Ultra-low-power Design and Implementation of Application-specific Instruction-set Processors for Ubiquitous Sensing and Computing

NING MA

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i Elektronik och Datorsystem onsdag den 4 november 2015 klockan 10.00 i Sal B, Electrum, Kungl Tekniska högskolan, Kista 164 40, Stockholm.

© Ning Ma, September 2015

Tryck: Universitetsservice US AB
Abstract

The feature size of transistors keeps shrinking with the development of technology, which enables ubiquitous sensing and computing. However, with the breakdown of Dennard scaling caused by the difficulties for further lowering supply voltage, the power density increases significantly. The consequence is that, for a given power budget, the energy efficiency must be improved for hardware resources to maximize the performance. Application-specific integrated circuits (ASICs) obtain high energy efficiency at the cost of low flexibility for various applications, while general-purpose processors (GPPs) gain generality at the expense of efficiency.

To provide both high energy efficiency and flexibility, this dissertation explores the ultra-low-power design of application-specific instruction-set processors (ASIP) for ubiquitous sensing and computing. Two application scenarios, i.e. high-throughput compute-intensive processing for multimedia and low-throughput low-cost processing for Internet of Things (IoT) are implemented in the proposed ASIPs.

Multimedia stream processing for human-computer interaction is always featured with high data throughput. To design processors for networked multimedia streams, customizing application-specific accelerators controlled by the embedded processor is exploited. By abstracting the common features from multiple coding algorithms, video decoding accelerators are implemented for networked multi-standard multimedia stream processing. Fabricated in 0.13 \( \mu \)m CMOS technology, the processor running at 216 MHz is capable of decoding real-time high-definition video streams with power consumption of 414 mW.

When even higher throughput is required, such as in multi-view video coding applications, multiple customized processors will be connected with an on-chip network. Design problems are further studied for selecting the capability of single processors, the number of processors, the capacity of communication network, as well as the task assignment schemes.

In the IoT scenario, low processing throughput but high energy efficiency and adaptability are demanded for a wide spectrum of devices. In this case, a tile processor including a multi-mode router and dual cores is proposed and implemented. The multi-mode router supports both circuit and wormhole switching to facilitate inter-silicon extension for providing on-demand performance. The control-centric dual-core architecture uses control words to directly manipulate all hardware resources. Such a mechanism avoids introducing complex control logics, and the hardware utilization is increased. Programmable control words enable reconfigurability of the processor for supporting general-purpose ISAs, application-specific instructions and dedicated implementations. The idea of reducing global data transfer also increases the energy efficiency. Finally, a single tile processor together with network of bare dies and network of packaged chips has been demonstrated as the result. The processor implemented in 65 nm low leakage CMOS technology and achieves the energy efficiency of 101.4 GOPS/W for each core.
To my family
Acknowledgments

First and foremost, I would like to express my deep gratitude to my advisors: Prof. Li-Rong Zheng, Assoc. Prof. Zhonghai Lu and Dr. Zhuo Zou. I am heartily thankful to Li-Rong for providing me the opportunity to be a doctoral student at KTH and all the support during these years. His vast expertise and broad visions greatly inspire me for the research. I am sincerely grateful to Zhonghai for his professional guidance and constructive suggestions. I appreciate all the discussions with him and I benefit a lot. I am really thankful to Zhuo for his valuable advices from doing research, writing scientific papers to trivia. It is exciting and helpful to work with him.

I would also like to express my gratitude to Dr. Zhibo Pang for his assistance in the early phase of my research. I gained the experience both from engineering and everyday life during that period. I am indebted to Stefan Blixt at Imsys AB for broadening and deepening my knowledge about processor architectures through plentiful technique discussions. His patience and enthusiasm greatly encourage me.

I would like to thank Dr. Qiang Chen and Dr. Geng Yang for sharing experiences and giving suggestions both on work and life. Many thanks to the previous and current colleagues at KTH and Imsys for their all kinds of help to make the Ph.D journey pleasurable.

I am grateful to Prof. Hannu Tenhunen for reviewing the draft of this dissertation. Many thanks to Dávid Juhász and Nicholas Rose for proofreading this dissertation and providing so many valuable comments. I would also like to express my sincere acknowledgment to Prof. Jari Nurmi from Tampere University of Technology for acting as my opponent, as well as to Prof. Peeter Ellervee from Tallinn University of Technology, Adj. Prof. Lauri Koskinen from University of Turku, and Assoc. Prof. Cristina Rusu from ACREO AB for serving as my committee members.

I also want to thank all the people I have met to make everyday special!

Finally I wish to express my gratitude to my wife Youjia for her understanding, patience and support, to my little boy Kevin for his innocent smiles, to my parents for their unconditional love and constant support. This dissertation is dedicated to you.

Ning Ma
September 2015
Stockholm
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>Arithmetic Logic Unit</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application-specific Integrated Circuit</td>
</tr>
<tr>
<td>ASIP</td>
<td>Application-specific Instruction-set Processor</td>
</tr>
<tr>
<td>CIF</td>
<td>Common Interchange Format</td>
</tr>
<tr>
<td>CISC</td>
<td>Complex Instruction Set Computer</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>CMP</td>
<td>Chip Multiprocessor</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CS</td>
<td>Circuit Switching</td>
</tr>
<tr>
<td>CW</td>
<td>Control Word</td>
</tr>
<tr>
<td>DF</td>
<td>Deblocking Filter</td>
</tr>
<tr>
<td>DM</td>
<td>Data Memory</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>FU</td>
<td>Function Unit</td>
</tr>
<tr>
<td>GOP</td>
<td>Group of Pictures</td>
</tr>
<tr>
<td>GPP</td>
<td>General-purpose Processor</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition</td>
</tr>
<tr>
<td>HEVC</td>
<td>High Efficiency Video Coding</td>
</tr>
<tr>
<td>IM</td>
<td>Instruction Memory</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IQ</td>
<td>Inverse Quantization</td>
</tr>
<tr>
<td>ISA</td>
<td>Instruction Set Architecture</td>
</tr>
<tr>
<td>IT</td>
<td>Inverse Transformation</td>
</tr>
<tr>
<td>MAC</td>
<td>Multiplication and Accumulate unit</td>
</tr>
<tr>
<td>MB</td>
<td>Macro Block</td>
</tr>
<tr>
<td>MC</td>
<td>Motion Compensation</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>MVC</td>
<td>Multi-view Video Coding</td>
</tr>
<tr>
<td>NoC</td>
<td>Network on Chip</td>
</tr>
<tr>
<td>NRE</td>
<td>Non-recurring Engineering</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computer</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-only Memory</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SDRAM</td>
<td>Synchronous Dynamic Random Access Memory</td>
</tr>
<tr>
<td>SIMD</td>
<td>Single Instruction Multiple Data</td>
</tr>
<tr>
<td>SMT</td>
<td>Sequence Mapping Table</td>
</tr>
<tr>
<td>SoC</td>
<td>System on Chip</td>
</tr>
<tr>
<td>UHD</td>
<td>Ultra High Definition</td>
</tr>
<tr>
<td>VC</td>
<td>Virtual Channel</td>
</tr>
<tr>
<td>VCEG</td>
<td>Video Coding Experts Group</td>
</tr>
<tr>
<td>VLD</td>
<td>Variable Length Decoding</td>
</tr>
<tr>
<td>VLIW</td>
<td>Very Long Instruction Word</td>
</tr>
<tr>
<td>VGA</td>
<td>Video Graphics Array</td>
</tr>
<tr>
<td>WS</td>
<td>Wormhole Switching</td>
</tr>
</tbody>
</table>
List of Publications

Papers included in the thesis:


Papers not included in the thesis:


## Contents

<table>
<thead>
<tr>
<th>Contents</th>
<th>xiii</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>xv</td>
</tr>
</tbody>
</table>

### 1 Introduction 1
- 1.1 Background ............................................. 1
- 1.2 Motivation ........................................... 5
- 1.3 Thesis contributions and organization ............... 6

### 2 Low-power application-specific instruction-set processors 11
- 2.1 Low-power techniques ................................. 12
- 2.2 Applications ......................................... 13
- 2.3 Architectures ......................................... 14
- 2.4 Approaches ........................................... 16

### 3 Single-ASIP implementation for multi-standard multimedia processing 21
- 3.1 Introduction ........................................... 21
- 3.2 Design methodology ................................... 22
- 3.3 Video coding structure ............................... 24
- 3.4 Accelerator-enriched architecture ................... 26
- 3.5 Design instance ....................................... 26
- 3.6 Implementation results ............................... 28
- 3.7 Summary ............................................... 29

### 4 Multi-processor architecture exploration for multi-view video decoding 31
- 4.1 Background ........................................... 31
- 4.2 Design space exploration ............................. 34
- 4.3 Communication network ................................. 41
- 4.4 Summary ............................................... 43

### 5 Customizable tile-ASIP implementation for Internet of Things 45
5.1 Background .................................................. 45
5.2 Multi-mode router ............................................. 47
5.3 Control-centric processor ................................. 51
5.4 Tile implementation ......................................... 56
5.5 Summary ..................................................... 57

6 Conclusions .................................................. 59
   6.1 Thesis summary ............................................. 59
   6.2 Future work ............................................... 60

Bibliography .................................................. 63
# List of Figures

1.1 Time line of computers ............................................. 1  
1.2 Trends of video applications ................................. 3  
1.3 One scenario of IoT ............................................. 4  
1.4 Applications and implementations ......................... 6  

2.1 Comparison of ASIC, GPP and ASIP ............................ 12  
2.2 Architectures of ASIPs ......................................... 15  
2.3 Design flow for extending a existing processor ............... 17  
2.4 Design flow for customizing a processor ...................... 18  

3.1 Video coding standards ........................................... 22  
3.2 Design flow .................................................... 23  
3.3 HW/SW partition ................................................ 24  
3.4 General structure of video coding .............................. 25  
3.5 Accelerator-enriched architecture .............................. 26  
3.6 Variety of data transferring .................................... 26  
3.7 Design instance for networked multimedia processing ....... 27  
3.8 Video decoding pipeline ....................................... 28  
3.9 Die photo and power consumption ............................... 29  
3.10 Test system and application scenario ......................... 29  

