistors are connected to the cross-coupled PMOS transistors driven by V_{DD} . The output signal Q is generated by an output inverter operating at V_{DD} . This output inverter separates the voltage converting stage and the output stage.

5.4 Operation Mode Investigations

In this section, the different operation modes of the proposed async QDI DVS technique are presented. There are four operating modes: low power operation mode, single charge up of current-preceding mode, single charge up of current-successive mode and double charge up mode.

Fig. 5.3 Transistor level implementation of level shifter

The completions of input data and dual-rail combinational logic operation at each pipeline stage are indicated by the *input_done* and *done* signals respectively. The proposed async QDI DVS operation of each pipeline is corresponding to three adjacent pipeline stages: $i-1$, i , $i+1$ pipeline stages and they are considered as the preceding, current and successive pipeline stages respectively. The operations of PSU in current pipeline stage are presented as follows. The PSU in current pipeline receives the signals from the preceding, current and successive pipeline stages, namely $input_done_{i-1}$, $done_{i-1}$, $input_done_i$, $done_i$, $input_done_{i+1}$ and $done_{i+1}$ respectively as input and supplies the scaled voltage VV_{DD} as output as depicted in Fig. 5.1.

5.4.1 Low Power Operation Mode

The timing diagram of the low power operation mode is depicted in Fig. 5.4. Initially, the voltage charge signals A and B are set to logic 1 and VV_{DD} is set to V_{DD} . The pipeline stage $Pipeline_{i+1}$ commences the operation by latching valid data $DATA_i$ and setting the $Input_Done_{i+1}$ to logic 1. Subsequently, $Pipeline_i$ accepts the null data in preceding pipeline stages and reset $Input_Done_i$ to logic 0. With the operation in $Pipeline_i$, the VV_{DD} decreases gradually to the minimum voltage (0.5V). Thereafter, $Pipeline_{i-1}$ will latch the input data from its preceding pipeline stage and commence a new operation. Resetting $Done_i$ to logic 0 indicates that the pipeline stage $Pipeline_i$ had completed its operation and the corresponding computed null data ($DATA_i$) is ready at the output of pipeline stage. Thereafter, the current pipeline $Pipeline_i$ stops its operation while waiting for the successive pipeline stage to receive the new output

data. The supply voltage of $Pipeline_i$ V_{DD} will not be charged and remain at low voltage level of minimum V_{DD} of 0.5V.

5.4.2 Single Charge Up of Current-preceding Mode

Fig. 5.5 shows the timing diagram of single charge up of current-preceding mode. Similar to the low power operation mode, the pipeline stage $Pipeline_{i+1}$ commences

Fig. 5.4 The timing diagrams of low power operation mode

the operation by latching the valid data in $DATA_i$ and $Input_Done_{i+1}$ is then set to logic 1. Subsequently, $Pipeline_i$ receives the null data in preceding pipeline stages $Pipeline_{i-1}$ and resets $Input_Done_i$ to logic 0. The V_{DD} gradually decreases because of the operation in $Pipeline_i$. Thereafter, $Pipeline_{i-1}$ will latch the input data from its preceding pipeline stage and commence the data processing. Among the three operations (in three pipeline stage respectively), $Pipeline_{i-1}$ features the fastest operating speed and finishes the operation before $Pipeline_i$ and $Pipeline_{i+1}$. The

Op_PS: Operation in Preceding Stage
Op_CS: Operation in Current Stage
Op_SS: Operation in Successive Stage

Fig. 5.5 The timing diagrams of single charge up of current-preceding mode

operation completion information of $Pipeline_{i-1}$ is passed to the PSU in $Pipeline_i$ through $Input_Done_{i-1}$ and $Done_{i-1}$, which trigger the PSU in $Pipeline_i$ to reset control signal A to logic 0 and conduct First Charge Path. The V_{DD} is charged up to medium voltage scale (0.8V) to speed up the operation completion in current pipeline stage. The first charge up is stopped when the operation in current pipeline stage $Pipeline_i$ is completed and the processed data ($DATA_i$) is ready for latching by the successive pipeline stage. The V_{DD} remains at medium level (0.8V) as no operation is performing and decreases to low voltage scale (0.5V) when the next operation commences.

In single charge up of the current-preceding mode, the evaluating/resetting speed of the current pipeline is slower than the preceding pipeline. One charge up is performed to increase the speed of the current pipeline stage. The operations are operating at the low voltage supply except for the charge up period. The V_{DD} is supplied by the Clamp Path and First Charge Path in the PSU. As compared to the nominal full V_{DD} power supply, single charge up of current-preceding mode will reduce the power dissipation significantly as the circuits mainly operate at the low supply voltage.

5.4.3 Single Charge Up of Current-successive Mode

In single charge up of current-successive mode, the operation in the successive pipelines stage is slower than current pipeline stage $Pipeline_i$, a charge up to medium voltage scale (0.8V) is required to speed up the operation in current pipeline stage. The Clamp Path and First Charge Path in the PSU are employed to supply the current

required by the operation. The timing diagram of single charge up of the current-successive mode is presented in Fig. 5.6.

The starting operations of the three pipeline stages are in the sequence of $Pipeline_{i+1}$, $Pipeline_i$ and $Pipeline_{i-1}$, which are similar to the two modes described earlier. Among the three pipeline stages, $Pipeline_{i+1}$ features the fastest speed and the information of operation completion passes to the current pipeline stage through $Input_Done_{i+1}$ and $Done_{i+1}$. Similarly, PSU conducts First Charge Path by resetting control signal A to logic 0. V_{DD} is then charged to the medium voltage scale (0.8V) and remains unchanged until the commencement of next operation.

Fig. 5.6 The timing diagrams of single charge up of current-successive mode

With the operation analysis of single charge up mode of current-successive mode, it can be observed that the dual-rail combinational logic mainly operates at the low power supply (0.5V). Thus, the proposed async QDI DVS technique has a significant reduction on power dissipation.

