Modeling Arbitrator Delay-Area Dependencies in Customizable Instruction Set Processors

Siew-Kei Lam
Centre for High Performance Embedded Systems, Nanyang Technological University, Singapore
(assklam@ntu.edu.sg)

Mohammed Shoaib
Indian Institute of Technology, Madras, India
(ee03b111@ee.iitm.ac.in)

Thambipillai Srikanthan
Centre for High Performance Embedded Systems, Nanyang Technological University, Singapore
(astsrikan@ntu.edu.sg)

Abstract

Instruction set customization is becoming a preferred approach for accelerating high-speed demanding applications. In this paper, we present performance and delay-area product estimation models to accelerate the design of custom instructions on the Nios II configurable processor platform. The proposed models outline the performance bandwidth and delay-area product to enable profitable selection on the type and number of custom instructions, without the need to undertake time-consuming hardware synthesis in the design exploration stage. The models exhibit a high degree of accuracy as they incorporate the architectural dependencies of the arbitrator logic between the Nios II processor and custom hardware. Experimental results reveal that the area-time implications of the arbitrator logic with respect to the number of custom instructions can significantly affect the system's performance and area utilization.

1. Introduction

General-purpose processors tend to suffer from performance when executing non-standard operations that are not supported by the instruction set. On the other hand, full hardware implementations necessitate lengthy design time that increases the TTM (Time-To-Market) pressure. In recent years, configurable processors [1][2][3][4] have emerged to bridge the full hardware and processor based implementation gap in the SoC (System-on-a-Chip) design continuum. The fundamental premise of configurable processors is that they can be modified or extended to address specific design issues by changing the processor's feature set without prohibitive cost and lengthy development time. In order to support post-manufacturing configurability, commercial RISPs (Reconfigurable Instruction Set Processors) [5] such as the Altera Nios [6] and Xilinx Microblaze [7] platforms have been introduced. A RISP facilitates critical parts of the application to be implemented on a reconfigurable fabric using a specialized instruction set.

Despite the advantages of configurable processors and RISPs, selecting custom instructions from a large number of possible candidates still remains an open issue [2]. Existing methodologies for custom instruction selection that incorporates a hardware synthesis flow in the design exploration process such as that proposed in [8] can become too time consuming. Others methods such as [9] aims to expedite the custom instruction selection process by modeling the performance gain of the custom instruction candidates, but fail to consider the inherent micro-architectural dependencies resulting from the generation of custom logic in RISPs.

In this paper, we introduce accurate performance and delay-area product estimation models that can be used to effectively select custom instructions for the Altera Nios II platform. We will show that the area-time implications of the arbitrator logic are substantial enough to affect the overall system performance and area utilization. Unlike previous works, the proposed models incorporate the architectural dependencies of the arbitrator logic between the Nios II processor and custom logic.

In the next section, we will provide an introduction to the Nios II platform. Section 3 illustrates an example of the Nios II system development flow that has been modified to include model-based custom instruction selection. Section 4 describes the proposed performance estimation model that incorporates the arbitrator's dependencies on the number of custom instructions. In the next section, the delay-area product estimation model is introduced with an example to justify its significance for area-time trade-off study. Finally, the paper concludes with some considerations for future directions.

Proceedings of the Third IEEE International Workshop on Electronic Design, Test and Applications (DELTA'06)
0-7695-2500-8/05 $20.00 © 2005 IEEE

2. The Nios II Configurable Processor Platform

The Nios II configurable processor platform used to derive the estimation models consists of the Altera Stratix II EP2S60F672-C5ES device and can cater up to 256 custom instructions. As shown in Figure 1, the custom instruction logic connects directly to the Nios II ALU (Arithmetic Logic Unit). There are different custom instruction architectures available to suit the application's requirements. The architectures range from simple, single-cycle combinatorial architectures to extended variable-length, multi-cycle custom instruction architectures. The most common type of implementation is the single-cycle combinatorial architecture that allows for custom logic realizations with two inputs and one output port.