4.1 Typical prediction structure ..................................... 33  
4.2 An example of 4x4 2D mesh NoC ............................... 33  
4.3 Mapping multimedia applications to NoC ....................... 34  
4.4 Simplified System ............................................. 35  
4.5 Work flow for the system exploration ......................... 36  
4.6 Hierarchical structure of MVC .................................. 37  
4.7 Picture-level parallelism ...................................... 38  
4.8 A possible data flow for picture-level parallel processing 39  
4.9 Design options for picture-level task assignment .......... 39  
4.10 A possible data flow for view-level parallel processing ... 40  
4.11 Design options for view-level task assignment ............ 41  
4.12 Router structures ............................................. 42
5.1 Proposed multi-mode router ........................................ 48
5.2 Maximum throughput with only CS or WS activated ............... 49
5.3 The comparison of throughput and latency .......................... 49
5.4 Areas for routers with different configurations ..................... 50
5.5 Overall throughput (OT) and power efficiency (PE) ............... 51
5.6 Complete structure of the on-chip multi-mode router in one tile ... 51
5.7 Architecture overview ............................................. 52
5.8 Structure of one control word ..................................... 53
5.9 Mapping from applications to the processor ........................ 53
5.10 Comparison of instructions processing ............................ 53
5.11 Control word timing ............................................... 55
5.12 Hierarchical implementation structure ............................ 55
5.13 Die photo together with gate count breakdown ..................... 56
5.14 Power consumption under different working modes and conditions 57
5.15 Network on a board and network in a package ..................... 58
Chapter 1

Introduction

1.1 Background

When the first electronic general-purpose computer ENIAC was announced in 1946, it used over 17000 vacuum tubes and covered 140 $m^2$ with power consumption of 174 kW [1]. To program it, manual setting of switches and cables was necessary. The vacuum tubes were later substituted by discrete transistors, and further by integrated circuits. The miniaturization and integration of devices greatly decreased the size and power consumption as well as increased performance, which enabled the application of personal computers and portable computers. Figure 1.1 shows a brief history of computers.

With the utilization of CMOS process and following Moore’s law, the monolithic integrated circuit can hold more and more transistors since the feature size scales down around every two years [2]. Table 1.1 gives the power scaling for fixed-sized chips. Assuming a scaling factor of $S$, if the size of chips remains the same as that

![Figure 1.1: Time line of computers](image-url)
of the previous generation, the quantity of transistors and the maximum working frequency scale with the factor of $S^2$ and $S$, respectively. The capability of chips with the same size will be improved by a factor of $S^3$. To keep the power consumption constant, the supply voltage needs to be scaled down correspondingly, which is called Dennard scaling [3]. During this period, the performance of processors is increased by raising the working clock frequency with the support of more and more sophisticated architectures, e.g. deeper pipelines, speculation, hierarchical caches, etc. powered by augmented transistors. The powerful yet small processors boost the miniaturization of computers, and the mobile phones are even more capable than the computers in old generations.

The further miniaturization brings about the mergence of computers to the ambient, and facilitates the ubiquitous sensing and computing empowered by the smart electronics. Application scenarios can be both high data throughput applications such as the human-computer interaction and low data throughput applications such as the Internet of Things (IoT). Ultra-low-power processors are highly demanded for the miniaturized devices. On the other hand, the transistor size is continuously shrinking, while the supply voltage cannot be scaled down correspondingly due to the leakage problems, which breaks down the Dennard scaling [4]. As shown in Table 1.1, if all the transistors work at the maximum frequency, the power consumption will increase exponentially. For a given power budget, either the chip size or activity of the transistors should be diminished.

The traditional low-power techniques such as clock gating and power gating, which either lower the working frequency or completely shut down the circuit, will decrease the power consumption while the performance is degraded at the same time. Dynamic voltage and frequency scaling (DVFS) is utilized in [5, 6, 7] to adapt the voltage supply and clock frequency for providing the proper performance for different application scenarios. When high performance is not needed, a sub-threshold supply voltage can even be applied to save the energy at the cost of large delay and slow speed as reported in [8, 9].

In order to lower the power consumption while maximizing the performance, energy efficiency and hardware utilization must be effectively improved in post-Dennard era. The augmented transistors from Moore’s law should be utilized efficiently instead of only increasing the working clock frequency of processors.
Therefore, the architectural level optimization for high energy efficiency becomes critical.

This dissertation explores the design of ultra-low-power application-specific instruction-set processors (ASIP) for ubiquitous sensing and computing in both high-throughput compute-intensive processing of multimedia and low-throughput low-cost processing of IoT scenarios. The low power consumption and high performance are realized by increasing the energy efficiency through customization and specialization at the architectural level for particular scenarios.

High-throughput applications

Multimedia processing in human-computer interactions always needs high performance processors for the extremely large data throughput. In the case of video processing, for example, the resolutions of videos have been increased from CIF (352x288), VGA (640x480) to HD (1920x1080), even to 8K UHD (7680x4320) [10]. Digital Cinema Initiatives (DCI) have already adopted 2K (2048x1080) and 4K (4096x2160) images [11]. NHK is going to do experimental broadcasting for Super-Hi Vision (7680x4320) in 2020 [12]. From another point of view, the included views of video streams can vary from single view through stereo to multiple views as shown in Figure 1.2: Trends of video applications.
in Figure 1.2. 3D video is an important application and having a great impact on our daily life [13, 14, 15]. The increase of resolution and views have resulted in tens or even hundreds of data volume increase. This trend will continue since it is the instinct of people to pursue the higher quality and vivid experience. Considering the varieties of similar applications and rapid updates, flexibility is also a merit for the design.

Low-throughput applications

As one of the enabling technologies for ubiquitous sensing and computing, Internet of Things (IoT) will connect everything in the surrounding seamlessly [16]. There will be billions or trillions of devices embedded in those “things”. Figure 1.3 shows one of the scenarios. Lightweight architectures capable of sensing, processing and communication are highly demanded in extremely low power and low cost manners. Low power consumption is one of the most important characteristics required for IoT devices due to their limited power supply. Furthermore, certain computational capability is still necessary for signal processing and communication. Flexibility and scalability should also be featured for the processors adapting to various and vast number of IoT devices.

Architecture alternatives

Many architectures might be used for the aforementioned application scenarios. General-purpose processors (GPP) have good programmability, and can be easily applied in different applications. However, limited performance and high power consumption make them insufficient for high throughput multimedia processing particularly for portable devices [17, 18]. The low energy efficiency of GPP is also the obstacle for IoT applications. Digital signal processors (DSP) enhance the compu-
tional capability for digital signal processing and keep the good programmability as well [19]. But the performance is still not adequate for sophisticated multimedia applications, and DSPs are not suitable for IoT applications because their capability for general-purpose controlling is weak and they are not energy efficient enough. Application-specific integrated circuits (ASIC) are superior to other architectures from the perspective of both performance and energy efficiency [20, 21, 22, 23, 24]. The drawback is the low flexibility and scalability when trying to adapt to different applications, which results in high design and implementation cost as well as long time-to-market.

Application-specific instruction-set processors (ASIP) can improve the energy efficiency and provide the flexibility as well [25, 26]. The programmable processors empower the adaptability for similar applications, while the customized function units guarantee the necessary performance. The implementation of ASIPs for multimedia and IoT have been reported in [27, 28, 29, 30, 31].

1.2 Motivation

The two mentioned scenarios represent two extremes of the applications. Different strategies are necessary for designing the ultra-low-power customizable energy-efficient processors following the implementation targets. This dissertation will explore and verify the design strategies by proposing and realizing the corresponding processors.

Networked multimedia processing is the performance-demanding application. Furthermore, there are many formats for the multimedia streams. In order to handle the networked multimedia streams, the ASIPs with high performance for high-definition video coding and flexibility for multiple formats should be designed. In this work, video accelerators supporting high-definition multi-standard video decoding are designed and implemented. Those accelerators are controlled by the specific instructions issued by the embedded processor, and both the number and computational capability of the accelerators are adjustable to provide the needed performance.

As introduced in last section, another trend for video applications is the utilization of multiple views for the same scene. The videos from different view directions are recorded simultaneously. Multi-view coding (MVC) has been standardized as an amendment of the H.264/MPEG-4 AVC video coding standard [32], and can be used in 3D video, free view video applications, etc. Compared with the architecture for single-view applications, to process the video with multiple views, performance should be improved by many times. The reasonable solution is to employ multiple processors by exploiting parallelism in the MVC. Considering the huge volume of data exchanged among different processors, an efficient communication architecture is required. Network-on-chip (NoC) based multiple processors are thus used in the proposed architecture for MVC applications in this dissertation.
For the applications discussed above, the design of processors is performance-driven, while from the IoT perspective, high energy efficiency under extremely low power conditions is more important. At the same time, to adapt to different scenarios, the ASIPs should also be featured with flexibility and scalability. Based upon the preceding considerations, a reconfigurable and scalable control-centric dual-core processor with an on-chip multi-mode router is proposed and implemented. The high energy efficiency is achieved by increasing the hardware utilization of functional units, decreasing non-functional units, and reducing excessive global data transferring. The reorganizable functional units enable reconfigurability of the processor, and the on-chip multi-mode router extends scalability of the processor even in the post-fabrication stage. Figure 1.4 summarize the applications, together with the required features and the proposed systems in this dissertation.

1.3 Thesis contributions and organization

This dissertation is organized as follows:
Chapter 2
This chapter will review the related work on the design of low power ASIPs. The possible application scenarios are shown in this chapter, including multimedia, communication, biomedical applications, etc. followed by the implementation architectures. The two approaches for designing ASIPs are given at the end of this chapter.

Chapter 3
The design and implementation of an ASIP for networked multimedia processing are described in this chapter. The design flow is firstly given for software/hardware co-design, followed by analysis of the application. The video processing accelerators are then customized and integrated with the embedded processor cores to construct the high performance and low power processor. The chip implementation and test system are demonstrated as the results.