5.4.4 Double Charge Up Mode

The pipeline and supply voltage operation of double charge mode will now be presented. This mode is designed for higher speed operation (when compared to the

Fig. 5.7 The timing diagrams of double charge up mode

three operation modes described earlier) with relatively low power dissipation. The three power supply paths (Clamp Path, First Charge Path and Second Charge Path in PSU) are fully used to supply the current drawn by V_{DD} domain.

The timing diagram of double charge mode is illustrated in Fig. 5.7. Similarly, the operation starts in the sequence of $Pipeline_{i+1}$, $Pipeline_i$ and $Pipeline_{i-1}$ and V_{DD} is discharged to low supply voltage at the early operation stage. In this double charge mode, the current pipeline stage $Pipeline_i$ features the lowest speed. Hence, the supply voltage V_{DD} of $Pipeline_i$ is firstly charged to medium voltage level (0.8V) when either the preceding or successive pipeline stage completes its operation (the preceding pipeline stage features the highest speed in Fig. 5.7). Thereafter, V_{DD} is charged to V_{DD} when the other pipeline stage completes the operation. V_{DD} begins to decrease when the next operation of current pipeline stage starts.

In this double charge mode, the supply voltage is higher than the other three modes for highest speed operation.

5.5 Power Supply Unit Size Determination

In this section, the sizing of transistor in PSU is discussed. From the implementation of PSU, it can be observed that the Clamp Path provides the current drawn by V_{DD} domain for the four operation modes discussed in Section 5.4. In order to prevent the V_{DD} lower than required value, the transistors in the Clamp Path are sized to be much larger in order to maintain a strong V_{DD} .

The power dissipation can be modeled as:

$$P = P_{Leakage} + P_{Internal} + P_{Dynamic} \quad (5.3)$$

Then, the current provide by Clamp Path is estimated as:

$$I_{Clamp} = \frac{P_{Leakage} + P_{Internal} + P_{Dynamic}}{VV_{DD}} \quad (5.4)$$

The maximum drop percentage of VV_{DD} during operation is assumed to be α ;

hence the maximum voltage drop on each transistor on Clamp Path is expressed:

$$V_{GS} = \frac{V_{DD} - (1 - \alpha)V_{DD}}{M} \quad (5.5)$$

where M is the number of transistors in the Clamp Path depicted in Fig. 5.2. As all the NMOS transistors in the Clamp Path are operating in saturation region. Thus, the current of each transistor can be expressed as:

$$I_D = \left(\frac{\mu_n C_{ox} (W / L)}{2} \right) (V_{GS} - V_{TH})^2 \quad (5.6)$$

Hence, $I_{Clamp} \leq I_D$ should be satisfied to provide sufficient current and with

Eq (5.1), (5.3), (5.5), the following is obtained:

$$\frac{W}{L} \geq \frac{2(P_{Leakage} + P_{Internal} + P_{Dynamic})M^2}{\mu_n C_{ox} \alpha^2 (V_{DD} - MV_{TH})^3} \quad (5.7)$$

5.6 Proposed FIR Filter with Async QDI DVS Technique

An async QDI FIR filter employing the async QDI DVS technique is proposed and the block diagram of the proposed FIR filter is depicted in Fig. 5.8. The proposed FIR filter operates as follows. The *State Machine* generates the write address and write enable signal on $SRAM_Addr$ and R/W respectively to receive the input $Data_In$.

Fig. 5.8 Block diagram of 8-bit 8-tap async QDI FIR filter with proposed DVs technique

Thereafter, *SRAM_Addr*, *ROM_Addr*, *R/W* signals are generated to read out data and

Fig. 5.9 Waveforms for the low power operation mode in Async QDI FIR filter multiplier pipeline stage

coefficient that are latched by Latch 3 and Latch 2 respectively. The read out data and coefficient then are fed to the 8-bit Multiplier for multiplication. The output of the Multiplier is held in Latch 5 and is accumulated by the Adder.

There are two DVS power domains: multiplier and adder pipeline stages, depicted in the two shade boxes in Fig. 5.8. The multiplier power domain performs DVS by receiving the input completion detection and data completion detection signals from the state machine and adder pipeline stages. Similarly, the adder pipeline stage accepts the status signals from multiplier and primary output stage for control.

The multiplier pipeline stage follows the four operation modes presented earlier to scale the supply voltage accordingly. The evaluations of all these modes are simulated in SPICE netlist model @ nominal voltage 1.2V. The simulated waveforms of the PSU in current pipeline stage, i.e. the multiplier pipeline stage, are presented. Basically, the input signals to the PSU, i.e. the status signals of preceding, current and

successive pipelines are *Input_Done_proceeding*, *Done_preceding*, *Input_Done_current*, *Done_current*, *Input_Done_successive* and *Done_successive* respectively. The control signals *A* and *B* are used to turn on First Charge Path and Second Charge Path of the PSU.

The low power operation mode is firstly evaluated and the waveform is captured in Fig. 5.9. The three pipeline stages of preceding, current and successive stages are operating simultaneously. The voltage waveform represents the voltage supply of the current stage. Initially, the supply voltage is charged to 1.2V and decreased to 0.5V when the multiplication operation in the current pipeline stage commences. In this operation, the current pipeline stage completes its operation before the preceding and successive pipeline stages. Thus, all the operations of current pipeline stage are

Fig. 5.10 Waveforms for the single charge current-preceding mode in Async QDI FIR filter multiplier pipeline stage

Fig. 5.11 Waveforms for the single charge current-successive mode in Async QDI FIR filter multiplier pipeline stage

performed at the low voltage level thereby maintaining lowest power dissipation of $89.9\mu\text{W}$.

The waveform of the single charge up of current-preceding mode is depicted in Fig. 5.10. Similar to the low power operation mode, the voltage supply of the current pipeline stage decreases to 0.5V from the fully charged V_{DD} range of 1.2V. However, the preceding pipeline stage completes operation before the current pipeline stage and V_{DD} is charged up to 0.7V (the charge up does not reach the prescribed 0.8V as it is stopped when the current pipeline completes its multiplication) by resetting the control signal A to logic 0. Similarly, the waveform of the single charge up of current-successive mode is depicted in Fig. 5.11. In this mode, the successive pipeline completes its operation before the current pipeline, resulting in the V_{DD} charges up to medium voltage.