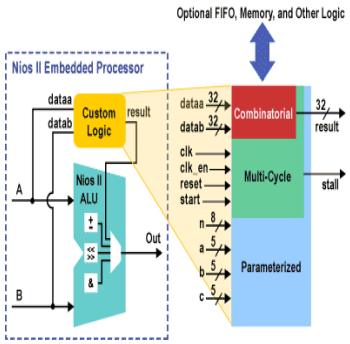

Figure 1: The Nios II configurable processor architecture

The Nios II platform utilizes the Avalon bus module to interface the Nios II processor with the custom logic, and other peripherals as shown in Figure 2.

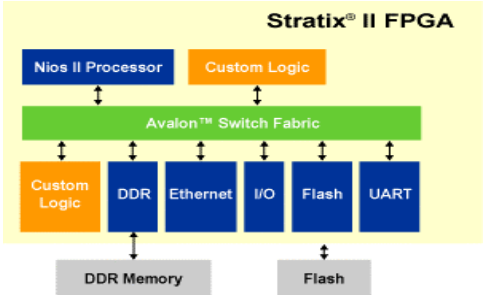

Figure 2: The Avalon bus module

The module is a configurable bus architecture that is auto-generated to fit the interconnection needs of the designer's peripherals. The Avalon bus module consists of the control, data and address signals, and arbitration logic that are connected to the peripheral components. In this paper, we will show that the delay and area of the arbitration logic grows deterministically with the number of custom instructions, which facilitates the design of accurate models.

3. Modified Development Flow

Figure 3 describes the Nios II system development flow with emphasis on instruction set customization. The process to analyze the system requirements have been modified to include two tasks: 1) Profiling to identify critical portions of the application, and 2) Model-based custom instruction selection. In the latter, the proposed estimation models can be employed to provide more reliable design exploration for selecting custom instructions in order to meet the application requirements.

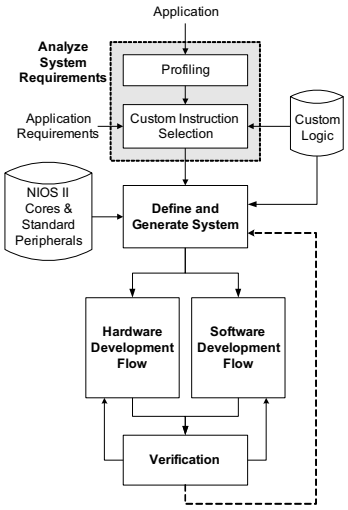

Figure 3: Modified NIOS II development flow

Next, the Nios II processor, required peripherals and custom logic are defined and automatically generated. The resulting system becomes the target platform for the hardware and software development flow. These development flows may undergo several iterations until the system has been successfully verified. In addition, it may be necessary to predefine and generate a new system with different choices of custom instruction logic in order to meet the application's requirements. This will incur iterations (denoted by the dashed line) that significantly increase the design time. The modified flow aims to reduced such iterations by providing more reliable custom instruction selection through accurate estimation models early in the design phase.

4. Proposed Performance Estimation Models

This section describes the proposed performance estimation model. It is noteworthy that the models were derived from experimentation based on unconstrained designs. Although more optimal solutions can be realized by setting tighter constraints for the synthesis process, this would lead to substantial expansion in the area utilization. Moreover, our attempt to formulate a model for constrained designs is hampered by the difficulty in obtaining deterministic results. As such, the proposed models are well suited for early design considerations to reduce subsequent optimization efforts (i.e. during the hardware development flow in Figure 3). This is justified by the fact that constraint settings are usually application dependent, and constrained-based design exploration can be a time-consuming process.