- **Contributions:** The design strategy of utilizing customizable accelerators together with embedded processors is verified for the applications where high performance and energy efficiency are required. As a result, an accelerator-enriched dual-core architecture is proposed for the performance-demanded applications. A design instance for high-definition multi-standard video decoding is then fabricated and tested with high performance and low power consumption.

The included papers are:


Chapter 4
This chapter explores the design space of multi-processor architecture for multi-view video decoding. It begins with the theoretical analysis on the simplified system, from which the exploration work flow is extracted accordingly. To implement multi-view video decoding, both picture-level and view-level task assignment schemes are introduced and discussed for the homogeneous multi-processor architecture. Based on the exploration platform, the design options for each task assignment scheme are studied and summarized.
• **Contributions:** Multi-processor architecture is proposed and explored for multi-view video applications, which demand even higher performance. To the best of our knowledge, the scalable and flexible architecture for the MVC decoding has not been studied before. It is also found that to achieve the specific performance target, the computation and communication subsystem should be balanced, when all the resources can be utilized efficiently.

The included papers are:


- **Paper V.** Ning Ma, Zhuo Zou, Zhonghai Lu, and Li-Rong Zheng. “Implementing MVC Decoding on Homogeneous NoCs: Circuit Switching or Wormhole Switching,” In *Proceedings of 23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp.387,391, 4-6 March 2015

**Chapter 5**

The energy efficient processor for IoT applications is proposed and implemented in this chapter. It contains two topics: designing multi-mode routers for modular extension and proposing processor architecture for high energy efficiency. The necessity of supporting both circuit and wormhole switching in one router to compose the inter-silicon network is discussed at first, and the corresponding multi-mode router is then introduced. Aiming at increasing the energy efficiency by improving the hardware utilization, the control-centric processor architecture is proposed. The implementation results of the module chip integrating both the processor and router are demonstrated at the end of this chapter.

• **Contributions:** The quantitative comparison of circuit and wormhole switching is presented to give the guidelines for choosing the proper switching mode in a specific scenario. To adapt to various application scenarios, a multi-mode router supporting both circuit and wormhole switching is proposed and implemented. The multi-mode router is further integrated with a reconfigurable dual-core control-centric processor to construct the processor tile. The tile can either be used independently or compose a large scale multi-processor network to provide high performance, energy efficiency and flexibility.

The included papers are:


Chapter 6

This chapter will conclude the dissertation and indicate the future work from processor architecture, programming models to application mapping. Brain-inspired computing will be the target for those work to further decrease the power consumption while achieving high performance.
Chapter 2

Low-power application-specific instruction-set processors

For specific applications, many architectures can be employed as alternative implementations. Application-specific integrated circuit (ASIC) utilizes specialized hardware units and determinate data paths to achieve parallel and dedicated processing. High performance and energy efficiency are the advantages. The weaknesses are poor generality, high design complexity and non-recurring engineering (NRE) cost. Furthermore, as more transistors can be integrated in one single chip, the design complexity has been greatly increased. Verifying and testing such huge scale circuits have led to tremendous challenges for ASIC implementations [33, 34].

On the other hand, general-purpose processors (GPP) serialize the operations of the arithmetic logic unit (ALU) through instructions for various applications. Wide generality is the merit, and the implementation of a specific application is simplified as designing the sequence of instructions for the GPP. The time-to-market is greatly shortened compared with the ASIC implementation. However, GPP implementations suffer from low performance and poor energy efficiency because of the serial processing of instructions and the overhead for fetching and decoding the instructions.

Application-specific instruction-set processors (ASIP) try to gain the balance of performance, efficiency and generality for specific applications. Dedicated hardware accelerators, data paths or processing flows are customized for the related applications to provide sufficient performance and efficiency. Meanwhile, the specialized functions alleviate the overhead of processing instructions in GPP. The instruction-controlled processing or reconfigurable logic guarantees the generality for similar applications, and gains the programmability and flexibility compared with ASICs. The comparisons of ASIC, GPP and ASIP are shown in Figure 2.1.
2.1 Low-power techniques

Decreasing the working frequency or supply voltage is the direct way to reduce power consumption, which lowers performance as well. Clock gating [35, 36, 37] and power gating [38, 39, 40] are often employed to deactivate the circuits when they are not used. Besides, dynamic voltage and frequency scaling (DVFS) tunes operating conditions at a finer grain for the processors to provide proper performance in diverse application scenarios [41]. In [42], the processor can work at full speed with the supply voltage of 1.14 V, frequency of 1 GHz, and consumes 125.3 W. It can also work for low power consumption of 24.7 W at 0.7 V and 125 MHz. Fine-grained adaptive power management is implemented in [43] by applying multiple DVFS configurations based on the real-time power and temperature data collected through on-die power and thermal meters. The purpose is to achieve the maximum performance while maintaining temperature and power under predefined constrains. In [44], an on-chip controller performs per-core DVFS by monitoring workload, temperature, voltage and current dynamically. The voltage supply in [45] can even be lowered to 210 mV for a JPEG encoder working at 5 MHz.

Heterogeneity is another way to lower the power consumption. ARM’s big.LITTLE heterogeneous multi-core processor [46] runs big core with short duration at peak frequency and LITTLE core at the majority of time with moderate operating frequencies. Up to 75 % energy can be saved without decreasing the delivered performance. Similarly, high-performance CPU1 runs the performance-intensive tasks with much higher power consumption, and general applications are mapped to the power efficient CPU2 in [47]. Two architecturally identical but physically different cores are utilized in [48]. One core is implemented using worst-case corners to achieve the target frequency, while the other one uses typical corners to gain energy...
efficiency. The resulted energy savings can be up to 20%. Asynchronous design is proposed in [49] for neural signal processors. Handshaking protocol circuitry is added between function units for data exchanging. Compared with the synchronous design, the asynchronous processor shows a 2.3x reduction in power.

Integrating the customized application-specific function units will gain both low power consumption and high performance. In [50], fine-grained parallelism is implemented by SIMD architectures to accelerate the classic sequence alignment procedures. Together with the multi-core structure, 25x less energy is used while achieving similar performance with a quad-core Intel Core i7 3820 processor. A data compression algorithm is mapped to a specific circuit in [51] for lowering the communication data rate in artificial vision systems. The energy reduction ratios can be up to 85% compared with its base processor. Linear matrix and vector operations are enhanced for adaptive filters in neural coding in [52], where the proposed application-specific instruction-set processors provide high performance and flexibility at low power consumption compared with the implementations on commercial CPUs.

2.2 Applications

ASIPs are implemented for applications such as multimedia processing, communication, health care, etc. High efficiency video coding (HEVC) is the new standard for video compression, and it can achieve very high compression ratio with high implementation complexity. An ASIP is proposed in [53] targeting at accelerating the motion compensation, which is the most compute-intensive task in video coding process. Three different models with diverse hardware configurations are evaluated to show the trade-offs between implementation cost and performance. Dedicated hardware accelerators for inter prediction and entropy coding could be found in the ASIP designed in [54] for H.264/AVC video coding standard. Hardware accelerators work in parallel with the main processor and are coordinated through the customized instructions. Image detection is another application scenario. The algorithm of Histograms for Oriented Gradients is run in the Support Vector Machine implemented by an ASIP in [55]. SIMD instructions, wide memory interfaces and VLIW extensions are added to a RISC processor to achieve high performance, power efficiency and programmability. An audio ASIP is implemented in [56] to decode MPEG-2/4 AAC audio streams. Specific instruction for inverse DCT, Huffman decoding, inverse quantization, etc. are integrated to accelerate the decoding speed.

Multi-standard multi-mode channel coding in wireless communication is another important application scenario for ASIPs. Two turbo decoders are presented in [57] with different design objectives. The TurbASIP supplies the maximum flexibility by supporting all of the different modes in existing wireless communication standards. Diverse parallelism techniques are exploited. The devised pipeline architecture together with dedicated instructions is properly selected for the target
Another decoder TDecASIP aims to increase the efficiency by limiting the flexibility and supporting the turbo codes with related parameters specified in 3GPP LTE, WiMAX and DVB-RCS standards. Both the instruction set architecture and memory organization are simplified to increase the execution efficiency. A multi-standard forward error correction ASIP processor is designed in [58] offering QC-LDPC decoding for WiMAX, Turbo decoding for 3GPP-LTE and Convolutional code decoding. By analyzing the supported decoding algorithms, a general-purpose algorithm that could be further tuned to the specific decoding algorithm is selected, and the common parts are abstracted and implemented by shared functional blocks. Optimized instructions manage the operations and data paths of functional blocks to implement specific decoding algorithms. In [59], a baseband ASIP processor is designed for multi-mode MIMO detection by parameterizing the dedicated hardware block for specific algorithms.

ASIPs can also be found in promising biomedical applications. In [60], a dual-core system is proposed for wearable sensor devices, which employs an ASIP core optimized for processing ECG, EMG, EEG signals and a low power embedded processor as a controller. The energy efficiency is greatly improved compared with the implementation in a general-purpose processor. An integrated SoC is implemented in [61] for digital hearing aids. The embedded ASIP extends a RISC instruction set by adding dedicated hardware accelerators and compacting certain instructions to provide both flexibility and power efficiency for digital audio signal processing. In [51], a RISC processor and run-length coding accelerators compose the ASIP for data compression in artificial vision systems. Special instructions are designed for data transferring and run-length counting. The energy consumption is greatly reduced for both compression and decompression process compared with the implementation in the base RISC processor. DNA sequence alignment is the target application scenario in [62]. SIMD instructions for vector-vector, vector-scalar arithmetic, logic, load/store and control are implemented to exploit fine-grained parallelism in the related algorithms, and coarse-grained parallelism is achieved by utilizing multi-processor architecture for querying multiple sequences.

Besides, in other applications, such as compressed sensing [63], encryption [64], packet classification [65], etc. which are scenarios with emphasis on both flexibility and energy efficiency, the utilization of ASIPs can meet the requirements properly.

