International Journal of Computer Applications (0975 – 8887)
Volume 95– No. 14, June 2014

ADA: Applications Define ASIP

Manoj Kumar Jain
Associate Professor
Department of Computer Science
Mohanlal Sukhadia University, Udaipur, India

ABSTRACT

Interest in Application Specific Instruction set Processors or ASIPs has increased significantly. Sincere efforts have been put in improving ASIP design methodologies in industry as well as in academia. By the close observation and analysis of these approaches, it was found that though the existing approaches are focusing on making the process automatic and providing better GUI to help the designers, core technique used in deciding the suitable architecture (processor and memory) is based on design space exploration. This exploration is done with the help of estimators. Such estimators are either simulator based or scheduler based. This study identifies that both types of techniques are very far from the ideal dream technique in which applications should have defined the suitable architecture configuration and these techniques are becoming unsuitable in current scenario. Each problem has a solution hidden in it. This scenario motivated us to propose a novel and revolutionary ASIP design technique making the dream true. The Proposed technique does not focuses on design space exploration, it focuses on directly defining processors for given applications rather than searching for suitable configuration in a jungle of configurations can be suggested by the architecture design space.

General Terms

Computer Architecture, Processor Design, Processor Design Technology.

Keywords

Application Specific Instruction Set Processor (ASIP), Embedded System Design, Real-time systems, Design Space Exploration.

1. INTRODUCTION

General Purpose Processors or GPP are not suitable for most of the real-time systems. Strict timing constraints are defined for such systems. Real-Time Operating System (RTOS) will not allow to proceed the applications further as violation of these constraints lead to a major disaster in such systems.

Applications Specific Integrated Circuits or ASICs provide a solution to meet out such constraints but at the cost of rigidness. Due to this rigidness, ASICs also become unsuitable for such systems. Application Specific Instruction set Processors or ASIPs provide a solution in such a case which gives performance better than GPPs and also provide the flexibility which is not provided by ASICs.

Jain et al. surveyed ASIP design methodologies, and identified five key steps as application analysis, design space exploration, instruction set generation, software tool set generation and hardware synthesis. They have also classified the existing techniques. The most challenging step is design space exploration as this decides the processor and memory configuration suitable for given applications. Most of the approaches estimate performance and other parameters using various estimates to know how a particular configuration will behave. Various configurations (processor + memory) are chosen one by one and estimates are generated with the help of estimators. The main estimator is the performance estimator which estimates the performance of a selected configuration. Performance estimation is usually performed using a simulator based technique. In this technique, a simulation model of architecture based on selected features is generated and the application is simulated on this model to get the performance estimates.

As simulator based performance estimators are usually slower (as simulations are slower) and the design space explored is also limited due to limitations of the tools used in this approach, another technique emerged as scheduler based approach. In this approach, a retargetable estimator is used to estimate the performance of various configurations.

All design space exploration techniques use a parameterized architecture model. Day by day, the number of parameters and the range of these parameters is increasing. It is leading into exploding the architecture design space. Now it is not a surprise to explore millions of configurations. None of the existing techniques is suitable in the present scenario.

We are presenting a novel technique to handle this situation. The proposed technique will be very useful and will prove to be a revolution in the processor design techniques.

2. RELATED WORK

An Application Specific Instruction set Processor (ASIP) is a processor designed for one particular application or for a set of applications usually from a particular domain. ASIPs are also known as domain specific processors and custom processors. Since the input applications are limited in number, it gives an opportunity to exploit special characteristics of given application(s) to meet out the desired performance, cost and power consumption requirements. ASIPs are balance between two extremes, namely, General Purpose Processors (GPPs) and Application Specific Integrated Circuits (ASICs). ASIPs provide the required flexibility which is not provided in ASICs, and they meet out the performance requirements which cannot meet out by GPPs.

Jain et al. surveyed the state of art in ASIP design methodologies, and identified five key steps as application analysis, design space exploration, instruction set generation, software tool set generation and hardware synthesis [1-3]. According to this survey a typical approach is shown if Figure 1.

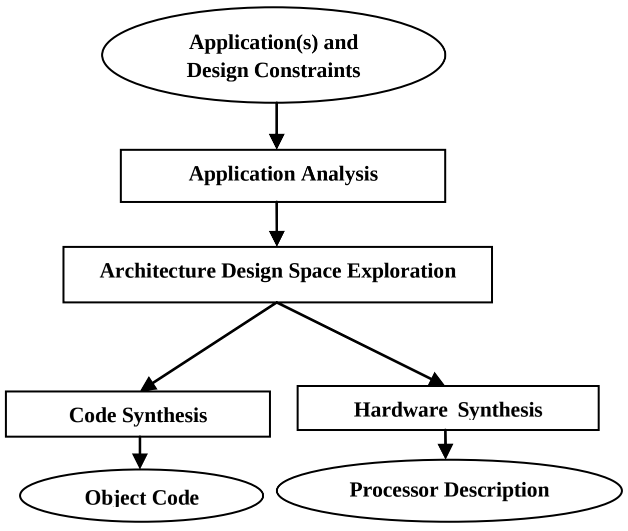

Fig. 1: ASIP Design Methodology

An application written in high level language is analyzed statically and dynamically. The analyzed information is stored in a suitable format and is used in the next steps of ASIP design. All approaches consider a parameterized architecture model for design space exploration. Information generated from the application analysis step, input design constraints and the defined architecture design space are used by the explorer to select suitable architecture or suggest a set of possible architectures. These architectures can be further studied in detail to get the desired architecture configuration. The selection process typically can be viewed to consist of a search technique over the design space driven by a performance estimator. The instruction set is generated either by synthesis or by selection technique. A retargetable compiler is used to generate code. The hardware is synthesized using the ASIP architecture starting from a description in VHDL/ Verilog using standard synthesis tools.

