MediaDSP: An Application Specific Heterogeneous Multiprocessor SoC

Di Wu, Per Karlström, Johan Eilert, Andreas Ehliar and Dake Liu
Department of Electrical Engineering, Linköping University, Linköping, SE 58183
Email: {diwu, perk, je, ehliar, dake}@isy.liu.se

***Abstract*— The emerging era of embedded computing demands both high computing power and flexibility to accommodate various applications. This paper presents a new heterogeneous multiprocessor SoC platform targeted at several application domains. The platform is scalable and can be tailored according to various scenarios.**

I. Introduction

DSP industry has seen the prominent evolution of mobile and multimedia applications. It is more and more difficult for single-core DSPs to accommodate new applications that demand huge computing power. Meanwhile, new multimedia standards are emerging with a number of legacy standards still existing, which makes it difficult for customized ASIC design to support. Thus the concept of multiprocessor SoC (MPSoC) for embedded computing has emerged as a new design paradigm. However, it also brings great challenges to designers to make good trade-off among a number of factors.

An application specific multiprocessor SoC namely MediaDSP has been proposed by us and the project is still ongoing. The goal of designing such a platform is to shrink the design space by targeting specific application domains and explore the optimal trade-off in heterogeneous MPSoC designs. The paper is organized in the following way: The first part briefly introduces targeted application domains. The second part elaborates the architecture of MediaDSP. Task partition and scheduling is addressed following this part. Then the toolchain and performance issues are introduced. The last part concludes the paper.

II. Application Domains

As the technology advances, a new era of consumer electronics has arrived. Smart gadgets such as mobile phone, MP3 players and GPS terminals have been ubiquitous in our daily life. It is believed that more and more multimedia functions will be integrated into one single terminal which brings a new challenge to multimedia embedded system design. As illustrated in Figure 1, multimedia application domain covers various applications ranging from audio to 3D. Limited by the size, mobile terminals can not afford the space required by a bunch of ASIC chips designed for different purposes. At the same time, a lot of applications share common computational features which make it possible for a single programmable HW to accommodate them.

A. Audio/Video Compression

MP3 players have already been integrated into mobile phones. IPTV and mobile TV services are ready to be released in several countries. However, there are still several barriers for the designers to cross such as computing power and heterogeneous standards support. Although the latest video compression technologies can obtain very high transmission efficiency, the computational complexity of codec also increases dramatically. Meanwhile, there are quite a few existing coding standards that are supported by different operators and content providers. Therefore, single chip solution that supports all existing standards is not only efficient but necessary.

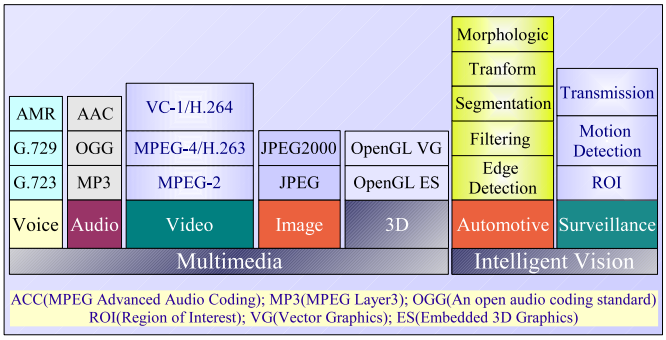

Fig. 1: Application Coverage of MediaDSP

Audio and video compression generally consists of several steps such as entropy coding, transform, estimation and other filtering operations. Entropy coding methods such as variable length coding (VLC) and arithmetic coding (AC) are used to reduce the entropy in the transmitted bit stream. The mechanism of transform is that since human eyes are sensitive only to the lower part of the spectrum of light, bandwidth efficiency can be achieved by converting spatial signal into the frequency domain and only transmitting those located in the lower part of the spectrum. Other filtering operations such as interpolation (usually involved in motion compensation for higher accuracy) and deblocking (to reduce artifacts introduced by intra prediction) are all intensive vector operations. Traditional scalar DSP can not supply enough computing power for these operations while it is difficult for ASIC to keep pace with the change of these algorithms. In order to utilize the parallelism that exists in audio and video coding, application profiling needs to be done and programmable hardware needs to be designed to supply enough performance.