### 2.3 Architectures

Figure 2.2 shows the architectures that are utilized for ASIPs. Dedicated hardware accelerators can serve as the execution units invoked by application-specific instructions as shown in Figure 2.2(a). In [66], a bit stream engine designed for variable length coding in multiple video formats is integrated into a conventional general-purpose processor, where the bitwise computation is the bottle neck. The coding/decoding instructions share common data path with general-purpose instructions but only on different execution paths. The processor gains a performance
increase for variable length coding and maintains the flexibility as well. For the application of software defined radio in [67], a trigonometric math unit, a Viterbi complex unit and a floating-point control law accelerator are coupled to a general-purpose CPU. The ASIP can provide enough performance for the existing complex wireless communication algorithms and shows the probability to implement additional standards.

Single instruction multiple data (SIMD) architectures exploit the data-level parallelism to accelerate compute-intensive tasks. Multiple data are operated with the same operations on multiple execution units simultaneously under the control of one instruction. A vector function unit (VFU) is designed in [68] for Fast Fourier Transform (FFT) used in various wireless communication standards. The VFU can perform sixteen 16-bit MACs, or eight 32-bit MACs simultaneously. Memory bandwidth is enlarged to ease the data transferring between the vector register file and the main memory. In [69], arithmetic, logic, load/store SIMD instructions realize data parallel processing on vectors of 8 or 16 elements according to the size of operated pixel-block in one image when implementing Advanced Video Coding (AVS) standard. The tasks with heavy duty of computation such as motion compensation, deblocking, etc. are greatly accelerated by the SIMD instructions.

Figure 2.2(c) presents the very long instruction word (VLIW) architecture which provides instruction-level parallel processing to increase the execution throughput and hardware utilization. Multiple independent instructions are issued and executed concurrently. In [70], the VLIW architecture is optimized for stereoscopic video applications. The basic instruction processing unit, i.e. vector unit, are replicated to utilize possible instruction parallelism. A VLIW processor is implemented on FPGA for image processing in [71]. The application is realized using a high level programming language at first. By analyzing the generated assembly codes, the independent instructions that can be executed in parallel are bound into the very long instructions.
To further increase the performance, task- or thread-level parallelism can be utilized by multi-core or multiprocessor architectures. In [72], multiple ASIP cores are connected by the butterfly NoC network, and can be dynamically configured through a dedicated bus-based communication structure. By selecting the suitable number of active ASIP cores and the corresponding configuration parameters, multimode and multi-standard turbo decoding have been implemented. Turbo decoders are also implemented in [73] by connecting ASIPs and memories with networks increasing the throughput. The proposed system employs ring-style topology and enables simultaneous data exchange between the ASIPs and memories. Multiple ASIPs are connected through the shared memory in [74] for H.264 video encoding. Each ASIP is responsible for coding macroblocks (MB) in one slice of a picture. The MB processing in different slices are executed in different ASIPs in parallel. Furthermore, several video accelerators are integrated into each ASIP to increase its processing performance. According to the throughput requirement for different video resolutions, the platform supports different configurations of the ASIP cores. A multi-core platform for multi-format video decoding is proposed in [75], the macroblock-row level parallelism is investigated and utilized by multiple ASIP cores. Every single ASIP core is featured with a dual-issue VLIW architecture and designated SIMD instructions for pixel-level acceleration.

2.4 Approaches

ASIPs are implemented by either extending a existing processor or customizing a processor. The former approach strengthens the off-the-shelf processor cores with application-specific function units. Application-specific instructions together with the original general-purpose instruction set compose the final application-specific instruction set. The tool chain for software development is inherited from the existing processor. The design flow of this type of ASIPs is shown in Figure 2.3. Based on the analysis of applications, the time-consuming tasks will be accelerated by the designated instructions, while the controlling task is managed by the general-purpose instructions.

In [50], 56 SIMD instructions for biological sequence alignment are implemented on a 32-bit Harvard RISC processor to efficiently utilize the fine-grained parallelism in the corresponding algorithms. By adding the specialized instruction set in the back end, an already existing compiler is extended to support the new ASIP. 40 carefully crafted specialized instructions for slice header processing are integrated into the base ISA of a 32-bit RISC processor (ARP32) for video decoding in [76]. Each customized instruction is mapped to an intrinsic function by the ARP32 compiler. The base ISA handles general-purpose controlling and processing, while the customized instructions deal with the complex slice header processing functions. The AES encryption algorithm is mapped to the vector units in the ASIP designed in [77]. By integrating the AES vector units in a 16-bit default processor, the proposed ASIP can be used in wireless sensor node applications, and the security
service is provided by the add-on application-specific instructions.

The other implementation approach is to customize a processor, which generates a new processor. The instruction set is tailored to only support the target applications. The control units, execution units, data paths, pipelines are all customized to maximize the performance for a cluster of applications. The software tool chain needs also to be adjusted to fit the new processor. The corresponding design flow is summarized in Figure 2.4. Compared with the previous design methodology, the totally new instruction set and hardware architecture will gain high performance and energy efficiency for the target applications. However, the target applications should include few general-purpose controlling and processing tasks due to the limitation of the instruction set.
In [78], an ASIP designed for elliptic curve cryptography employs a finite-state machine (FSM) to control the execution of eight designated instructions implementing the functions of point addition and doubling. Different algorithms can be realized by programming with the assembly codes and the final program is stored in the internal ROM. Multi-standard forward error correction (FEC) decoding is supported in the SIMD ASIP proposed in [79]. The computational units, data paths, memory system are all customized according to the common features of the FEC algorithms. In order to decode a specific format, the assembly codes are utilized to designate the different fields of the instructions for controlling the computation and data flow. As a result, the proposed ASIP can provides high-throughput decoding of turbo, QC-LDPC and CC codes. The ASIP in [80] implements motion estimation (ME) for video codecs. Three instructions including Sum Absolute Difference (SAD), Compare, and Mode Decision, together with an efficient hardware
architecture, are designed to handle the real-time processing of ME for HD video.

This dissertation utilizes both two of the approaches. Besides, the proposed customizable control-centric processor supports two approaches simultaneously. Either existing general-purpose ISAs or completely customized ISAs can be mapped to the processor as needed.
Chapter 3

Single-ASIP implementation for multi-standard multimedia processing

3.1 Introduction

Large data volume is one of the most important features for multimedia applications. For high-definition (HD) video applications, if the resolution of frames is 1920x1080 and the frame rate is 30 fps, about 93 MB/s raw data is to be processed for the YUV 4:2:0 format. It is a great challenge for both transmitting and storing the data. Moreover, the requirement for higher resolution is growing, and ultra HD (UHD) is proposed. For 4K UHD and 8K UHD, the data volume is 4 and 16 times of that for HD video, respectively.

To make the data fit into the capacity of network and storage devices, diverse effective compression algorithms are proposed and applied. They employ advanced algorithms to gain higher compression ratio, which results in high implementation cost as well. There is always a trade-off between compression ratio and implementation complexity. However, since the computational capability of processors keeps increasing, more complex compression algorithms can be mapped to the processor. The result is that data rate has been decreased significantly for the same quality. It also means that using the same data rate achieves much better quality.

There are mainly two organizations standardizing the video compressing process: ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG), who developed the H.26x series and MPEG series standards, respectively. Figure 3.1 shows the performance of some of those standards, and compression ratios are from [81]. From H.261, which only supports CIF (352x288) and QCIF (176x144), to the recently published H.265, which supports up to 8K UHD, and from MPEG-1, which could achieve compression ratio of around 26:1, to MPEG-H Part 2 (H.265), the compression ratio of which could be
around 400:1 [81], many video coding standards can be selected for different application scenarios taking both the required performance and implementation cost into consideration. There are also other popular video coding formats like RealVideo, WMV, VC series, VP series, and so on. Various video formats exist simultaneously on the Internet. To process the networked media, a flexible architecture with supporting multi-standard multimedia processing is required. If ASICs are implemented for decoding multi-standard multimedia, the decoder for each standard should be integrated together [82, 83], which increases the implementation cost and hinders flexibility.

In this chapter, a dual-core ASIP with dedicated video processing accelerators is proposed and implemented to support multi-standard video decoding for network applications. High performance is guaranteed by the dedicated video processing accelerators on one hand, and the dual-core processor supplies the flexibility on the other hand. Working at full speed, the ASIP is capable of decoding real-time 1280x720@25fps MPEG-2/MPEG-4/RealVideo video streams at 216 MHz with maximum power consumption of 414 mW, which can be lowered to 95 mW at working frequency of 27 MHz for 352x288@25fps video decoding.

### 3.2 Design methodology

The design flow is shown in Figure 3.2, and the detailed description is as follows:
• Application analysis: multimedia streams from Internet need to be processed. There are many different formats for both audio and video. In this case, MPEG-2/MPEG-4/RealVideo formats for the HD video contents are the targets.

• Function definition: the major functions for this application are network protocol processing, media data depacketization, multi-standard audio decoding and multi-standard video decoding.

• HW/SW partition: to make the architecture flexible, we use software to implement most of the required features, except the ones with extremely high performance demand. Hardware accelerators are needed for variable length decoding, inverse quantization, inverse transformation, motion compensation and deblocking functions used for video decoding as shown in Figure 3.3.
• HW/SW interface design: dedicated instructions are designed for the embedded processors controlling the video accelerators.

• HW/SW implementation: the related software and hardware are implemented in parallel.

• HW/SW co-verification: during this phase, the whole system is verified for the application, and the maximum performance is tested. If the performance is not adequate for the HD video decoding, adjustment should be made between the software and hardware to gain the required performance.

3.3 Video coding structure

To design the suitable accelerators, video coding algorithms are studied at first. Most of the video coding standards employ block-based algorithms, which utilize certain-sized pixel blocks as the basic coding units. For example, the 16x16 pixels blocks are commonly used in most of the standards, except the latest HEVC/H.265 standard, where blocks are as large as 64x64 pixels. The coding structures of most of the standards are similar as shown in Figure 3.4, and the differences are the detailed implementation algorithms.