4.1. Modeling the Arbitrator Delay Dependencies

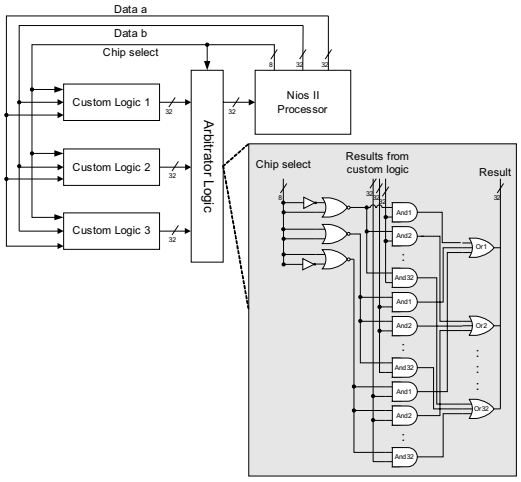

Figure 4: The arbitrator logic

The arbitrator dependencies on the number of custom instructions were first studied. Figure 4 shows the overview of the arbitrator between the Nios II processor and the custom logic, and a schematic diagram of the arbitrator logic for a three custom logic implementation.

The arbitrator employs a complemented BCD coding using the chip select signals to select the required custom logic result. It is evident that the complexity of the arbitrator will grow with the number of custom instructions. For example, as the number of custom instructions increases, the number of Nor gates which serve as the BCD decoder will increase along with number of And gates and Or gate levels. Hence, although the arbitrator logic hierarchy remains the same (i.e. Nor-And-Or) with the increase of custom logic, the number of gates will grow accordingly, introducing combinatorial and routing delays through the arbitrator. The critical path through the arbitrator was experimentally obtained and was found to follow a deterministic pattern with increasing number of custom instructions as shown in Figure 5. It is noteworthy that the delay through the arbitrator can significantly affect the maximum clock frequency of the system.

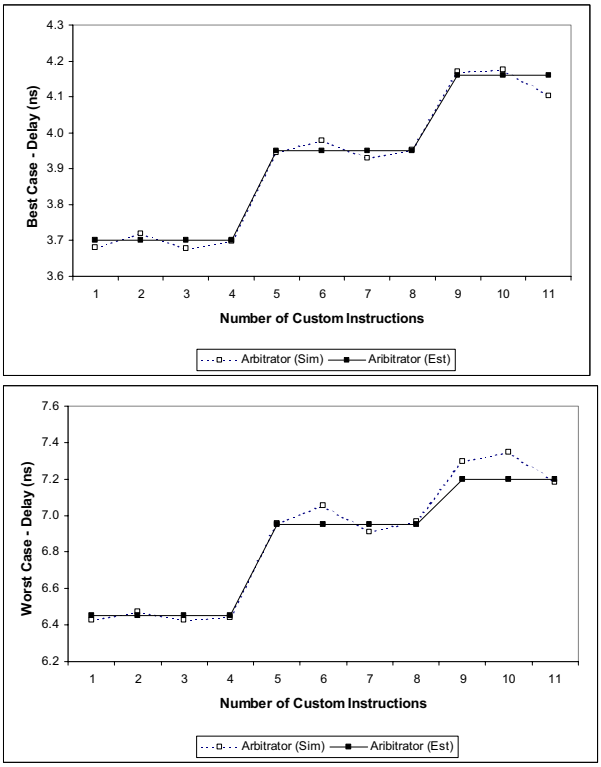

Figure 5: Simulated (dotted lines) and modeled arbitrator delay with number of custom instructions

The variation in the delay observed from Figure 5 is step-wise predictable due to the regular increase in the arbitrator's logic with the number of custom instructions. In addition, the height of the step reduces due to the marginal delay incurred in the arbitrator hierarchy with increasing number of custom logic.