B. 3D Gaming and GUI

N-Gage from Nokia will not be the only mobile phone that supports 3D gaming and attractive 3D GUI. Compared to traditional GPU which even consumes more power than CPU, 3D processor for mobile applications should meet strict constraints in power and cost. Thus how to accommodate 3D processing power into mobile terminal is also a challenge.

C. Intelligent Computer Vision

Intelligent computer vision is emerging in areas such as automotive and surveillance. Applications such as driving-assistant and anti-collision are playing important roles in next generation automotive industry. The surveillance industry is driven by the need of private and national security. Applications such as object detection, pattern recognition and IP based video transmission are evolving rapidly. Since most of these applications are hard real-time, hardware with high performance is needed to processing collected information in real-time. At the same time, since algorithms are evolving continuously, ASIC with fixed functionality is not flexible enough, which means programmability is important. Thus the optimal trade-off among factors such as performance, flexibility, cost and power consumption needs to be investigated.

III. Architecture

MediaDSP explores and utilizes four kinds of parallelism that exists in the above application domains such as instruction level parallelism (ILP), data level parallelism (DLP), memory level parallelism (MLP) and task level parallelism (TLP). As depicted in Figure 2, MediaDSP is a multiprocessor SoC consisting of a number of processor cores, memory banks and ASIC accelerators if necessary. All cores are connected to a crossbar on-chip connection network (OCN) which is controller by a controller. By configuring the OCN, these cores can be organized into different combinations targeted at different applications. Till now, two processors namely Spock and Schubert have been designed for research.

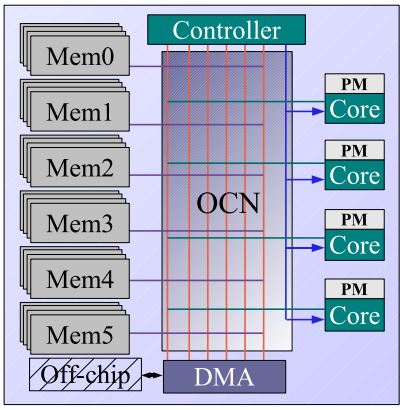

Fig. 2: Architecture of MediaDSP

A. Spock: A Single-Issue DSP

As is shown in Fig.3, Spock is currently a scalar DSP processor consisting of a 16×16 MAC, an ALU and a $32 \times 16b$ register file. It is responsible for the execution of tasks without high parallelism that can be utilized by vector machine, such as audio coding and entropy coding in video compression.

Since traditional scalar DSP processors are inefficient in entropy coding which is mainly bit manipulation, instructions for bit stream parsing will improve the entropy decoding performance by 3–7 times [1]. In order to provide fast processing capability of entropy coding such as VLC and AC, an internal acceleration unit called EnP [2], [3] is designed and integrated into the datapath of Spock which enables the full coverage of basic inner-loop operations used in entropy coding, which is flexible enough for various existing entropy coding methods involved in heterogeneous video compression standards (e.g. the instruction showzeros returns the number of leading zeros before the first '1' in the bit stream within one cycle, which is very suitable for searching binary codes of various length with large number of leading zeros).

Besides the instruction extension targeted at entropy coding, conditional execution is supported in Spock in order to reduce the overhead brought by conditional branches (e.g. jump).

For low-end applications, one Spock core can act as the controller of other cores and the crossbar OCN. It can issue transaction commands to the DMA controller and initiate the tasks running on Schubert. Furthermore, Spock can also be designed as a simple version of superscalar to supply higher computing power for high-end applications.