In the general structure of video compression, prediction is the most efficient technique to decrease data volume. Instead of sending the raw data, only the residuals between the raw data and predictions are put into the final stream. The prediction can be performed on the neighbor pixels in the same frame, which is
known as intra-frame prediction or spatial prediction. Considering similarities between the consecutive frames in video sequence, the prediction could also be made by referring to previous coded frames, which is inter-frame prediction or temporal prediction. Furthermore, to get more accurate values for inter-frame prediction, motion estimation is calculated between the coded frame and the current frame to find motion vectors which indicate how the blocks move from the previous frame to the current one. The motion vectors will help to achieve accurate prediction in the coded frames. To ensure that the decoder gets the same data as those in the encoder end, a decoder is included in the encoder structure as shown in Figure 3.4 with dashed lines, and only the decoded frames are used as the references for prediction.

The residuals after prediction are transformed from spatial to frequency domain to ease the further data compression in the quantization phase. The high frequency part will be quantized with a large step size because people are not sensitive to high frequency data. Finally, the quantized coefficients together with the motion vectors are coded using entropy coding and then formatted to compose the compressed stream.
3.4 Accelerator-enriched architecture

To implement applications with the requirement of high performance and energy efficiency, an accelerator-enriched architecture is proposed and shown in Figure 3.5. The CPU group is responsible for high-level task parsing and processing. The application-specific function units (FU) together with the tightly coupled data memories (DM) provide high performance and energy efficiency for low-level data processing. They are controlled by the embedded processor (DSP or CPU) through the application-specific instructions, and can work in parallel to maximize the utilization. The deployment of instruction memory (IM) decouples the embedded processor from FUs, and enables both parts to run independently.

To avoid global moving of data, neighbor FUs share a DM in between, and an FU can access both its neighbor DMs as shown in Figure 3.6(a). Furthermore, any two of the DMs can exchange local data managed by the DMA controller (DMAC) as shown in Figure 3.6(b). If data transfer with external memories is required, DMA operations will be issued by the DMAC for fast data accessing as shown in Figure 3.6(c).

3.5 Design instance

Based on the proposed architecture and analysis of the application, a design instance for networked multimedia processing is shown in Figure 3.7.
One of the RISC processors, i.e. HRISC is utilized to run the operating system, manage network devices, configure system environment, process network protocols, depacketize multimedia data, etc. The other RISC processor, i.e. MRISC mainly works on the multimedia processing. Multi-standard audio decoding and low-definition multi-standard video decoding will be implemented on the MRISC. For high-definition video decoding, the hardware video decoding accelerators will be activated. In that case, MRISC analyzes the video stream, collects the necessary parameters and assigns tasks to corresponding accelerators by writing customized instructions to the CMD cache.

The video enhancement unit (VEU) consists of VLD, IQ, IT, MC and DF modules, which are derived from the general decoder model in last section and responsible for high-definition video decoding. The VEU is designed by adding the individual specific features of MPEG-2/MPEG-4/RealVideo to the common structure. The VEU parses instructions from the CMD cache and performs operations on the data stored in the corresponding buffers. These instructions are different from the normal instructions which are generally simple arithmetic, logic, or data moving operations. For each module in VEU, one instruction includes the required parameters and operations for the whole coding block. When starting processing, each module will perform the corresponding operations for one coding block and write the results to proper memory space without requiring the processors to be involved.

In order to make the proposed architecture capable of decoding high-definition videos featured with huge data volume, an efficient data communication structure is necessary. In the proposed architecture, internal buffers are placed between the VEU modules to localize the video data and accelerate the computation. The data transferring between internal buffers and external memories is managed by the DMA module. Due to the deployment of internal buffers, the modules in VEU are able to work in parallel in a pipelined way at the block level as shown in Figure 3.8, and at most five blocks can be processed simultaneously by the VEU.
Flexibility

The flexibility of the proposed architecture is three-fold:

- Dual-core processor: when high performance is not necessary, the VEU can be turned off, and two embedded RISC processors are fully programmable for various multimedia applications.

- Activity of VEU modules: the number of active modules in VEU can be adjusted for the specific performance requirement. The unnecessary modules can be turned off to save power.

- Working frequency of each module: working frequency of modules in VEU together with two embedded processors can be tuned according to the specific applications to provide adequate performance while keeping power consumption at a low level.

3.6 Implementation results

The ASIP is implemented using 0.13 μm CMOS technology with gate count of around 5 million, and the die size is 6.4 mm x 6.4 mm which is shown in Figure 3.9 together with the power consumption. The maximum working frequency is 216 MHz, when 1280x720 @ 25 fps MPEG-2/MPEG-4/RealVideo streams will be processed with power consumption of 414 mW. Only 13% is consumed by the VEU, which shows the efficiency of hardware accelerators. For low resolution videos, such as CIF-sized pictures with resolution of 352x288, the working frequency can be lowered to 27 MHz, when the power consumption is only 95 mW. The test system and application scenario are demonstrated in Figure 3.10.
3.7 Summary

To implement networked multi-standard multimedia processing applications, a dual-core ASIP with high performance and high flexibility is proposed and implemented. Highly specific accelerators provide the necessary performance. The generality of accelerators enables the adaptability for other similar applications. In addition, high flexibility is provided by the embedded processors. The balance between performance-to-power ratio and flexibility is made by software/hardware co-design.
Memory organization and data transmission structure are also key issues for applications with high throughput requirement. In this architecture, distributed internal buffers are deployed to increase the processing throughput and DMA is implemented to speed up the data transmission between internal buffers and external memories. The buses connect memories and functional units, and the communication throughput is adequate for this application. But for applications with heavy communication load, a more efficient communication infrastructure will be explored in the next chapter.
Chapter 4

Multi-processor architecture exploration for multi-view video decoding

For applications with even higher performance requirement, multiple ASIPs can be connected to increase the performance while providing sufficient flexibility. This chapter explores multi-processor architectures for multi-view video processing including design space exploration and implementation of communication infrastructure.

4.1 Background

3D video applications are emerging into our daily life, such as 3DTV, 3D movie, 3D football match, etc. To present 3D video, the pictures from two or more views should be recorded, coded and transmitted. Compared with single-view video, data volume and throughput are multiplied by many times. Because similarities exist between pictures from different views, many algorithms are proposed to compress the multi-view video efficiently by making use of the correlations between different views [85]. As a result, performance of decoders needs to be improved by many times compared with that required for single-view video decoding. There are many challenges not only for processing units but also for communication architectures. Some sophisticated ASICs have already been reported for decoding MVC video in [21, 86, 87]. High speed, high processing throughput circuits could provide adequate performance for high-definition multi-view video decoding. However, the design complexity is also considerable, and flexibility and scalability are limited.

By utilizing multiple processors, the applications can be implemented in another way by exploring the parallelism and employing multi-processor architectures. We use this approach which has several advantages:
• Low design complexity: the design problems are simplified as: choosing a single-ASIP with proper performance, selecting the necessary number of ASIPs and connecting those processors with an efficient communication structure.

• High flexibility: these processors can be re-programmed to adapt to similar applications.

• High scalability: the on-demand performance can be provided for a particular application by adjusting the number of processors.

Multi-view video coding (MVC)

As Annex H of H.264 [32], MVC inherits all the sophisticated coding algorithms in H.264 for single-view video coding. Besides, inter-view prediction is utilized as a very efficient technique to obtain large compression ratio [88]. For single-view video coding, the inter-frame prediction uses temporal neighbor frames as the reference to predict the data in the current frame, which greatly increases the coding efficiency compared with the techniques for still image encoding. While in MVC, not only temporal neighbor frames but also spatial adjacent frames, i.e. frames from neighbor views, are utilized to predict the current frame. The merit is that coding efficiency is significantly increased. The drawback is the complicated coding structures which will be obstacles for parallel processing.

A typical prediction structure in one group of pictures (GOP) is shown in Figure 4.1 for eight views, and totally 64 pictures exist in one GOP. In a base view, encoding or decoding of the pictures will not rely on pictures from other views. The single-view decoder can only process the data in a base view, which keeps the bit stream compatible for normal single-view decoders. For anchor pictures, the temporal prediction will not be applied, and only the inter-view prediction can be utilized. The purpose is to prevent error propagation between GOPs and to support random access among GOPs. For the other pictures, both the temporal inter-frame prediction and spatial inter-view prediction are employed to reduce data redundancy. As a result, MVC streams are characterized by larger number of pictures to be processed and complex dependences among the pictures. In order to handle MVC streams, multiple times of data processing throughput are required compared with that for single-view streams.

Network on Chip (NoC)

To connect multiple ASIPs, an efficient data communication infrastructure should be designed to provide sufficient throughput. In [50], multiple ASIPs are connected via a bus structure together with a shared memory for data exchanging, as well as a master core for managing the work flow and collecting the results. Burst mode and DMA operations are provided in the bus system for data transmission to reduce the possible contention. However, for applications with heavy data load,
Figure 4.1: Typical prediction structure; one square represents one picture, and arrows show the reference relations.

Figure 4.2: An example of 4x4 2D mesh NoC
the non-sharable and non-scalable bus structure becomes the bottleneck in a multi-
ASIP system. By connecting the ASIPs with an on-chip network, the network-
on-chip (NoC) \[89\] can supply high communication bandwidth, high efficiency and scalability.

NoC architectures decouple the communication subsystem from the computa-
tion subsystem. Data communication is performed in a dedicated on-chip network, and the computation part needs only to interact with the network interface for data exchanging. Design problems related to the communication subsystem include the selection of topology, task assignment scheme, switching modes, flow control, routing algorithms, and so on \[90\]. In this work, we choose 2D-mesh topology considering the intrinsic modularity and scalability. An example of 4x4 2D-mesh networks is shown in Figure 4.2. Besides, to implement NoCs for specific applications, design decisions for the computation part, such as choosing the processing capability of each node, selecting suitable number of nodes, etc. need also to be taken into consideration. The computation capability and communication capacity should be balanced to achieve an optimized overall performance for the whole system.

