A New 3-Phase Design Exploration Methodology for Video Processor Design

Wing-Yee Lo, Daniel P.K. Lun and Wan-Chi Siu
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University, Hong Kong
Email: {03723060g, enpkln, enwcsiu}@polyu.edu.hk

Abstract—When making video processor design, conventional design exploration methodologies take extremely long time in parameter optimization but the final design may not necessarily meet the application requirements since the architecture cannot deviate too much from the initial design. To speed up the design process, statistical performance models were used to guide the simulation; however their accuracy is questionable. In this paper, a new 3-phase design exploration methodology for video processor is proposed. It makes use of an almost cycle-accurate performance model to provide information for refining the processor architecture. It can derive the optimal architecture in a much shorter period of time than the conventional methods. We successfully implemented a few video coding/decoding applications on the video processor derived from the proposed methodology. Simulation results show that it outperforms other video processors in both cost and performance perspectives.

I. INTRODUCTION

Design exploration is an architectural optimization methodology that has been adopted in processor design for years. The idea is to develop a software model with configurable structures to simulate the architecture of the target processor. The performance of the architecture can then be analyzed with different parameter settings so as to find out the optimal architecture design. The traced based approaches have been used traditionally for this purpose [1]-[2]. They use very complex simulation models that take extremely long time for the simulation process in order to get the optimal architecture. To improve the efficiency of the design process, various approaches were proposed such as reducing the benchmark program suite size [3], using trace sampling to reduce the run time [4]-[5], varying only the parameters that have maximum impact [6], and using intelligent optimization method such as the genetic algorithm for architecture optimization [7]. These traditional approaches often require good initial processor architecture and use heuristic searching algorithms to adjust only a few pre-defined parameters to fine-tune the architecture design. Such methodology can only allow limited architectural variations controlled by the pre-defined parameters. The optimized architecture may not necessarily meet the application requirements since the architecture is limited by the initial architecture. Besides, it takes time to derive good initial architecture which cannot fulfill the fast time-to-market need of the industry.

Another design exploration methodology is the statistical simulation modeling [8]-[9]. This methodology first generates a synthetic instruction trace based on statistical profiling. The instruction trace is then fed into a trace-driven simulator along with the statistics of branch prediction accuracy, cache miss rate, pipeline stall rate and instructions issued per cycle (IPC), etc. The advantage of this methodology is that the simulation time is very short. However, they are in general not accurate.

In this paper, we propose a new 3-phase design exploration methodology and apply it to video processor design. Instead of presetting the initial video processor architecture and searching blindly on all combinations of parameter settings, the proposed methodology allows modifications to the baseline architecture based on the outputs of a performance model. Unlike the statistical model, the performance model used is hardware-friendly and almost cycle-accurate. The proposed design exploration methodology consists of 3 design phases. It starts with a baseline video processor architecture derived from a few basic design criteria. Based on the outputs of the performance model, major architectural changes are made to the baseline architecture to reduce the performance overheads. Finally the design parameters are fine-tuned based on the performance model to achieve an optimal design. Comparing with the conventional design exploration methodology, the proposed approach requires a much shorter design time and allows a higher degree of freedom in selecting the best design parameters. The proposed methodology was adopted in the design of a video processor. It leads to a substantial improvement over the original design and allows the final video processor to be one of the best in the literature measured in terms of cost and efficiency.

II. PROPOSED DESIGN EXPLORATION METHODOLOGY

The proposed design exploration methodology consists of 3 phases (see Figure 1) which are described as follows:

A. Phase I: Baseline Architecture Development

The first phase starts with a baseline video processor architecture (see Figure 2) designed with the goal of maximizing the degree of data level parallelism (DLP), instruction level parallelism (ILP) and thread level
To maximizing the degree of DLP, we equipped the baseline architecture with our recently proposed parallel memory architecture and the configurable SIMD (CSIMD) structure enabled by the CSIMD look-up table (CSLUT) [10]. The ILP is achieved by dynamically issuing a maximum of 4 instructions per cycle using an intelligent instruction fetch and dispatch unit. The TLP is enabled by adopting the multithreading architecture, which supports the execution of a maximum of 2 threads in parallel. The design of this baseline architecture involves detailed analysis of the kernel operations of video coding and decoding. Due to page limitation, they are not detailed in this paper.