The double charge up mode is now presented. This mode accommodates the higher speed operation while minimizing the power dissipation. The operation waveform is depicted in Fig. 5.12. Specifically, the operation in this current pipeline stage completes after both the preceding and successive pipeline stages. The first charge up is triggered at the completion of successive pipeline stage operation. Thereafter, the preceding pipeline stage also completes before the current pipeline and the second charge is triggered by resetting the control signal B to logic 0. The double charges are stopped when the current pipeline completes its operation, resulting in V_{DD} being charged to 0.9V (instead of the prescribed 1.2V).

Fig. 5.12 Waveforms for the double charge up mode in Async QDI FIR filter multiplier pipeline stage

The power dissipation of the proposed async QDI DVS technique is also presented and the results are tabulated in Table 5.3. The power dissipations of the four proposed operation modes are simulated and the conventional async QDI circuit design methodology [21] is benchmarked accordingly. The simulation duration and number of operations are set to be the same for all the operation modes. All the modes are normalized to the proposed low power mode as it features the lowest power dissipation. Based on the simulation results, the single charge up modes and the double charge up mode feature $1.73 \times$, $2 \times$ and $3.53 \times$ higher power dissipation. The conventional async QDI circuit design methodology [21] features the highest power dissipation at $6.82 \times$ of the low power mode of the proposed async QDI DVS. From the results, it is observed that the proposed async QDI DVS technique is effective in reducing the power dissipation.

Table 5.3 Power dissipation comparisons ($\mu\text{W}@ 1.2\text{V}$) between the four operation modes with the proposed async QDI DVS technique and the nominal voltage (1.2V) operation

Mode	CAQDI	Proposed I	Proposed II	Proposed III	Proposed IV
Power	$6.82 \times$	$3.53 \times$	$2 \times$	$1.73 \times$	$1 \times$ (89.9)

CAQDI: Conventional Async QDI circuits

Proposed I: the proposed double charge mode

Proposed II: the proposed current-preceding single charge mode

Proposed III: the proposed current-successive single charge mode

Proposed IV: the proposed low power mode

6.

Conclusions and Recommendations for Future Work

6.1 Conclusions

In this thesis, the design and realization of low overhead robust async QDI digital systems with low design effort has been proposed. The proposed design methodologies have been verified through several design examples and the effectiveness and competitiveness in low overhead and robustness have also been demonstrated.

First, a synthesis tool of robust async QDI systems from sync specifications has been proposed. The proposed synthesis tool has the following novel features. The targeted synthesized circuits feature the QDI attribute, allowing them to be inherently timing-robust to accommodate the PVT variations. More specifically, four-phase handshake protocol, dual-rail data coding and push communication channel have been adopted for the targeted robust QDI systems. The Verilog HDL language can be used in the proposed synthesis tool to specify (input) async QDI circuits which is similar to the standard coding for sync circuits. No special design or coding rules were required.

Hence, the proposed synthesis tool could be seamlessly integrated with standard sync design flow and fully leverage on the commercial EDA tools. The proposed synthesis tool is applicable to accept various QDI library cells, enabling the full merit of different library cells to be explored. To validate the proposed synthesis, a FIR filter has been synthesized and benchmarked. It dissipated 3.7mW @ 5MHz sampling rate and featured $1.53 \times$ higher in power efficiency when benchmarked with the competitive design methodology.

Second, a design approach to mitigate the hardware overhead of the Data Completion Detection circuit in async QDI circuits has been proposed. In this proposed design approach, a novel microcell-interleaving approach has been proposed to reduce the number of completion detection circuits whilst retaining the QDI attribute. The performance of the QDI circuits based on the proposed microcell-interleaving approach has been analyzed in terms of power dissipation, transistor-count and delay and evaluated/determined the upper and lower boundaries of these performance profiles. A Microcell-Interleaving Genetic Algorithm has been proposed to stochastically optimize the proposed microcell-interleaving approach on power dissipation, transistor-count and delay. To validate the proposed design approach, a complete performance profile of ISCAS-85 C499 circuit has been implemented and simulated on the basis of DCVSL and DSI microcells. The efficiency of the proposed design approach has been demonstrated by benchmarking against the competing DCVSL, NCL and DSI designs on five ISCAS-85 circuits - the proposed designs, on average, were $1.77 \times$ better in power dissipation, $1.4 \times$ better

in area and $1.58\times$ better in a composite metric of power $\times$ area $\times$ delay and reasonably slower for the lowest power dissipation points. Finally, the proposed MIGA has outperformed Greedy Algorithm and Dynamic Programming by 10.1-11.5% improved efficiency.

Third, a pipeline-granular async QDI dynamic voltage scaling technique has been proposed to operate a pipeline stage either at low power dissipation or at high speed according to the working loads and PVT variations. In the proposed async QDI DVS technique, the completion of operations has been innately detected by the completion detection circuits and no timing margin is required to ensure correct circuit operations. A PSU with three voltage scales is integrated with each pipeline. The supply voltage could be adaptively scaled at each pipeline stage according to their corresponding working loads and make energy saving more efficient in this pipeline-granular manner. The proposed async QDI DVS was free of the error-data-vulnerability compared with the reported DVS techniques, as robustness of the circuits was ensured by QDI attribute. The proposed async QDI DVS technique has been applied to FIR filter. The results showed that the proposed async QDI DVS techniques provide $3.53\times$, $2\times$ and $1\times$ of minimum power dissipation and featured $6.82\times$ lower power dissipation than the conventional async QDI circuit.

6.2 Recommendations for Future Work

This thesis recommends two areas for future research. The two areas relate to the design methodologies proposed in this thesis and extend the research of async QDI

digital design.