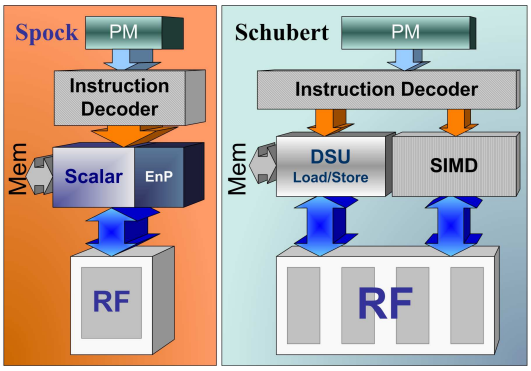

Fig. 3: Processor Cores of MediaDSP

B. Schubert: A Microcode based Processor

Schubert is the name of a two issue four-way SIMD architecture based on microcode. The trade-off between flexibility and silicon costs is explored by using microcode directly as the program. Since high parallelism exists in signal processing such as multimedia and baseband processing, both ILP and DLP need to be explored in architecture design. Currently Schubert includes one four-way SIMD datapath and one data shuffling unit (DSU). The DSU is a single scalar unit which is responsible for branch, address generation, DMA preparation, data load/store and permutation. Both the SIMD datapath and the DSU support conditional execution. The separation of datapath and DSU enables the datapath to continue data processing while the DSU is preparing the data for next round of execution, such as load/store and permutation of data stored in the register file. In case the SIMD datapath is the only one using microcode, a good trade-off will be achieved.

Two working modes called Customized mode and SIMD mode are currently supported in the SIMD datapath. In the Customized mode, customized instructions are issued for irregular operations optimized for specific operations. For example, the operation most intensively used in motion estimation is sum of absolute different (SAD), which can be realized by an instruction mapped to a tree-based datapath.

The SIMD mode issues general SIMD instructions to the four-way datapath. Subroutines such as deblocking involve a large number of branches in inner-loop operations. General branch execution such as JUMP is not efficient enough to achieve high performance. In order to achieve DLP and reduce branch penalty, conditional execution has been adopted by Schubert. Based on the execution flag, the datapath decides whether it should write the result back to the register file (to update the destination registers with the new result) or just skip it. Although with conditional execution, the cycle cost of certain subroutine is fix which is the

worst case, it is still more efficient compared with the traditional branch execution because both extra cycles for JUMP are saved and the data parallelism is exploited along the four-way datapath.

Schubert currently has 64 general-purpose registers, 64bit wide each, which enables vector operations in different combination such as four-way 16bit and two-way 32bit vector operations. In order to load data stored in the off-chip memory, Schubert can initiate DMA transaction directly.

C. On-chip Memory

Since MediaDSP is targeted at specific application domains, current design does not include cache. However, the solution with cache is under benchmarking. In current design, both Spock and Schubert have their own local program memory. As for local data storage, Spock has its own data memory and tap memory which are both scratch-pad. As illustrated in Fig 2, a scratch-pad memory based memory cluster called 2D memory (Mem0...Mem5) can be connected to Schubert as its local storage. The connection can be either fixed or dynamically configurable. In current design, each of these memories contains four memory banks. The reason is that in image and video processing, most of the data are two dimensional which enables high MLP and in the latest video coding standards, 4x4 block is usually the basic data unit to be processed. By having the 2D memory architecture, four 16bit pixel values can be access simultaneously from the four memory banks thus the four-way SIMD datapath can be fed without encountering memory bottleneck. The DSU is responsible for the prefetch of data from the external memory to the local storage before the data is needed by the datapath, thus makes the data movement cycle implicit.

Generally there are two memory organization modes for parallel architectures: shared memory and distributed memory. For the sake of simplicity and efficiency, currently MediaDSP employs a physically distributed and logically shared memory organization with single program flow. Shared memory means that, though each core has its own local storage, there is only one addressing space. Thus the address of data stored in the local storage needs to be explicitly mapped to the global address space.

D. On-chip Network (OCN)

Originally, OCN is a crossbar network connecting the processor cores, on-chip memories and the DMA controller. Physically, data bus and control bus are separated in order to make the design simple. The OCN is configured during the initialization of the platform by the central controller (e.g. a Spock processor). During the configuration, one 2D memory is connected to a processing core and the connection can be fixed during the run-time or be reconnected to another core during the run-time under the control of controller. Thus it enables two modes: processor cores with dedicated local storage and those with dynamically connected local storage.