B. Phase II: Major Architecture Changes Using Performance Model

In the second phase, major architecture modifications are performed on the baseline architecture based on the outputs of a performance model, which is a software system simulating the operation of the target processor. The structure of the performance model is very hardware-like and follows the multi-pipeline stage structure of the baseline video processor. There are five pipeline stages in the video processor. They are instruction fetch (FE), instruction issue (IS), register file data retrieval (RF), execution (EX) and result write back (WB). The performance model does not care about the implementation details except it has to ensure the task assigned in each pipeline stage has sufficient time to accomplish. Major modifications on the baseline architecture are performed based on the outputs of the performance model, which include the architecture performance metrics such as instruction per cycle (IPC) as shown in the instruction issue log, profiling data and statistical data. To save the design exploration time, application kernel functions are used as input trace since the kernel functions can contribute up to more than 75% of the execution time of the whole application. The input trace is generated by the compiler and software scheduler. It is basically the instruction sequence with meta data that are useful for performance analysis. The output files generated by the performance model are analyzed to see where the performance bottlenecks of the baseline architecture come from. New features are brought up to reduce the performance bottlenecks. Then the performance model, compiler, ISA and/or software scheduler are modified to include the new features and generate another round of performance model outputs. When most application requirements are satisfied, the video processor architecture can be fine-tuned in phase 3. Figure 3 shows the final performance model after the abovementioned iteration process. The major architectural changes that have been made will be described in next section.

C. Phase III: Architecture Fine Tuning

In the last design exploration phase, the processor architecture is fine-tuned. In this phase, the application trace is input to the performance model for the adjustment of different
parameters to achieve an optimal design. The performance model in fact allows a wide range of parameters to be configured, which include

- Instruction cache organization: cache size, tag size, number of associative sets, etc.
- Number of instructions fetched in one cycle.
- Instruction buffer size.
- Instruction window size (number of instructions from each thread to be selected for dispatch).
- Number of instructions to be issued.
- Multi-functional execution unit.
- Branch prediction algorithm.
- Issue/fetch policy: round-robin, priority order, etc.
- Bypass buses: bypass data between pipeline stages should be supported.
- Number of IO ports in different memory units.
- Out-of-order issue: All or some instructions can be issued out-of-order.

The parameters will be refined until all application requirements are satisfied.

III. RESULTS

By using the proposed design exploration methodology, several performance bottlenecks of the baseline architecture were identified. A brief account of these bottlenecks, the architectural changes that have been made to deal with them, and the resulting improvements are given as follows:

1) Multithreading: The first major architecture change is the support of quad-thread from dual-thread. During design exploration, it was observed that the IPC is increased by 21.92% when the kernel functions are partitioned to execute in 2 threads. While further splitting the execution to more threads should be advantageous, it is desirable if it can be achieved automatically so as to reduce the programming effort of users. A software scheduler was developed for such purpose. The improvement was verified by the performance model that, when using the software scheduler to achieve quad-thread execution, an overall IPC improvement of 39.48% can be achieved compared with using only single thread.

2) Execution units: There are two architectural changes related to the execution units. The first one is the addition of one more Arithmetic Login Unit (ALU). It stems from the profiling results generated from the performance model that significantly more instructions are executed by the ALU than other execution units. Another change is the replacement of the Multiply-and-Accumulate (MAC) units with normal multiplier (MULT) and adder (ADD). It was noticed during design exploration that the lengthy MAC pipeline introduces many pipeline bubbles. Replacing it with normal MULT and ADD pipelines reduce the pipeline length and also the possibility of pipeline bubbles. Besides, it also reduces the branch mis-prediction latency due to the shortened pipeline. The performance model verified that the IPC improvement after the abovementioned modifications is 70.37%.

3) Processor pipeline: The initial video processor supports pipelines of different lengths in various execution units. The instruction issue log generated by the performance model shows that pipelines of irregular length can introduce long execution latency similar to the MAC execution pipeline mentioned above. After going through several rounds of refinement, the final video processor has unified pipeline structure. The performance model showed that this architectural change can lead to a 25.12% IPC improvement.