6.2.1 Co-synthesize Synchronous and Asynchronous Systems from One Specification

The design trend of future digital systems is predicted to embrace sync and async approaches simultaneously in view of their respective advantages. However, most of the reported sync-async co-design methodologies are still not well developed. Hence, it is recommended to further investigate this area of research.

The preliminary sync-async co-design flow is depicted in Fig. 6.1. The co-design flow is largely partitioned into two parts. Part I is the front end design flow. Part II is the sync-async co-design flow. Inside the Sync-async co-design flow, the overall overhead will be estimated, or conditions will be specified. Thereafter, the flow will partition the specification into sync domain and async domain. The number of domains will depend on global planning of systems to meet the requirements of robustness and overall overhead. The sync and async domains will be synthesized by the standard EDA synthesis tools and the proposed async synthesis tool in Chapter 3 of this thesis respectively. Simultaneously, the sync-async interfaces will be automatically generated. Then, all these netlist will be integrated together to obtain the sync-async netlist. The netlist will go through functional test and be placed and routed. The final layout will be investigated if it meets the requirements. If not, the flow will start from the beginning of Part II to calibrate the conditions accordingly.

In the co-design flow, the challenge to effectively partition sync-async domains is encountered. The effective partition has to take both the robustness and overhead into

Fig. 6.1 Design flow of Sync-async Co-synthesis

account. For robustness, the PVT variations and the signal propagation distance has to be considered. The overheads occurred in both QDI circuits and sync-async interfaces have to be considered. With the two parameters of robustness and overhead, an

optimization algorithm is required to automatically lead the sync-async systems to global optimal solution. Another issue requiring careful consideration is how to effectively design sync-async interfaces to cope with the occurrence of metastability.

6.2.2 Dynamic Voltage Scaling Technique on Sync-async Systems

The next recommendation for future work is to apply DVS technique to the sync-async systems synthesized by the sync-async co-design flow to achieve globally minimum power dissipation. The DVS technique mainly applies to three main types of circuits, namely sync circuits, sync-async interface circuits and async circuits.

The DVS technique for async circuits has been presented in this thesis. For the sync environment, the DVS technique can be explored. The QDI attribute in this thesis can be partially/fully integrated into the DVS technique in sync systems to achieve circuit operational robustness. The DVS for the sync-async interface circuits is controlled by the clock signal of sync circuit and the completion detection signal of the interface circuit.

Bibliography

- [1] J. Bourne, R. Schupbach, B. Hollosi, D. Jia, A. Lostetter, and H. A. Mantooth, "Ultra-Wide Temperature (-230°C to 130°C) DC-Motor Drive with SiGe Asynchronous Controller," in *IEEE Aerospace Conference*, 2008, pp. 1-15.
- [2] B. W. Hunt, K. S. Stevens, B. W. Suter, and D. S. Gelosh, "A single chip low power asynchronous implementation of an FFT algorithm for space applications," in *Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1998, pp. 216-223.
- [3] H. Miao, L. Hai, C. Yiran, W. Qing, and G. S. Rose, "BSB training scheme implementation on memristor-based circuit," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2013, pp. 80-87.
- [4] T. Reece and W. H. Robinson, "Hardware Trojans: The defense and attack of integrated circuits," in *IEEE 29th International Conference on Computer Design*, 2011, pp. 293-296.
- [5] A. B. Uan-Zo-li, V. I. Chuev, and S. I. Volsky, "A design of defense circuits of high-frequency thyristored inverters," in *Sixth International Conference on Power Electronics and Variable Speed Drives*, 1996, pp. 336-341.
- [6] I. W. Chung and Y. C. Liang, "A low-cost photovoltaic energy harvesting circuit for portable devices," in *IEEE Ninth International Conference on Power Electronics and Drive Systems*, 2011, pp. 334-339.
- [7] M. Guarnieri, V. Di Noto, and F. Moro, "A Dynamic Circuit Model of a Small Direct Methanol Fuel Cell for Portable Electronic Devices," *IEEE Transactions on Industrial Electronics*, vol. 57, pp. 1865-1873, 2010.
- [8] T. Lin, K. S. Chong, J. S. Chang, and B. H. Gwee, "An Ultra-Low Power Asynchronous-Logic In-Situ Self-Adaptive V_{DD} System for Wireless Sensor Networks," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 573-586, 2013.
- [9] J. Shouling and C. Zhipeng, "Distributed Data Collection in Large-Scale Asynchronous Wireless Sensor Networks Under the Generalized Physical Interference Model," *IEEE/ACM Transactions on Networking*, vol. 21, pp. 1270-1283, 2013.
- [10] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, "Sensor Search Techniques for Sensing as a Service

- Architecture for the Internet of Things," *IEEE Journal of Sensors*, vol. 14, pp. 406-420, 2014.
- [11] S. Tozlu, M. Senel, M. Wei, and A. Keshavarzian, "Wi-Fi enabled sensors for internet of things: A practical approach," *IEEE Communications Magazine*, vol. 50, pp. 134-143, 2012.
 - [12] F. Zhuo, Z. Zhiyu, and L. Peng, "Parallel On-Chip Power Distribution Network Analysis on Multi-Core-Multi-GPU Platforms," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 19, pp. 1823-1836, 2011.
 - [13] Z. T. Deniz, Y. Leblebici, and E. A. Vittoz, "On-Line Global Energy Optimization in Multi-Core Systems Using Principles of Analog Computation," *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 1593-1606, 2007.
 - [14] *International Technology Roadmap for Semiconductors* www.itrs.net.
 - [15] K. S. Chong, K. L. Chang, B. H. Gwee, and J. S. Chang, "Synchronous-Logic and Globally-Asynchronous-Locally-Synchronous (GALS) Acoustic Digital Signal Processors," *IEEE Journal of Solid-State Circuits*, vol. 47, pp. 769-780, 2012.
 - [16] Y. Jing, T. L. Naing, and R. W. Brodersen, "A 1 GS/s 6 Bit 6.7 mW Successive Approximation ADC Using Asynchronous Processing," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1469-1478, 2010.
 - [17] K. S. Chong, B. H. Gwee, and J. S. Chang, "Energy-Efficient Synchronous-Logic and Asynchronous-Logic FFT/IFFT Processors," *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 2034-2045, 2007.
 - [18] C. Piguet, "Logic synthesis of race-free asynchronous CMOS circuits," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 371-380, 1991.
 - [19] G. M. Jacobs and R. W. Brodersen, "A fully asynchronous digital signal processor using self-timed circuits," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 1526-1537, 1990.
 - [20] S. R. Whitaker and G. K. Maki, "Pass-transistor asynchronous sequential circuits," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 71-78, 1989.
 - [21] J. Sparso and S. Furber, *Principles of asynchronous circuit design: a systems perspective*: Kluwer Academic Pub, 2001.
 - [22] W. F. McLaughlin, A. Mitra, and S. M. Nowick, "Asynchronous Protocol Converters for Two-Phase Delay-Insensitive Global Communication," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 17, pp. 923-928,