According to different applications, the OCN can be configured to enable different multiprocessor architectures. For example, Fig. 4 illustrates the case that several processor cores are connected in a chain. Other modes such as parallel and combinational modes are similar while the cores are differently connected. In the chained mode, if each core has its dedicated local storage, then data need to be transfer from the current core to the next core via DMA or ping-pong buffer, which will introduce a large amount of data swapping. Instead, if a 2D memory can be dynamically reconnected to another core when the current core finishes processing data stored in this memory, data swapping can be avoided. To be noted is that the dynamic reconnection might also bring extra overhead. It is still under investigation which mode is more suitable for a certain application.

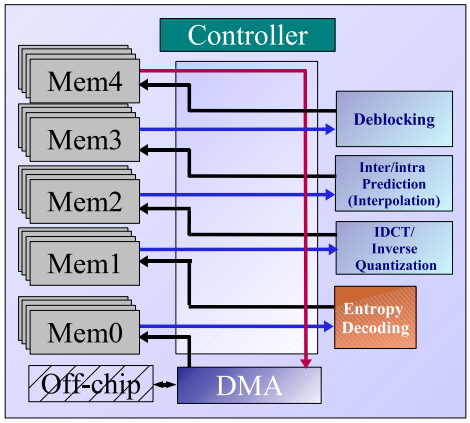

Fig. 4: Vertical Connection

For high-end applications, a large number of cores need to be connected. Therefore the OCN should be scalable. From our experience, single layer crossbar is only suitable for connecting small number of cores. Multi-layer interconnection network-on-chip with low latency and necessary scalability is under investigation by us. Our research is focus on latency hiding and low cost rather than flexibility.

E. DMA

The intelligent DMA with low silicon cost with our unique adaptive rate control features is responsible for swapping data between on-chip memories and the off-chip memory. It supports both burst and sporadic read/write modes. In current design, both Spock and Schubert can initiate DMA transactions to any global address. In the parallel mode, the DMA is also responsible for moving data from the local storage of one processor to that of another processor. There are several parallel channels in the DMA controller and DMA requests received will be added to one of request queues with different priority. Thus the DMA controller can provide QoS to tasks requiring low latency. Ways to expose the memory access cost during the early design stage is another focus of our research.

IV. Task Partition and Scheduling

In the early design, MediaDSP has a centralized architecture. The controller dispatches control information which is called control packet to each processor and initiates the task to be executed on each processor by assigning starting address in the local program memory of each processor.

Fig. 4 illustrates the tasks partition and data flow when mapping H.264 decoding to the MediaDSP platform. The design consists of six cores. Entropy decoding is mapped to one Spock processor and vector operations such as transform, interpolation and deblocking are mapped to three Schubert processors.

V. Toolchain

Since the number of processor cores and accelerators integrated into MediaDSP can be large, the design and verification complexity might increase rapidly. At the same time, how to program a multiprocessor SoC is another big challenge. Thus the design of both simulator and compiler is prominently important in MediaDSP design.

A. Simulator Framework

When designing the simulator for MediaDSP, several factors need to be considered carefully. Since the design complexity of the targeted architecture is increasing dramatically, it is a challenge to maintain high simulation speed while exposing necessary details. The second issue is simulator scalability. Since different design solutions will contain various numbers of processor cores and memory architectures, it is important that a legacy simulator can be easily tailored or expanded for the new solution or to incorporate components from other vendors in order to save investment and time-to-market.

Two simulator frameworks have been proposed and designed by us. The first framework is block based design using C++, in which instructions can be mapped to different blocks. This simulator framework supports modeling with different level of abstraction, which makes it scalable from modeling single processor to the whole MPSoC. Limited by the number of pages, only the second one which is a SystemC based simulator framework will be described in details as it follows.

Recently, both the academia and industry have realized the importance of establishing a standard simulation library that supports simulation with different level of abstraction. SystemC [4] is basically a C++ library for modeling and simulation from RTL level to high abstractive level targeted at hardware/software codesign. By adopting the latest methodology of HW/SW codesign such as transaction level modeling (TLM) [4], a SystemC based simulator framework with different level of abstraction is under development targeted at MediaDSP and other MPSoC platform. The framework consists of core simulator, communication simulator and MPSoC simulator.

1) Core Simulator: Two versions of core simulators have been designed both for Spock and Schubert. In the functional simulator, different number of cycles will be assigned to each instruction without considering too much detail such as pipeline stall. In the performance simulator which is cycle accurate, RTL level SystemC is used to expose the features of hardware design. The functional simulator is roughly 2-3 times faster than the core simulator.