4.2 Design space exploration

There are two strategies for implementing multimedia applications on NoC archi-
tectures. One is partitioning the whole process into several phases, implementing
different phases on different NoC nodes and pipelining those phases to gain performance improvement as shown in Figure 4.3(a). The merit is the simple traffic pat-
tern and determinate data dependency between consecutive nodes in the network, while the drawback is the limited scalability constrained by the partitioned processing phases. Examples with this mapping strategy can be found in \[91, 92\]. Another
Figure 4.4: Simplified System (a) the whole system, (b) one unit, (c) work schedule with inadequate link bandwidth, (d) work schedule with excessive link bandwidth; $S_d$ for the size of tasks, and its dispatching time $t_d$; $S_c$ for the size of results, and its collecting time $t_c$; $t_p$ is the processing time. (Adapted from [93])

implementation strategy is based on parallel tile processing shown in Figure 4.3(b). Each tile is capable of dealing with the whole process, and the performance is increased by enabling concurrent tile processing. The scalability is extended at the cost of complex traffic patterns and possible large volume of data transferred among the tiles. Emphasizing scalability and flexibility, the second strategy is utilized for the following exploration.

Exploration for a simplified system

A theoretical analysis is firstly performed to formulate the exploration work flow. To ease the theoretical analysis, a simplified system as shown in Figure 4.4(a) is studied to explore the design space. The data dependency between processing tiles is eliminated, which means the processing tiles will get all the necessary data from the task distribution node and send the results only to the converging node. Assuming that the task size ($S_d$) is larger than the result size ($S_c$), the relation
of system processing throughput (Θ_{sp}), single-node processing throughput (θ_{p}), physical link bandwidth (BW_{link}) and number of processing nodes (N_p) has been theoretically analyzed in [93]. For the case of $S_d$ smaller than $S_c$, a similar analysis can be applied. Combining with the simulation, for a given target system processing throughput (Θ_{sp}), optimized configurations of θ_{p}, BW_{link} and N_p can be explored through the procedure as shown in Figure 4.5.

1. By performing a theoretical analysis as that in [93], the reference value of link bandwidth (BW_{linkγ}) is calculated from the given value of target system processing throughput.

2. For each network with certain size (N_p), based on BW_{linkγ} and the relation of BW_{link} versus θ_{p} and N_p, the minimum required value of θ_{p} is acquired.

Figure 4.5: Work flow for the system exploration
3. For each $N_p$ and $\theta_p$, sweep $BW_{\text{link}}$ whose value is around $BW_{\text{link}^*}$ and evaluate them in the simulation platform.

4. If the simulation result of system processing throughput is larger than the given value, stop sweeping $BW_{\text{link}}$, change the size of network and goto 2 to find another group of $BW_{\text{link}}$, $\theta_p$ and $N_p$.

**Exploration for MVC implementation**

Based on the aforementioned analysis, the NoC system for MVC decoding is explored accordingly, and the target is to find optimized configurations of the system to implement full HD (1920x1080) MVC decoding at 30 fps (frames per second).

Figure 4.6 shows the hierarchical structure of MVC streams. The basic processing unit is the macroblock (MB), which consists of 16x16 pixels. Neighbor MBs or the MBs in neighbor frames may be utilized to generate the current MB. One picture is composed of a number of MBs. If only spatial prediction is employed, the picture is an I frame. If only forward pictures in the time domain can be used as the reference to predict the current picture, the picture is a P frame. If both forward and backward pictures in the time domain are selectable as reference pictures, the current picture is a B frame. Thus, at the picture level, to decode an I frame, no related data are required from other frames, while to decode P and B frames, data from other frames are necessary. In MVC, pictures further constitute views as introduced in the previous section. Decoding streams from one view may need to refer to data from other views. On top of views, GOP processing is independent from each other, and can be executed in parallel.

Task assignment at different levels of MVC will generate diverse traffic patterns, and result in varied configurations of the system. If the task assignment is at the MB level, frequent data transferring leads to performance degradation. On the other hand, GOP-level task assignment needs large storage space in each node and will cause large processing delay as well. So the picture-level and view-level task assignments are studied for implementing the MVC in the following sections.
Picture-level task assignment

Data dependences exist not only between pictures of the same view, but also between pictures in different views. However, by analyzing the dependence of pictures in one GOP, the pictures can be subdivided into several sets, and in each set the pictures are independent of each other [95]. The pictures with larger set index may need the pictures with smaller index as the reference as shown in Figure 4.7. To process each set, the dispatching node sends all the necessary data to a processing node for single picture decoding. Results are collected by the converging node. Only after all the pictures with smaller set index are processed, the task for processing the pictures with larger set index can be issued. A shared memory is deployed between the dispatching and converging nodes. A possible data flow for picture-level task assignment is depicted in Figure 4.8.

Picture-level task assignment in this work only issues tasks of decoding pictures in the same set to processing nodes, and transferring data between processing nodes is avoided. In this case, the exploration algorithm proposed for the simplified system can be directly utilized for the analysis. After the exploration in [94], possible configurations of the system are summarized in Figure 4.9. Both 3x3 and 4x4 mesh NoCs can be selected depending on the physical link bandwidth and single-node performance. With 3x3 networks, the minimum performance requirement for single node is the capability of processing full HD single-view stream at 50 fps, correspond-
Figure 4.8: A possible data flow of core (1,3) for the picture-level parallel processing (Adapted from [94])

Figure 4.9: Design options for picture-level task assignment; $\Gamma$ for network size, $\Phi_p$ for single-node performance and $B_L$ for link bandwidth (Adapted from [94])

...
decoding at 30 fps, to support eight-view MVC decoding, only 33% performance improvement is required.

View-level task assignment

The view-level task assignment scheme issues single-view decoding tasks to each processing node. Considering the dependence between views, data exchanging among the processing nodes is necessary. Figure 4.10 shows an example of the data flow for decoding view 2 on node (1,3). The task dispatching node sends coded streams to node (1,3) for decoding, and meanwhile, data from node (1,0) for view 0 are also sent to node (1,3) as the reference. Furthermore, decoded streams need to be sent to the converging node for result collection and to node (0,1) and (3,1) as the reference for view 1 and view 3.

View-level task assignment scheme complicates the traffic pattern in the network. However, the exploration algorithm for the simplified system can still be utilized as a preliminary analysis. After that, by tuning the single-node performance and physical link bandwidth, optimized configurations of the NoC system can be obtained from the simulation platform to meet the performance requirement.

Feasible configurations of the system using view-level task assignment scheme are shown in Figure 4.11. For eight-view MVC decoding, if only one GOP can be processed at a time, at most eight processing nodes are needed, which is limited by the view number. In this case, increasing the network size will not help to increase the system performance. This explains the fact that options 1 and 3, as well as options 2 and 4, have the same requirement for single-node performance and
link bandwidth even though the network size is different. To utilize more processing nodes, some GOPs should be processed in parallel. Option 7 shows the case of three GOPs being processed simultaneously with a 5x5 network, when the requirement for single-node performance is only 20 fps for single-view streams processing and the link bandwidth is 9.6 Gbps. However, since introducing the GOP-level parallelism will increase the implementation cost and processing delay, the maximum number of GOP being processed in parallel should be constrained to a small value.

4.3 Communication network

Besides network topology, the selection of switching techniques also has a large effect on both the performance and the implementation cost. Circuit switching (CS) and wormhole switching (WS) are two commonly used techniques for on-chip networks. Circuit switching will reserve the whole end-to-end path before starting to send data, and the data transmission uses a dedicated link without sharing it with other transmissions. The advantages are the lightweight architecture and small transmission delay. The disadvantages are the large link setup delay and lower network utilization. In contrast, a data transmission is divided to many flits in wormhole switching, and starts immediately once there is any free buffer in the downstream node. A header flit contains the routing information, and body flits only need to follow the header flit to reach the destination node. To increase utilization of the network, virtual channels (VC) can be deployed to share the physical channels. Different data transmissions can use different virtual channels but share one physical channel. The merits are the zero setup delay and higher network utilization. The drawback is the uncertain transmission delay and relatively high implementation cost.
Circuit switching is suitable for applications with large infrequent data transmission, otherwise, wormhole switching should be used [96]. For a specific application scenario, the decision of choosing the suitable switching technique still need to be made by carrying out the evaluation on performance and cost. In this case, the analysis is performed for MVC decoding in [97].

The simulation router models of CS and WS used for evaluating the performance and cost are shown in Figure 4.12. Five pairs of in/out ports connect four neighbor routers with the local processor. In the CS router, any one of the output ports can only be linked to one input port during one transmission. For the WS router, a number of VCs share one input port, and one output port can be associated with several VCs during data transferring.

To find the suitable switching technique for MVC decoding, the optimized configuration of WS on link bandwidth, buffer depth, number of VCs is firstly explored. Then, the optimized structure of WS is compared with CS on the performance of decoding speed, network utilization and delay for decoding MVC streams with the same topology of NoCs. The results indicate that employing CS and WS achieves similar performance for MVC decoding. Considering the simpler architecture which will result in lower implementation cost, CS should be used when implementing MVC decoding with NoC architectures.
4.4 Summary

By connecting multiple ASIPs in a NoC manner, the system provides high performance together with scalability and flexibility. To implement MVC decoding, if single ASIP is utilized, performance is to be improved by several times which are determined by the number of views supported. While with multi-processor NoC architectures, the required performance improvement can be much smaller both with picture-level and view-level task assignment schemes. Meanwhile, to gain the necessary performance for the whole system, design options including single-node performance, network size, link bandwidth and task assignment schemes are balanced. If the single-node performance is not enough, it can be compensated by choosing a suitable task assignment scheme, network size and link bandwidth, what is also applicable when the network size or link bandwidth is inadequate. For the communication structure, circuit switching instead of wormhole switching is appropriate for MVC decoding considering their similar performance and lightweight architecture of CS.
Chapter 5

Customizable tile-ASIP implementation for Internet of Things

In this chapter, we focus on the implementation of a low power customizable ASIP for Internet of Things (IoT). In the vision of IoT, everything will be connected and can talk to each other [98, 99]. The ubiquitous sensing, processing and communication should be seamlessly performed by the attached devices, which will demand hardware architectures with characteristics like small-sized memory requirement, flexibility and adaptability to various devices with diverse performance demands, low power consumption but sufficient computational performance for functions such as digital signal processing, communication protocol processing, encryption, etc.