2009.

- [23] L. Jia, F. Peper, S. Adachi, and S. Mashiko, "Universal delay-insensitive systems with buffering lines," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, pp. 742-754, 2005.
- [24] S. C. Leung and H. F. Li, "A syntax-directed translation for the synthesis of delay-insensitive circuits," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 2, pp. 196-210, 1994.
- [25] A. Saifhashemi, D. Hand, P. A. Beerel, W. Koven, and H. Wang, "Performance and Area Optimization of a Bundled-Data Intel Processor through Resynthesis," in *IEEE 20th International Symposium on Asynchronous Circuits and Systems*, 2014, pp. 110-111.
- [26] M. Imai and T. Nanya, "A novel design method for asynchronous bundled-data transfer circuits considering characteristics of delay variations," in *IEEE International Symposium on Asynchronous Circuits and Systems*, 2006, pp. 10 pp.-77.
- [27] S. Tugsinavisut, H. Youpyo, K. Daewook, K. Kyeounsoo, and P. A. Beerel, "Efficient asynchronous bundled-data pipelines for DCT matrix-vector multiplication," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, pp. 448-461, 2005.
- [28] C. Ren-Der and J. Jer-Min, "STG-level decomposition and resynthesis of speed-independent circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, pp. 1751-1763, 2002.
- [29] A. Kondratyev, J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev, "Logic decomposition of speed-independent circuits," *IEEE Proceedings*, vol. 87, pp. 347-362, 1999.
- [30] P. A. Beerel, H. Cheng-Ta, and S. Wadekar, "Estimation of energy consumption in speed-independent control circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 672-680, 1996.
- [31] R. Zhou, K. S. Chong, B. H. Gwee, and J. S. Chang, "A Low Overhead Quasi-Delay-Insensitive (QDI) Asynchronous Data Path Synthesis Based on Microcell-Interleaving Genetic Algorithm (MIGA)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 989-1002, 2014.
- [32] I. Lemberski, "LUT-oriented dual-rail quasi-delayinsensitive logic synthesis," *Electronics Letters*, vol. 50, pp. 503-505, 2014.

- [33] K. L. Chang, J. S. Chang, B. H. Gwee, and K. S. Chong, "Synchronous-Logic and Asynchronous-Logic 8051 Microcontroller Cores for Realizing the Internet of Things: A Comparative Study on Dynamic Voltage Scaling and Variation Effects," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, pp. 23-34, 2013.
- [34] K. Weidong, Z. Peiyi, J. S. Yuan, and R. F. DeMara, "Design of Asynchronous Circuits for High Soft Error Tolerance in Deep Submicrometer CMOS Circuits," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 18, pp. 410-422, 2010.
- [35] R. O. Ozdag and P. A. Beerel, "An Asynchronous Low-Power High-Performance Sequential Decoder Implemented With QDI Templates," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, pp. 975-985, 2006.
- [36] S. Mohammadi, S. Furber, and J. Garside, "Designing robust asynchronous circuit components," *IEEE Proceedings*, vol. 150, pp. 161-166, 2003.
- [37] M. Neidengard and B. A. Minch, "Hysteretic threshold logic and quasi-delay insensitive asynchronous design," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, pp. 1423-1428, 2002.
- [38] T. Nanya, Y. Ueno, H. Kagotani, M. Kuwako, and A. Takamura, "TITAC: design of a quasi-delay-insensitive microprocessor," *IEEE Design & Test of Computers*, vol. 11, pp. 50-63, 1994.
- [39] A. J. Martin, *The limitations to delay-insensitivity in asynchronous circuits*: New York: Springer-Verlag, 1990.
- [40] A. J. Martin and M. Nystrom, "Asynchronous Techniques for System-on-Chip Design," *IEEE Proceedings*, vol. 94, pp. 1089-1120, 2006.
- [41] S. Keller, M. Katelman, and A. J. Martin, "A Necessary and Sufficient Timing Assumption for Speed-Independent Circuits," *IEEE Symposium on Asynchronous Circuits and Systems*, 2009, pp. 65-76.
- [42] A. Saifhashemi, H.-H. Huang, and P. A. Beerel, "Reconditioning: Automatic Power Optimization of QDI Circuits," *IEEE International Symposium on Asynchronous Circuits and Systems*, 2014, pp. 77-84.
- [43] N. Imam, F. Akopyan, J. Arthur, P. Merolla, R. Manohar, and D. S. Modha, "A Digital Neurosynaptic Core Using Event-Driven QDI Circuits," *IEEE International Symposium on Asynchronous Circuits and Systems*, 2012, pp. 25-32.
- [44] P. A. Beerel, R. O. Ozdag, and M. Ferretti, *A designer's guide to asynchronous*

VLSI: Cambridge University Press, 2010.