Equations (1) and (2) describe the proposed delay models of the arbitrator with respect to the number of custom instructions (c). When compared to the simulated results in Figure 5, the models exhibit an average percentage error of only 0.1232% and

0.3476% for the best and worst case respectively. In addition, the maximum percentage error for the arbitrator delay model is only 1.414% and 2.041% for the best and worst case.

$$\text{Best case arbitrator delay } (ns) = T_{arb}^{best} = \begin{cases} 3.70 & n < 1 \\ 3.70 + \sum_{m=1}^{n} (0.25 - 0.04(m-1)) & n \geq 1 \end{cases} \quad (1)$$

$$\text{Worst case arbitrator delay } (ns) = T_{arb}^{worst} = \begin{cases} 6.45 & n < 1 \\ 6.45 + \sum_{m=1}^{n} 2^{-m} & n \geq 1 \end{cases} \quad (2)$$

$$n = \lceil \log_2 c \rceil - 2 \quad (3)$$

4.2. Performance Estimation Model

Based on the arbitrator delay model, we introduce a performance estimation model that provides an indication of the speedup achieved over the full Nios II processor implementation, when custom instructions are introduced. The proposed performance estimation model P is shown in equation (4), where $(Clock\ cycles)_{proc}$ and $(Clock\ cycles)_{proc+hw}$ is the number of clock cycles when the entire application is implemented on the Nios II processor, and when custom instructions are introduced respectively.

$$P = \frac{(Clock\ cycles)_{proc} \times T_{proc}}{(Clock\ cycles)_{proc+hw} \times T_{proc+hw}} \quad (4)$$

T_{proc} is the clock delay when no custom instructions are introduced and can be potentially very low (i.e. 7.22 ns). $T_{proc+hw}$ is the clock delay when custom instructions are introduced and is limited by the arbitrator and the custom logic delays as described in (5), where T_{arb} can be calculated from (1) and (2), and T_{hw} denotes the longest critical path of the custom instructions.

$$T_{proc+hw} = \max\left(T_{proc}, T_{arb} + T_{hw}\right) \quad (5)$$

It is noteworthy that although $(Clock\ cycles)_{proc+hw}$ is generally lower than $(Clock\ cycles)_{proc}$, the clock frequency that can be applied after introducing custom instructions (i.e. $\frac{1}{T_{proc+hw}}$) is restricted by the arbitrator delay and critical path of the custom logic. Hence, in general, we find that $T_{proc+hw} \geq T_{proc}$. The proposed performance estimation model P therefore provides a more effective means for selecting profitable custom instructions by taking into consideration the reduced clock cycles and maximum clock frequency that can be applied.

4.3. Validating the Proposed Performance Estimation Model

In order to validate the proposed performance model, a test program was written with up to two million custom instruction execution calls. This is a reasonable number in data-intensive applications such as image processing. For example, if every pixel in a 1024x768 image necessitates only one custom instruction call, this would already incur close to 0.8 million calls for a single image frame processing. In order to validate the models with respect to the number of custom instruction, each custom logic were made identical to that shown in Figure 6.

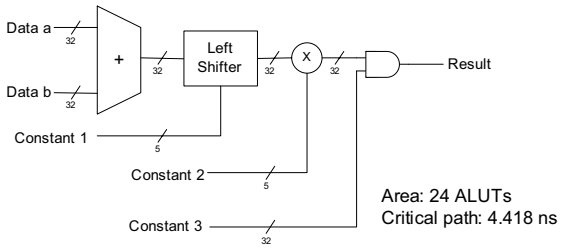

Figure 6: Custom logic implementation

The delay introduced by the arbitrator due to the critical path and arbitrator delay is significant and becomes a performance bottleneck, which restricts the maximum clock frequency that can be applied. For example, a full Nios II processor implementation can be clocked at 138.5 MHz, while the maximum frequency that is applied for increasing number of custom instructions varies as shown in Figure 7. The limitation in the operation frequency of the customized architecture introduced by the delay through the arbitrator can be observed to be dependant on the number of custom instructions as we maintained all the custom logic to be of the same type.