2) Communication Simulator: During our development of a high-performance embedded multi-core platform for multimedia, the behavior of independent core is rather deterministic and relatively simple while the communication among different cores and memory access pattern have become the most difficult and performance dominant issues. In order to efficiently simulate the communications behavior of multi-core platform, SystemC TLM was adopted to enable the exchangeability of modules between different levels of abstraction. For example, during the early design stage, the communication simulator exposes the data movement and memory access features. By using TLM modeling, communication can be separated from the behavior model of each core which enables higher level abstraction. In the later development stage, the RTL level modeling of OCN and DMA will expose the behavior of communication with cycle-accuracy.

3) MPSoC Simulator: When the functional core simulator is integrated with the communication simulator, an MPSoC simulator will be available for functional verification. And when the performance core simulator is integrated with the communication simulator with lower level abstraction, it will enable cycle-accurate simulator of the whole MPSoC. Since the simulator is based on SystemC and modular design, it is scalable and requires little effort to accommodate components from other vendors that are developed in SystemC.

B. Compiler

As the number of heterogeneous processors integrated into an MPSoC increases, it has become a great challenge to design a good compiler. In MediaDSP project, both instruction level compilers and task compiler are under development. During the early stage, tasks are manually partitioned and mapped to different processors based on the knowledge of the programmer. Those tasks that are not computationally intensive will be mapped to Spock and compiled by a C compiler targeted with Spock. Intensive inner subroutines such as entropy coding operations executed in Spock and vector operations in Schubert will be manually written in optimized assembly code and embedded in the C code. Finally the compiler will cross compile the whole program.

During the second stage, though tasks will still be manually partitioned. However, partitioned C code on each processor will be compiled by Spock compiler and Schubert compiler into objects and be linked by the linker of controller (e.g. Spock).

The ultimate goal of compiler design is to provide both task compiler which automatically partitions tasks and processor compilers that compile the partitioned tasks into executables. In order to reach this goal, parallel programming model such as OpenMP [5] and vectorization technologies are to be further investigated and adopted.

VI. Performance of Simple Benchmarking

Based on the cycle-accurate Spock and Schubert core simulators, benchmarking of video (H.264 baseline) and audio (OGG Vorbis) coding has been done. The video benchmarking is based on several sequences with difference scenarios and result shows that the combination of one Spock and Schubert can support real-time H.264 decoding at CIF resolution (352×288, 30fps) together with audio decoding (roughly 30MIPS) at 150 MHz working frequency. By having more Spock and Schubert connected to the OCN, decoding capacity of HDTV (1920x1080) can be expected. However, since the communication and MPSoC simulator is still under development, the overall system performance will be benchmarked later.

VII. Conclusion

This paper presents a brief introduction of the ongoing MediaDSP project initiated by us. MediaDSP platform is not just an implementation but rather a way to expose problems in multiprocessor HW/SW codesign and provide us the opportunities to tackle down those problems. Application based case study ensures that our research is not only based on hypothesis but solid ground. The design of MediaDSP is not fixed, instead it will keep evolving driven by the knowledge accumulated by us.

MediaDSP project is supported by STRINGENT program from SSF.

References

[1] Berekovic, M.; Stolberg, H., Kulaczewski, M., Pirsch, P.; M?ller, H; Runge, H.; Kneip, J.; Stabernack, B., *Instruction Set Extension for MPEG-4 Video, Journal of VLSI Signal Processing, Volume 23,issu 1, 27-49(1999)*

[2] Wu, D.; Hu, T.; Liu, D., *A Single Issue DSP based Multi-standard Media Processor for Mobile Platform*, 8th Workshop of Parallel Systems and Algorithms, ARCS 2006

[3] Flordal, O.; Wu, D.; Liu, D., *Accelerating CABAC Encoding for Multi-standard Media with Configurability*, IEEE Reconfigurable Architectures Workshop 2006

[4] Website of SystemC Community (http://www.systemc.org/)

[5] Website of OpenMP API (http://www.openmp.org/)