4) Micro opcode instruction out-of-order issue: To reduce application program size, the video processor supports macro instructions. A macro instruction consists of a sequence of micro opcode instructions. An example is the block based Sum of Absolute Difference (SAD) instruction which is commonly used in H.264 video coding. Due to the long instruction sequence, it is inevitable for these SIMD macro instructions to have pipeline bubbles. It has large performance impact since the most frequent and performance critical operations are grouped into macro instructions. To solve the problem, we modified the baseline architecture to allow the micro opcode instructions in consecutive independent macro instructions to be issued out of order. It effectively fills up the pipeline bubbles. To fully utilize this feature, the software scheduler was refined to put more independent macro instructions together. The performance model showed that the IPC can be improved by 22.04%.

5) Dynamic load balancing: It was noticed during design exploration that the loading between threads is never balanced due to the picture contents and other reasons. The final video processor architecture supports dynamic load balancing. It provides an instruction for the programmer to indicate any function task to be executed by current thread or another thread, depending on which thread finishes the specified functions first. With this support, the performance model showed that a 15.12% IPC improvement can be achieved.

IV. PERFORMANCE COMPARISON

To illustrate the effectiveness of the proposed design exploration methodology and the accuracy of the performance model in predicting the true performance of the processor, we ported several video applications to the final optimized video processor and compared its performance with other video processors in the literature. The final optimized processor is implemented with TSMC 0.13μm technology. The die area is about 8mm² including 32KB instruction cache, 48KB internal SRAM memory and 8KB LUT memory. We show in TABLE I a comparison of the processor frequency and the die area of different video processors designed for encoding or decoding digital video in H.264 format. It shows that the final optimized video processor outperforms others in terms of cost (die area) and performance (frequency). For instance when comparing with [11] and [12], the final optimized video processor works at a much lower frequency to achieve the same frame rate when performing H.264 video encoding. When comparing with [13] and [14], the die area of the final optimized video processor is much smaller to achieve a similar performance in H.264 video decoding. Note that the die area of the design in...
In this paper, we have proposed a new 3-phase design exploration methodology and applied to video processor design. The methodology starts with a baseline video processor architecture. Based on the outputs of an almost cycle-accurate performance model, the baseline architecture, as well as the compiler, ISA and software scheduler can be refined along the design exploration process. The architecture is then fine-tuned by further adjusting different design parameters. The final optimized video processor showed a substantial improvement over other video processors in the literature in terms of cost and efficiency. It illustrated the effectiveness of the proposed design exploration methodology and the accuracy of the performance model in predicting the true performance of the final video processor.

### V. SUMMARY

In this paper, we have proposed a new 3-phase design exploration methodology and applied to video processor design. The methodology starts with a baseline video processor architecture. Based on the outputs of an almost cycle-accurate performance model, the baseline architecture, as well as the compiler, ISA and software scheduler can be refined along the design exploration process. The architecture is then fine-tuned by further adjusting different design parameters. The final optimized video processor showed a substantial improvement over other video processors in the literature in terms of cost and efficiency. It illustrated the effectiveness of the proposed design exploration methodology and the accuracy of the performance model in predicting the true performance of the final video processor.

### TABLE I. COST AND EFFICIENCY COMPARISONS WITH OTHER VIDEO PROCESSORS.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Technology (μm)</th>
<th>Die Area (mm²)</th>
<th>Frequency (MHz)</th>
<th>Resolution</th>
<th>Frame Rate (fps)</th>
<th>Application</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final optimized processor</td>
<td>0.13</td>
<td>8</td>
<td>90</td>
<td>VGA</td>
<td>25</td>
<td>H.264 Decode</td>
<td></td>
</tr>
<tr>
<td>[11]</td>
<td>0.13</td>
<td>7.45</td>
<td>120</td>
<td>CIF</td>
<td>25</td>
<td>H.264 Encode</td>
<td></td>
</tr>
<tr>
<td>[12]</td>
<td>0.13</td>
<td>32.24</td>
<td>90</td>
<td>CIF</td>
<td>25</td>
<td>H.264 Encode</td>
<td></td>
</tr>
<tr>
<td>[14]</td>
<td>0.065</td>
<td>14.94</td>
<td>333</td>
<td>VGA</td>
<td>30</td>
<td>H.264 Decode</td>
<td>8 MPE is 5.06x5.06, needs 3 MPE: ~(5.06x5.06)<em>(1/3+2/3</em>3/8)</td>
</tr>
</tbody>
</table>

### REFERENCES