5.1 Background

Modular design

In IoT applications, the demanded performances for processors are greatly varied with the devices from sensor nodes to servers and also varied with the vast number of application scenarios. The straightforward solution is to design and implement specific processors for each application scenario. However, the design, verification and mask costs are extremely high and become the obstacles for this solution [100]. A modular design, instead, utilizes mature extensible modules to compose a specific system for the target performance and cost. Suitable number of modules and proper connections can be selected after analyzing the application. The implementation costs are greatly decreased. The challenge, as a result, is the implementation of such extensible and efficient modules. The hardware-level conjunction of the system should be transparent to the high-level programming, and the performance and power efficiency should also be close to that of a single processor implementation.
The concept of “Lego Chip” is proposed in [100]. In this concept, a system including two or more chips behaves like a whole single SoC. Diverse but mature Lego Chips are seamlessly connected via the highly efficient Lego Interconnects to achieve the target performance. The engineering cost can be dramatically decreased by using that technique. XMOS architecture and its implementation are introduced in [101]. Multiple processors can be connected with point-to-point communication links, and it allows designers to build the system either on multiprocessor chip, in packages, on boards or as a distributed system. A number of threads can be directly mapped to the hardware resources in each core, and controlled by a hardware scheduler. The event-driven threads are able to reduce power when waiting for events. For constructing the multi-core system, each core has its global address and can communicate with the others through control and data packets. Networks-in-package is implemented in [102] by connecting four NoC chips with off-chip links in one package to demonstrate the scalability and modularity of the system. For each NoC chip, heterogeneous function units are integrated together with the packet-switched communication infrastructure. If higher performance is required, suitable number of NoC dies can be clustered to construct the networks-in-package. It alleviates the problems of super large scale circuits such as low yield and high costs.

**Processor architecture**

When the computer was introduced at the beginning, the memory size was limited, and the compiler and interpreter were very simple. Many instructions which could perform various complex functions were designed together with the necessary hardware units. Those instructions constitute the basic instruction set architecture (ISA), based on which applications could be implemented. In order to make the memory requirement smaller and make the programming easier, abundant instructions were included in the ISA for various functionalities. In this case, due to the complex operations implemented in different instructions, multiple cycles are needed to execute one instruction and the execution time for different instructions varies, which makes the execution of instructions not easy to be pipelined.

Compared with the aforementioned design methodology, which is called complex instruction set computer (CISC), reduced instruction set computer (RISC) simplifies the functionality of instructions. Most of the instructions are finished in one single cycle. The hardware design is simplified. Complex functions are implemented by combining the simple instructions in a suitable way, which is done by a compiler. The memory requirement for storing executable codes is thus larger and the code density is lower in RISC [103].

With the rapid development of semiconductor technology, the speed and number of transistors are greatly increased for single chips. The memory space for storing codes is not a problem any more. By pipelining the execution of instructions, the speed of RISCs can be very fast due to the simple hardware architecture. Caches are deployed to fill the gap between the extremely fast executing core logic and the
relatively low speed memories. Owing to the simple hardware architecture and fast execution speed, RISCs become popular for embedded applications.

For the IoT applications in this case, lightweight architectures with low power consumption and high energy efficiency are highly demanded. Memories occupy large proportion of the power consumption [104]. Small memories should be used considering both the area and power consumption. The utilization of hardware units is supposed to be explored as much as possible. Meanwhile, due to the varieties of application scenarios in IoT, reconfigurability and scalability are also of importance for IoT processors.

In this work, a reconfigurable and scalable control-centric processor with an integrated multi-mode router is proposed and implemented as one tile for the possible modular extension. The router supports circuit and wormhole switching simultaneously to fit into the different traffic patterns for various application scenarios. The techniques that can ease modular extension also apply to the multi-mode router. Aiming at high energy efficiency, the processor simplifies control logics, and basic functional units are directly managed by the sequence mapping table (SMT) saved in internal storage of the processor. The decoding hardware is thus avoided. By designing the corresponding SMT, different ISAs can be implemented to the processor. Code density is increased through optimizing the SMT for specific applications. Dedicated SMTs can even be mapped to the application directly once extremely high energy efficiency is demanded.

5.2 Multi-mode router

NoC routers that utilize both circuit and wormhole switching to improve communication capacity are popular in many studies [105, 106, 107, 108, 109, 110, 111, 112]. The main purpose of those studies is to combine the merits of two switching techniques to increase the transmission speed and lower the latency, but not to fit into the diverse traffic patterns by adaptively selecting the proper switching technique for particular application scenarios, which is emphasized in this work for the modular design.

The proposed router architecture is shown in Figure 5.1. Circuit switching (CS) and wormhole switching (WS) utilize the common physical channels and control logics. The use of a specific switching mode is determined by the header sent from the source node in the network. A header in CS will setup an end-to-end link between the source and destination nodes, while a header in WS allocates one available virtual channel (VC) for the following data packets. A maximum of four virtual channels can share one physical channel if it is not occupied for circuit-switched messages. A circuit-switched message will preempt the physical channel being used for WS. As a result, each physical channel supports dynamical and free switch between CS and WS for different application scenarios.
Necessity

As described in the last chapter, CS is suitable for large infrequent data transmission, and applying WS is appropriate in other cases. However, “large” or “small” is relative to the hop count number and network status when performing the data transmission. Even if the data are “large” for one transmission, they might be “small” for another transmission.

Figure 5.2 gives the simulation results of maximum throughputs with only CS or WS mode activated for a 10x10 mesh NoC when the maximum hop count of each transmission is restricted to 2, 4, and 6. When the hop count number is increased, the performance degradation of CS is much larger than that of WS. It means that CS is more sensitive to the distance of data transmission. For the message size with label “P1” in Figure 5.2, if the maximum hop count number is 2, employing CS is suitable. However, when increasing the maximum hop count to 4 and 6, utilizing WS is appropriate. This shows the importance of the capability for dynamically adjusting the switching mode to actual traffic patterns.

In some application scenarios, “large” and “small” data transmissions need to be handled at the same time. In that case, using only one switching technique will restrict the performance of the network. The previous 10x10 mesh NoC is still used as an example, and a maximum hop count number of 4 is applied. Assume that there are two types of data messages in one application at the same time, 10-flit-sized messages (message size of 320 bits in Figure 5.2) as “small” data and 100-flit-sized messages (message size of 3200 bits in Figure 5.2) as “large” data, and they...
Figure 5.2: The maximum throughput with only CS mode (CSM) or WS mode (WSM) activated; $H\ n$ means that the maximum hop count is $n$; S1 and S2 are the message sizes corresponding to the throughput cross points of the CSM and WSM for H2 and H4, respectively (Adapted from [113]).

Figure 5.3: The comparison of throughput and latency for 10-flit messages and 100-flit messages with only CSM, only WSM and mixed CSM/WSM (Adapted from [113]).
are generated with the same probability. Three possible strategies can be utilized for this application: CS for all messages, WS for all messages, and mixed CS/WS adaptively to message sizes. The simulation results for throughputs and latencies are shown in Figure 5.3. The mixed mode provides both higher performance and lower latency.

The previous two situations show the necessity of supporting CS and WS simultaneously in one router when various traffic patterns may be present on the network.

**Implementation**

The proposed router is implemented with parameterized buffer depth of VCs and link width of physical channels. The areas for routers with buffer depth of 8, 16, 32 and link width from 8 to 64 bits are given in Figure 5.4 after being synthesized in 65 nm low leakage CMOS process. The areas are almost doubled when the buffers or physical links are duplicated. The corresponding throughput and power efficiency are shown in Figure 5.5, under the conditions of 10x10 network, 40-flit-sized messages, maximum hop count of 4, and CS for hop count not larger than 2 otherwise using WS. Increasing the buffer depth will improve performance, but decrease power efficiency. Because of that, buffer depth of 8 is selected in this work. Nevertheless, increasing the link width can both increase the performance and power efficiency.

As a proof of concept, one tile including processors and the proposed router with buffer depth of 8 and link width of 8 bits, has been fabricated in 65 nm low leakage CMOS process. Figure 5.6 depicts the complete structure of the implemented
5.3 Control-centric processor

The proposed architecture of the customizable ASIP is shown in Figure 5.7. A dual-core reconfigurable control-centric processor is integrated together with the
multi-mode router, internal memories, power management unit, clock generation unit, debug interfaces, and other peripherals.

Control-centric architecture

In each core (CORE_L or CORE_R), functional units and data paths are directly manipulated by a control word (CW) without internal interpretation. The result is simplified control logics and the capability of parallel execution of all functional units. The structure of one CW is shown in Figure 5.8. One CW is partitioned into several sections, each of which contains the necessary control bits for corresponding functional units or data paths. CWs defining specific functions constitute SMTs, which are saved either in CROM for the fixed configurations of the processor or in CRAM to provide reconfigurability.

SMTs can be customized to support general-purpose ISAs or application-specific instructions, and even directly implement a particular application, as shown in Figure 5.9 for application mapping schemes. The comparison of energy efficiency and complexity for those schemes are also presented in that figure. The implementations with general-purpose ISAs gain low design complexity, while dedicated SMTs for specific applications can achieve higher energy efficiency.