- [45] A. Kondratyev and K. Lwin, "Design of Asynchronous Circuits Using Synchronous CAD Tools," *IEEE Design & Test of Computers*, vol. 19, no.4, pp. 107-117, 2002.
- [46] T. Anh Thien, T. Dean Nguyen, and B. Baas, "A Reconfigurable Source-Synchronous On-Chip Network for GALS Many-Core Platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 897-910, 2010.
- [47] L. Kangmin, L. Se-Joong, and Y. Hoi-Jun, "Low-power network-on-chip for high-performance SoC design," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, pp. 148-160, 2006.
- [48] M. Alioto, M. Poli, and S. Rocchi, "A General Power Model of Differential Power Analysis Attacks to Static Logic Circuits," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 18, pp. 711-724, 2010.
- [49] M. Alioto, L. Giancane, G. Scotti, and A. Trifiletti, "Leakage Power Analysis Attacks: A Novel Class of Attacks to Nanometer Cryptographic Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, pp. 355-367, 2010.
- [50] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, pp. 666-677, 1978.
- [51] K. van Berkel, J. Kessels, M. Roncken, R. Saeijs, and F. Schalijs, "VLSI-programming language Tangram and its translation into handshake circuits," Amsterdam, Neth, 1992, pp. 384-389.
- [52] D. Edwards and A. Bardsley, "Balsa: An asynchronous hardware synthesis language," *The Computer Journal*, vol. 45, p. 12, 2002.
- [53] A. Bardsley, "Implementing Balsa handshake circuits," PHD dissertation, the University of Manchester, 2000.
- [54] G. K. Theodoropoulos, G. K. Tsakogiannis, and J. V. Woods, "Occam: an asynchronous hardware description language?," *Proceedings of the 23rd EUROMICRO Conference*, 1997, pp. 249-256.
- [55] C. A. R. Hoare, *Communicating sequential processes* vol. 178: Prentice-hall Englewood Cliffs, 1985.
- [56] V. Akella and G. Gopalakrishnan, "SHILPA: a high-level synthesis system for self-timed circuits," *IEEE/ACM International Conference on Computer-Aided Design*, 1992, pp. 587-591.

- [57] H. Barringer, D. Fellows, G. Gough, and A. Williams, "Rainbow: Development, Simulation and Analysis Tools for Asynchronous Micropipeline Hardware Design," *Computer Journal*, vol. 45, pp. 2-11, 2002.
- [58] H. Barringer, D. Fellows, G. Gough, and A. Williams, "Abstract modelling of asynchronous micropipeline systems using Rainbow," *Proceeding of Computer Hardware Description Languages and their Applications*, 1997, pp. 285-304.
- [59] T. Chelcea and S. M. Nowick, "Resynthesis and peephole transformations for the optimization of large-scale asynchronous systems," *Proceedings Design Automation Conference*, 2002, pp. 405-410.
- [60] G. Gopalakrishnan, P. Kudva, and E. Brunvand, "Peephole optimization of asynchronous macromodule networks," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 7, pp. 30-37, 1999.
- [61] L. A. Plana, S. Taylor, and D. Edwards, "Attacking control overhead to improve synthesised asynchronous circuit performance," *IEEE International Conference on Computer Design*, 2005, pp. 703-710.
- [62] T. Murata, "Petri nets: Properties, analysis and applications," *IEEE Proceedings*, vol. 77, pp. 541-580, 1989.
- [63] L. Kuan-Jen and K. Chi-Wen, "On synthesis of speed-independent circuits at STG level," *Proceedings of Design Automation Conference*, 1997, pp. 619-624.
- [64] D. Prothero, *Modelling and implementation of petri nets using VHDL*. Boston: Kluwer Academic Publishers, 2000.
- [65] I. Blunno and L. Lavagno, *Deriving signal transition graphs from behavioral verilog HDL: Hardware Design and Petri Nets*, Ed. Kluwer Academic Publishers, 1999.
- [66] I. Blunno and L. Lavagno, "Designing an asynchronous microcontroller using Pipefitter," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 12, pp. 696-699, 2004.
- [67] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems*, vol. 3, pp. 315-325, 1997.
- [68] H. Jacobson, E. Brunvand, G. Gopalakrishnan, and P. Kudva, "High-level asynchronous system design using the ACK framework," *Proceedings of Advanced Research in Asynchronous Circuits and Systems*, 2000, pp. 93-103.

- [69] I. Blunno and L. Lavagno, "Automated synthesis of micro-pipelines from behavioral Verilog HDL," *Proceedings of Advanced Research in Asynchronous Circuits and Systems*, 2000, pp. 84-92.
- [70] M. A. Pena and J. Cortadella, "Combining process algebras and Petri nets for the specification and synthesis of asynchronous circuits," *Proceedings of Advanced Research in Asynchronous Circuits and Systems*, 1996, pp. 222-232.
- [71] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, "Desynchronization: Synthesis of asynchronous circuits from synchronous specifications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 1904-1921, 2006.
- [72] M. Simlastik, V. Stopjakova, L. Majer, and P. Malik, "Clockless Implementation of LEON2 for Low-Power Applications," *Design and Diagnostics of Electronic Circuits and Systems*, 2007, pp. 1-4.
- [73] M. Ligthart, K. Fant, R. Smith, A. Taubin, and A. Kondratyev, "Asynchronous design using commercial HDL synthesis tools," in *Proceedings of Advanced Research in Asynchronous Circuits and Systems*, 2000, pp. 114-125.
- [74] S. C. Smith and J. Di, "Designing asynchronous circuits using NULL convention logic (NCL)," *Synthesis Lectures on Digital Circuits and Systems*, vol. 4, pp. 1-96, 2009.
- [75] T. Chelcea, G. Venkataramani, and S. C. Goldstein, "Area Optimizations for Dual-Rail Circuits Using Relative-Timing Analysis," *IEEE International Symposium on Asynchronous Circuits and Systems*, 2007, pp. 117-128.
- [76] J. Cheoljoo and S. M. Nowick, "Optimization of Robust Asynchronous Circuits by Local Input Completeness Relaxation," *Asia and South Pacific Design Automation Conference*, 2007, pp. 622-627.
- [77] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, "Coping with the variability of combinational logic delays," *IEEE International Conference on Computer Design*, 2004, pp. 505-508.
- [78] Z. Yu, D. Sokolov, and A. Yakovlev, "Cost-aware synthesis of asynchronous circuits based on partial acknowledgement," *IEEE/ACM International Conference on Computer-Aided Design*, 2006, pp. 158-163.
- [79] M. Nakai, S. Akui, K. Seno, T. Meguro, T. Seki, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano, and M. Shimura, "Dynamic voltage and frequency management for a low-power embedded microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 28-35, 2005.
- [80] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic

- voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuit*, vol. 35, pp. 1571-1580, 2000.
- [81] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai, and T. Furuyama, "Variable supply-voltage scheme for low-power high-speed CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 454-462, 1998.
 - [82] Z. Xiaoxiao, F. Boussaid, and A. Bermak, "32 Bit x32 Bit Multiprecision Razor-Based Dynamic Voltage Scaling Multiplier With Operands Scheduler," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 22, pp. 759-770, 2014.
 - [83] P. N. Whatmough, S. Das, and D. M. Bull, "A Low-Power 1-GHz Razor FIR Accelerator With Time-Borrow Tracking Pipeline and Approximate Error Correction in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 49, pp. 84-94, 2014.
 - [84] S. Das, G. S. Dasika, K. Shivashankar, and D. Bull, "A 1 GHz Hardware Loop-Accelerator With Razor-Based Dynamic Adaptation for Energy-Efficient Operation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, pp. 2290-2298, 2014.
 - [85] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. M. Harris, D. Blaauw, and D. Sylvester, "Bubble Razor: Eliminating Timing Margins in an ARM Cortex-M3 Processor in 45 nm CMOS Using Architecturally Independent Error Detection and Correction," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 66-81, 2013.
 - [86] D. Ernst, S. Das, L. Seokwoo, D. Blaauw, T. Austin, T. Mudge, K. Nam Sung, and K. Flautner, "Razor: circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, pp. 10-20, 2004.
 - [87] S. Das, C. Tokunaga, S. Pant, M. Wei-Hsiang, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw, "RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 32-48, 2009.
 - [88] S. Das, D. Roberts, L. Seokwoo, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "A self-tuning DVS processor using delay-error detection and correction," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 792-804, 2006.
 - [89] B. Rahbaran and A. Steininger, "Is Asynchronous Logic More Robust Than Synchronous Logic?," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, pp. 282-294, 2009.

- [90] M. M. Wu and M. C. Loui, "Modeling robust asynchronous communication protocols with finite-state machines," *IEEE Transactions on Communications*, vol. 41, pp. 492-500, 1993.
- [91] J. Ebergen and R. Berks, "Response-time properties of linear asynchronous pipelines," *IEEE Proceedings*, vol. 87, pp. 308-318, 1999.
- [92] D. Kearney and N. W. Bergmann, "Performance evaluation of asynchronous logic pipelines with data dependent processing delays," *Proceedings of Asynchronous Design Methodologies*, 1995, pp. 4-13.
- [93] "IEEE Standard VHDL Language Reference Manual," *IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002)*, pp. c1-626, 2009.
- [94] "IEEE Standard Verilog Hardware Description Language," *IEEE Std 1364-2001*, pp. 0_1-856, 2001.
- [95] M. Renaudin, P. Vivet, and F. Robin, "A design framework for asynchronous/synchronous circuits based on CHP to HDL translation," *Proceedings of Advanced Research in Asynchronous Circuits and Systems*, 1999, pp. 135-144.
- [96] C. F. Law, B. H. Gwee, and J. S. Chang, "Modeling and Synthesis of Asynchronous Pipelines," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 19, pp. 682-695, 2011.
- [97] J. J. Watkins and J. Robin, "Graphs: An Introductory Approach," ed: John Wiley & Sons, 1990.
- [98] M. Najibi and P. A. Beerel, "Slack matching mode-based asynchronous circuits for average-case performance," *IEEE/ACM International Conference on Computer-Aided Design*, 2013, pp. 219-225.
- [99] G. Gill, V. Gupta, and M. Singh, "Performance estimation and slack matching for pipelined asynchronous architectures with choice," *IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 449-456.
- [100] P. A. Beerel, A. Lines, M. Davies, and K. Nam-Hoon, "Slack matching asynchronous designs," *IEEE International Symposium on Asynchronous Circuits and Systems*, 2006, pp. 11 pp.-194.
- [101] Y. Shiu-Yin and C. Chi Kin, "A Genetic Algorithm That Adaptively Mutates and Never Revisits," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 454-472, 2009.
- [102] Y. Yang and X. Yu, "Cooperative Coevolutionary Genetic Algorithm for Digital IIR Filter Design," *IEEE Transactions on Industrial Electronics*, vol.

54, pp. 1311-1318, 2007.

- [103] H. Ali, A. Doucet, and D. I. Amshah, "GSR: A New Genetic Algorithm for Improving Source and Channel Estimates," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, pp. 1088-1098, 2007.
- [104] B. H. Gwee and J. S. Chang, "A hybrid genetic hill-climbing algorithm for four-coloring map problems," *Design and application of hybrid intelligent systems*, ed: IOS Press, 2003, pp. 252-261.
- [105] J. A. Vasconcelos, J. A. Ramirez, R. H. C. Takahashi, and R. R. Saldanha, "Improvements in genetic algorithms," *IEEE Transactions on Magnetics*, vol. 37, pp. 3414-3417, 2001.
- [106] S. C. Ng, S. H. Leung, C. Y. Chung, A. Luk, and W. H. Lau, "The genetic search approach. A new learning algorithm for adaptive IIR filtering," *IEEE Signal Processing Magazine*, vol. 13, pp. 38-46, 1996.
- [107] T. Wei, S. Zhenwei, and W. Ying, "Regularized Simultaneous Forward-Backward Greedy Algorithm for Sparse Unmixing of Hyperspectral Data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, pp. 5271-5288, 2014.
- [108] L. Shaobo, R. Yuanhua, S. Xingping, and X. Zongben, "Learning Capability of Relaxed Greedy Algorithms," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, pp. 1598-1608, 2013.
- [109] B. Babadi and V. Tarokh, "GADIA: A Greedy Asynchronous Distributed Interference Avoidance Algorithm," *IEEE Transactions on Information Theory*, vol. 56, pp. 6228-6252, 2010.
- [110] A. Tkacenko and P. P. Vaidyanathan, "Iterative greedy algorithm for solving the FIR paraunitary approximation problem," *IEEE Transactions on Signal Processing*, vol. 54, pp. 146-160, 2006.
- [111] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 51-65, 2005.
- [112] S. J. Reeves, "An efficient implementation of the backward greedy algorithm for sparse signal reconstruction," *IEEE Signal Processing Letters*, vol. 6, pp. 266-268, 1999.
- [113] J. Yu and J. Zhong-Ping, "Robust Adaptive Dynamic Programming and Feedback Stabilization of Nonlinear Systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, pp. 882-893, 2014.