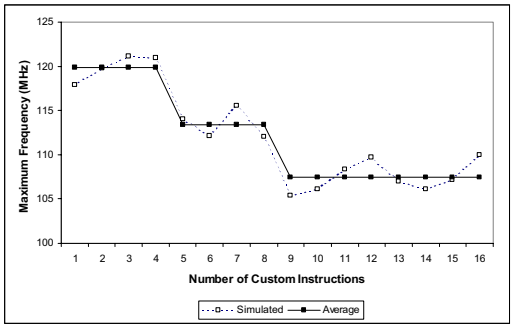

Figure 7: Simulated (dotted line) and average maximum operating frequency with number of custom instructions

By substituting T_{proc} = 7.22 ns, and the maximum operating frequency obtained in Figure 7 in equation (4), the performance estimates of the system can be obtained. Figure 8 shows the computed best and worst-case performance estimation model, and the actual simulated result (dotted line). We observe the expected performance falls well within the proposed bandwidth. It is noteworthy, that the simulated results will vary for different data sets, but it is guaranteed that they will always fall within the performance bandwidth. In addition, the average simulated performance (non-dotted line) exhibits a similar pattern to the proposed model, further justifying the accuracy of the proposed model.

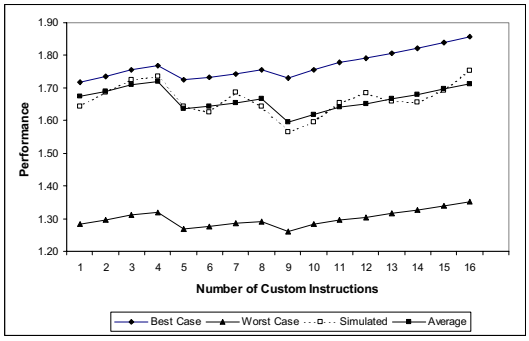

Figure 8: Validating the performance model

The proposed performance models can be easily extended to multi-cycle custom instruction architectures as it is based only on the number of clock cycles and clock delay, where the estimated clock delay can be derived using (5).

5. Modeling the Arbitrator Delay-Area Dependencies

In cases where area-time trade-offs are of concern, the delay-area product provides a more effective metric for custom instruction selection. To illustrate the significance of delay-area product in design exploration, we use an example to evaluate the feasibility of employing extended custom instruction architecture on the Nios II platform.

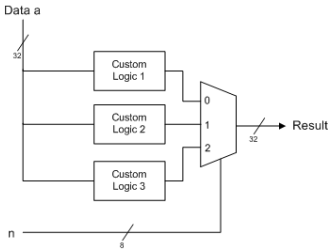

Figure 9: Extended custom instruction implementation

Figure 9 shows the extended custom instruction architecture that allows for a single custom logic block to output results for different operations through multiplexers. The choice of using extended custom instruction implementation over the combinatorial architecture may be motivated by the area gain that can be obtained. As shown in the area comparison plot in Figure 10, the area of the multiplexer is lower than the arbitrator for different number of custom instructions. It is also evident that the area incurred by the multiplexer and arbitrator is substantial when compared to the area utilization of the custom logic in Figure 6.

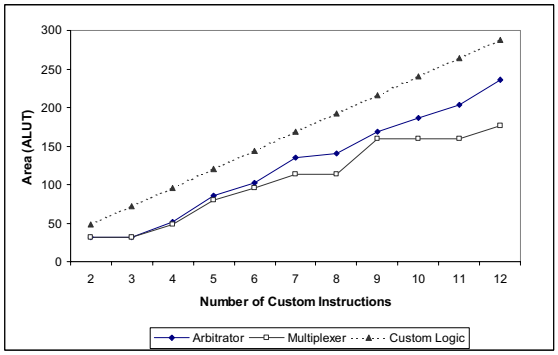

Figure 10: Area comparison between the arbitrator and multiplexer (estimated area of custom logic is also shown)