When the processor is used to execute instructions, the fetching (IF), decoding (ID) and executing (EX) processes are all carried out by the SMT with the same
Figure 5.8: Structure of one control word

Figure 5.9: Mapping from applications to the processor (Adapted from [114])

Figure 5.10: Instructions processing in the proposed processor compared with other data-centric processors

hardware units, compared with data-centric processors which employ dedicated hardware units for instruction fetching (IF), decoding (ID) and executing (EX), respectively, as described in Figure 5.10. For data-centric processors, pipelines are
normally utilized to increase the processing throughput. Extra hardware resources are required to support speculative execution of the instructions, since the successive instruction starts to be processed before the processing of current one is finished. In contrast, a control-centric processor changes functions of the same hardware units with different configurations to perform the corresponding operations. Hardware can be utilized to a high extent to achieve high energy efficiency.

**Low power considerations**

The low power design considerations are listed below:

- **Reconfigurable architecture**: the reprogrammable SMT supports both general-purpose and application-specific instructions. Therefore, dense software programs will be generated. The results are the small memory requirement and low access frequency to the memory, which can both lower the power consumption.

- **On-chip memories**: the on-chip ROM and RAM are used both for storing SMTs and application programs. For many IoT applications, on-chip memories are enough for the programs, and the power consumptions on the pins for connecting the external memories can be saved.

- **Power management**: different working modes are designed to provide proper performance for adapting to different scenarios. The activity and working frequency of each core and peripherals can be dynamically adjusted following the change of work load.

**High performance**

Two cores are equipped with the same kinds of computational units. Besides the general-purpose ALU, a multiplication and accumulate unit (MAC) together with a shifter is also integrated for each single core to increase the performance for digital signal processing. Moreover, all the function units can be activated simultaneously and work in parallel, which is controlled by the SMT. By organizing function units in a suitable way, hardware resources are utilized efficiently and high performance can be achieved.

For each single core, to make the critical path short and guarantee sufficient execution time for each control word, one execution cycle consists of two clock cycles. The address of the SMT memory is given in the first cycle, while the data is available in the next cycle, as shown in Figure 5.11(a). By connecting two cores to the same memories, and letting them access memories alternatively, the memories are utilized efficiently and the performance is doubled without the need of arbiters between two cores, as shown in Figure 5.11(b).
Flexibility

For applications to be implemented in the proposed architecture, the hierarchical flexible structure is shown in Figure 5.12. The application layer flexibility is realized by providing a certain number of predefined common functions to support rapid mapping from applications to the processor. Besides, general-purpose programming is applicable in layer 3 and 4 by using high-level languages or assembly codes. For some special applications with critical timing or power requirement, the reprogrammability in layer 2, which is the SMT coding layer, can be exploited. As the flexibility in layer 1, i.e. the hardware layer, the number of active cores is adjustable for different applications. If the performance of two cores is not sufficient, more cores can work together by being connected through the integrated multi-mode router to construct a network.

The memory structure is also flexible. For applications with small memory re-
requirement, internal memories are enough. If more memories are required, external SDRAM can be connected to the processor. The internal and external memories utilize a uniform memory space, and it is transparent to higher levels programming. Besides, the pins for SDRAM interfaces are shared with general-purpose I/O pins, and functions of those pins are configurable depending on the connectivity of external memories.

### 5.4 Tile implementation

The customizable tile-ASIP including dual cores and the multi-mode router has been fabricated in 65 nm low leakage CMOS process. The die photo and gate count breakdown are presented in Figure 5.13. Most of the gates are used by the memories. Adding the second core (CORE_R) contributes only 0.77% to the gates, but doubles the performance of the processor. The maximum working frequency is 350 MHz with power consumption of 22 mW when all the hardware resources are activated. If one core is turned off, the power consumption is decreased to 15 mW. In order to keep the status of the processor when all clock sources are shut off, 0.74
Figure 5.14: Power consumption under different working modes and conditions

mW is needed. The power consumptions for different configurations are shown in Figure 5.14. Based on this figure, appropriate working strategy can be selected to achieve the on-demand performance while keeping the power consumption as low as possible.

With the proposed ASIPs, on-board networks with packaged chips and in-package networks with bare dies are feasible to further increase the performance of the whole system. Such configurations are demonstrated in Figure 5.15. A maximum of 256 tiles can be connected with single link bandwidth up to 1.4 Gbps.

5.5 Summary

This chapter discussed the design and implementation of a customizable ASIP for low power applications, which demand high energy efficiency. To achieve that, a multi-mode router and a control-centric processor are integrated to provide the on-demand performance. By utilizing the multi-mode router, diverse inter-silicon networks can be constructed to meet the performance requirement for specific applications. With the control-centric processor, control logics are simplified, and hardware resources are directly operated by the control words of SMT inside the processor. By properly orchestrating the SMTs, both general-purpose ISAs and application-specific instructions can be mapped to the processor. Furthermore, customized SMT can dedicate all resources of the processor to particular applications if required. Finally, the tile-ASIP is fabricated in 65 nm low leakage CMOS process, and different working strategies are proposed for various application sce-
narios. Meanwhile, both on-board and in-package networks are demonstrated to show the feasibility of performance extension.
Chapter 6

Conclusions

6.1 Thesis summary

This dissertation explores the design and implementation of energy-efficient ASIPs for ubiquitous sensing and computing in both high-throughput compute-intensive processing of multimedia and low-throughput low-cost processing of IoT scenarios. For high-throughput processing of networked multi-standard multimedia, single-ASIPs integrated with general-purpose processors and application-specific accelerators are proposed and implemented. Control-intensive tasks such as protocol processing, depacketizing, stream parsing, etc. are more suitable to be processed by the embedded processor, while the dedicated accelerators are efficient for the computation-intensive functions. The instruction-controlled accelerators and general-purpose processor also provide the flexibility to support both the existing and future standards. The chip has been fabricated and a test system is built to handle networked multimedia streams.

Multiple ASIPs are connected with an on-chip network to gain even higher performance and flexibility for applications such as multi-view video decoding. The design problems including the selection of the number of ASIPs, required performance for a single ASIP, link bandwidth of the network and task assignment schemes, have been studied for high-definition eight-view MVC decoding. To use multi-ASIP architectures, parallel processing at the proper level has been explored. Both picture-level and view-level task assignment schemes are proposed, and the corresponding communication and computation subsystems are analyzed to obtain the design options for the whole system. The results show that a limitation of one factor can be compensated by the enhancement of the others under the condition that the minimum requirement for each factor is guaranteed.

A tile-ASIP is designed and implemented for applications with emphasis on low power consumption, high energy efficiency, and high adaptability. The integrated multi-mode router supports both circuit and wormhole switching simultaneously, and the two schemes can be dynamically selected to adapt to the actual traffic.
pattern. The embedded processor core utilizes control words to directly operate the functional units and avoid complex control logics so increasing the hardware utilization. The control words compose the sequence mapping table, which is stored in the on-chip memories. By creating the proper tables, general-purpose ISAs, application-specific instructions and mappings dedicated to particular applications can be implemented by the processor. Dual processor cores are deployed sharing both the control and data storage, which doubles the performance and further improves energy efficiency. Thanks to the integration of the multi-mode router, a suitable number of tile-ASIPs can be connected with the inter-silicon network to provide on-demand performance for diverse application scenarios.

To conclude, different strategies for designing ASIPs are utilized for diverse application scenarios in ubiquitous sensing and computing. Employing application-specific accelerators will effectively increase the performance with low power consumption, while they pose high design and verification cost together with the relatively low flexibility. They can be used in the ASIPs that only similar applications will be mapped to. An example is multi-standard multimedia processing. When the applications demand even higher performance, connecting the existing high performance ASIPs to construct the multi-ASIP architecture is a feasible solution instead of further increasing the single-ASIP performance. As a contrast, the tile-ASIP increases energy efficiency by improving hardware utilization. As a result of the reconfigurability of the processor and the scalability provided by the on-chip multi-mode router, the capability of the tile-ASIP providing on-demand performance lowers the cost and gains high energy efficiency.

6.2 Future work

One revolutionary approach to achieve high performance at extreme low-power consumption is the application of brain-inspired computing [4, 115, 116]. The human brain runs at very low power consumption, but performs many complex tasks that even the current computers cannot handle. Specialization is one of the most important features to achieve high energy efficiency [117]. Different parts of the brain are trained and customized to perform diverse functions. For the proposed tile-ASIP in this dissertation, the reconfigurable and customizable processor cores together with the integrated router make it a competitive solution for constructing a brain-inspired computer. Future research can be performed in the following three aspects to facilitate the implementation of brain-inspired computing with the tile-ASIP.

Hardware improvement

In the current version of the tile-ASIP, fixed configurations of the ASIP are saved in the internal ROM. To enhance reconfigurability and customizability, re-programmable non-volatile memories can be utilized to replace the ROM. Meanwhile, the SMTs
and instructions will share the internal RAM to maximize the flexibility instead of using independent storages like in the current version.

The integrated router is designed for mesh networks, and up to 256 nodes are supported. The reliability and adaptability can be increased by supporting irregular topology, adding fault detection and correction algorithms, etc. Energy-efficient inter-router connections could also be developed. Besides, for both the router and processor, event-driven working modes are necessary to further increase the energy efficiency by activating the resources only for the time period while incoming events are being recorded.

**Programming model**

The programming of single tile-ASIP including SMTs and application programs is supported. To ease the integration of hundreds or even thousands of tile-ASIPs, programming models for configuring and scheduling multiple tile-ASIPs need to be studied and formulated. Besides, since SMTs can be reprogrammed at real time to change the processor behavior, in-system optimization for specific tasks is desired with the support of customized software tool chain.

**Application mapping**

Brain-inspired computing has drawn much attention due to the extremely high energy efficiency of the human brain compared with traditional computers. The brain works with high specialization, low processing duty cycles and low-voltage operations [4]. One million programmable spiking neurons and 256 million configurable synapses are implemented in a single chip in [115], and many chips can be further tiled to support even more complex neural networks. 18 ARM968 processing cores are integrated as a chip multiprocessor (CMP) in [116] for simulating large-scale spiking neural networks. The architecture can scale up to 65536 chips enabling modeling up to 1 billion neurons.

Considering the customizable processor core, its tightly coupled internal memories, and the scalable communication structure of the proposed tile-ASIP, the mapping of brain-inspired computing can be studied as future work.
Bibliography