- [114] J. Nascimento and W. B. Powell, "An Optimal Approximate Dynamic Programming Algorithm for Concave, Scalar Storage Problems With Vector-Valued Controls," *IEEE Transactions on Automatic Control*, vol. 58, pp. 2995-3010, 2013.
- [115] T. Guangming, S. Ninghui, and G. R. Gao, "Improving Performance of Dynamic Programming via Parallelism and Locality on Multicore Architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, pp. 261-274, 2009.
- [116] R. C. Chen and G. L. Blankenship, "Dynamic programming equations for discounted constrained stochastic control," *IEEE Transactions on Automatic Control*, vol. 49, pp. 699-709, 2004.
- [117] H. Ney and S. Ortmanns, "Progress in dynamic programming search for LVCSR," *IEEE Proceedings*, vol. 88, pp. 1224-1240, 2000.
- [118] Z. Liming, C. Huiyou, and X. Ruitian, "Equal-Width Partitioning Roulette Wheel Selection in Genetic Algorithm," *Conference on Technologies and Applications of Artificial Intelligence*, 2012, pp. 62-67.
- [119] C. Ruey-Maw and F. Chun-Han, "Intrusion detection analysis by integrating roulette wheel and pseudo-random into back propagation networks," *International Conference on Machine Learning and Cybernetics*, 2011, pp. 751-756.
- [120] H. Yi and J. Chuanbao, "Clonal Selection Algorithm with Adaptive Mutation and Roulette Wheel Selection," *International Conference on Computational Intelligence and Security Workshops*, 2007, pp. 93-96.
- [121] H. Someya, "Striking a Mean- and Parent-Centric Balance in Real-Valued Crossover Operators," *IEEE Transactions on Evolutionary Computation*, vol. 17, pp. 737-754, 2013.
- [122] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Diversity Improvement by Non-Geometric Binary Crossover in Evolutionary Multiobjective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 985-998, 2010.
- [123] Z. Mengjie, G. Xiaoying, and L. Weijun, "A New Crossover Operator in Genetic Programming for Object Classification," *IEEE Transactions on Cybernetics*, vol. 37, pp. 1332-1343, 2007.
- [124] G. Wenying and C. Zhihua, "Differential Evolution With Ranking-Based Mutation Operators," *IEEE Transactions on Cybernetics*, vol. 43, pp. 2066-2081, 2013.

- [125] G. R. Raidl, G. Koller, and B. A. Julstrom, "Biased mutation operators for subgraph-selection problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 145-156, 2006.
- [126] The ISCAS-85 set is available at: <http://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html>.
- [127] R. Bellman, "Dynamic Programming and Lagrange Multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, pp. 767-769, 1956.
- [128] Z. Kejiu, L. Shiguo, T. X. Wu, and I. Batarseh, "New Insights on Dynamic Voltage Scaling of Multiphase Synchronous Buck Converter: A Comprehensive Design Consideration," *IEEE Transactions on Power Electronics*, vol. 29, pp. 1927-1940, 2014.
- [129] L. Hyun-Woo, K. Ki-Han, C. Young-Kyoung, S. Ju-Hwan, P. Nak-Kyu, K. Kwan-Weon, K. Chulwoo, C. Young-Jung, and C. Byong-Tae, "A 1.6 V 1.4 Gbp/s/pin Consumer DRAM With Self-Dynamic Voltage Scaling Technique in 44 nm CMOS Technology," *IEEE Journal of Solid-State Circuits*, vol. 47, pp. 131-140, 2012.
- [130] X. Shangyang, Q. Weihong, G. Miller, T. X. Wu, and I. Batarseh, "An Active Compensator Scheme for Dynamic Voltage Scaling of Voltage Regulators," *IEEE Transactions on Power Electronics*, vol. 24, pp. 307-311, 2009.
- [131] X. Changjiu, L. Yung-Hsiang, and L. Zhiyuan, "Dynamic Voltage Scaling for Multitasking Real-Time Systems With Uncertain Execution Time," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 1467-1478, 2008.
- [132] B. H. Calhoun and A. P. Chandrakasan, "Ultra-dynamic Voltage scaling (UDVS) using sub-threshold operation and local Voltage dithering," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 238-245, 2006.
- [133] Y. Lin and Q. Gang, "Analysis of energy reduction on dynamic voltage scaling-enabled systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 1827-1837, 2005.
- [134] N. Chabini and W. Wolf, "Reducing dynamic power consumption in synchronous sequential digital designs using retiming and supply voltage scaling," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 12, pp. 573-589, 2004.
- [135] K. J. Nowka, G. D. Carpenter, E. W. MacDonald, H. C. Ngo, B. C. Brock, K. I. Ishii, T. Y. Nguyen, and J. L. Burns, "A 32-bit PowerPC system-on-a-chip with

support for dynamic voltage scaling and dynamic frequency scaling," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 1441-1447, 2002.