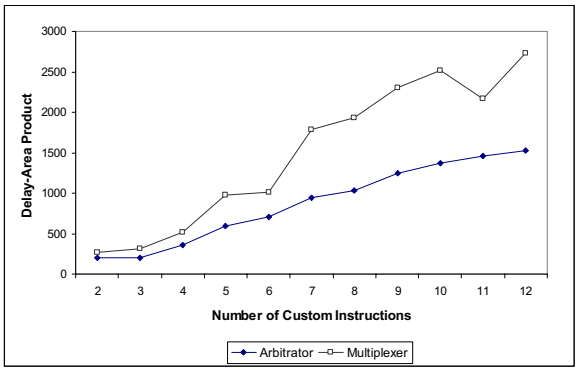

Figure 11: Delay-area product comparison between the arbitrator and multiplexer

However, the delay-area product comparison shown in Figure 11 reveals that the combinatorial implementation provides the best area-time trade-offs, and could therefore be a better choice for custom instruction implementation. The proposed delay-area product estimation models are shown in (6) and (7), where c is the number of custom instructions.

$$Best\ case\ delay-area\ product = 40(2c-1) \quad (6)$$

$$Worst\ case\ delay-area\ product = 150(c-1) \quad (7)$$

Figure 12 shows the best case and worst-case delay-area product of the arbitrator with increasing number of custom instructions. The simulated values of the delay-area product are shown in dotted lines, which increase in a near-linear fashion for higher orders of custom instructions.

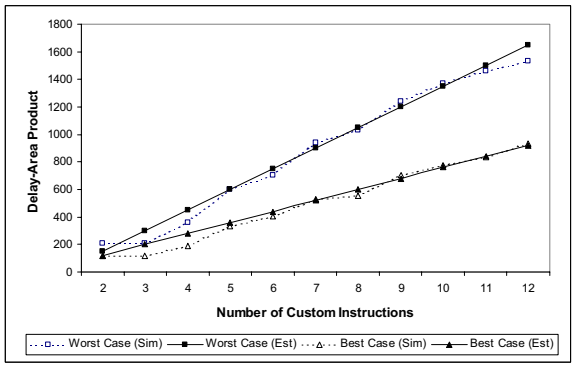

Figure 12: Delay-area product of arbitrator with number of custom instructions

6. Conclusion

We have proposed accurate estimation models to facilitate efficient custom instruction selection for the Nios II configurable processor platform. The models incorporate the arbitrator's dependencies, which have been shown to be significant enough to affect the overall system's performance and area utilization. An analysis of the arbitrator reveals that the critical path increases deterministically with increasing number of custom instructions. The performance estimation model takes into consideration, the reduced clock cycles and maximum operating frequency when custom instructions are incorporated. In addition, delay-area product estimation model was proposed to facilitate area-time trade-off based design exploration. The proposed models provide for rapid analysis in the custom instruction selection process. Finally, the proposed models are being explored to further improve their accuracy for highly constrained designs by taking into account other inherent micro-architectural constraints and dependencies in RISPs.

7. References

1. Dutt, N., Choi, K.: Configurable Processors for Embedded Computing, Computer, Vol. 36, No. 1, 2003, 120-123
2. Henkel, J.: Closing the SoC Design Gap, IEEE Computer, Vol. 36, No. 9, 2003, 119-121.
3. Xtensa Microprocessor: http://www.tensilica.com
4. ARCtangent Processor: http://www.arc.com
5. Barat, F., Lauwereins, R., Deconinck, G.: Reconfigurable Instruction Set Processors from a Hardware/Software Perspective, IEEE Transactions on Software Engineering, Vol. 28, No. 9 (2002) 847-862
6. Altera Nios Soft Core Embedded Processor: http://www.altera.com
7. Xilinx Platform FPGAs: http://www.xilinx.com
8. Fei Sun, Ravi, S., Raghunathan, A., Jha, N.K., "Custom-Instruction Synthesis for Extensible-Processor Platforms", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 23, No. 2, February 2004, pp. 216-228.
9. Jason Cong, Yiping Fan, Guoling Han and Zhiru Zhang, "Application-Specific Instruction Generation for Configurable Processor Architectures", International Symposium on Field Programmable Gate Arrays, February 2005, pp. 99-106.