Design Space Exploration: A typical design space explorer takes application parameters extracted in the application analysis as input. It picks up one configuration from the suggested designs in the architecture design space. An estimator will be core part of the design space explorer. This estimator generates estimates for the selected configuration. Based on this estimate and the given design constraints, it is decided that the configuration under evaluation is a candidate or not of the possible designs to be suggested. There is a search controller which will try to trim some options in the design space. An explorer with performance estimator is shown in Figure 2. Examples of such approaches are [4-15].

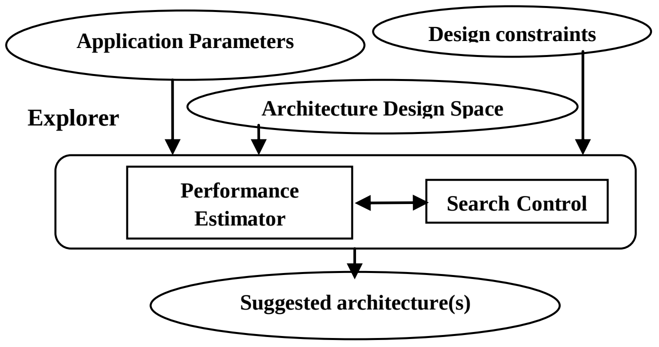

Fig. 2: Design Space Explorer

Performance Estimation: Performance estimation is done in two ways, namely, simulator based and scheduler based techniques. In the simulator based approach, a simulation model of architecture based on the selected features is generated and the application is simulated on this model to get the performance. Such an approach is shown in Figure 3.

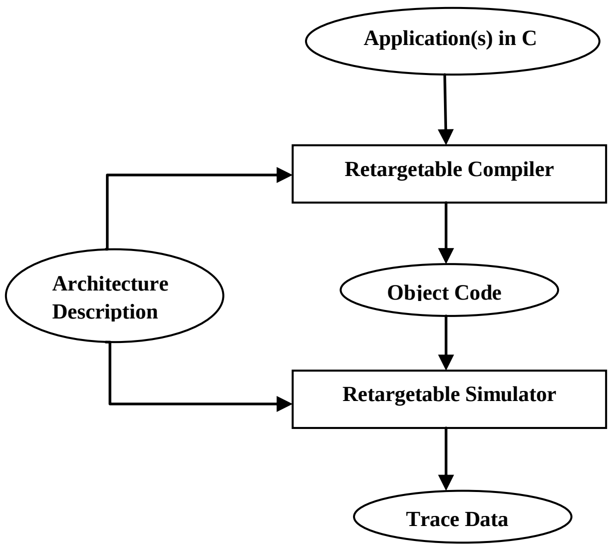

Fig. 3: Simulator based performance estimator

It is important to note that such techniques need and fully dependent on retargetable compilers and retargetable simulators. There are mainly two problems associated with such approaches. Practically it is not possible to get retargetable compilers and retargetable simulators which can address a reasonably large architecture design space (consisting of processor and memory configurations). Hypothetically, even if it is assumed (which is not true) that such tools are available then also the problem is the time required to perform simulation. So this technique which is being used by most of the researchers in industry and academia who are working on ASIP design is becoming unsuitable for design space exploration. Though a lot of work is done on automation of the method, developed better GUIs for the designers, their core technique which is simulator based is becoming obsolete.

Considering the problems associated with the simulator based approaches, another technique proposed is the scheduler based approach. In this approach, the problem is formulated as a resource constrained scheduling problem with the selected architecture components as the resources and the application code is scheduled to generate an estimate of the cycle count. Profile data is used to obtain frequency of each operation. Such an approach is shown in Figure 4. Examples of such approaches are [16-19].

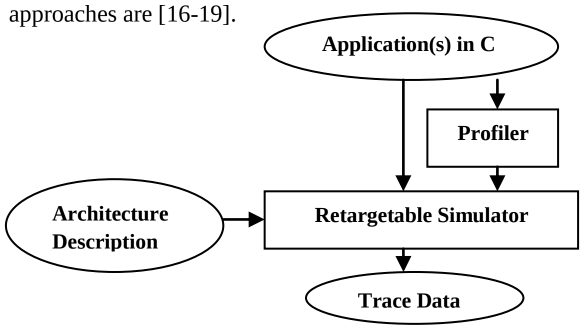

Fig. 4: Scheduler based performance estimator

Since this technique is a bit newer and very few details of architectures are included is the model. So it is not possible to explore large design space using this approach.

Considering problems associated with all the existing approaches of design space exploration and their unsuitability for deciding a suitable architecture configuration in present scenario, some other novel and revolutionary technique is deadly required. An attempt is being made to present such a technique in this study.

3. ADA (APPLICATIONS DEFINE ASIP) A NOVEL ASIP DESIGN TECHNIQUE

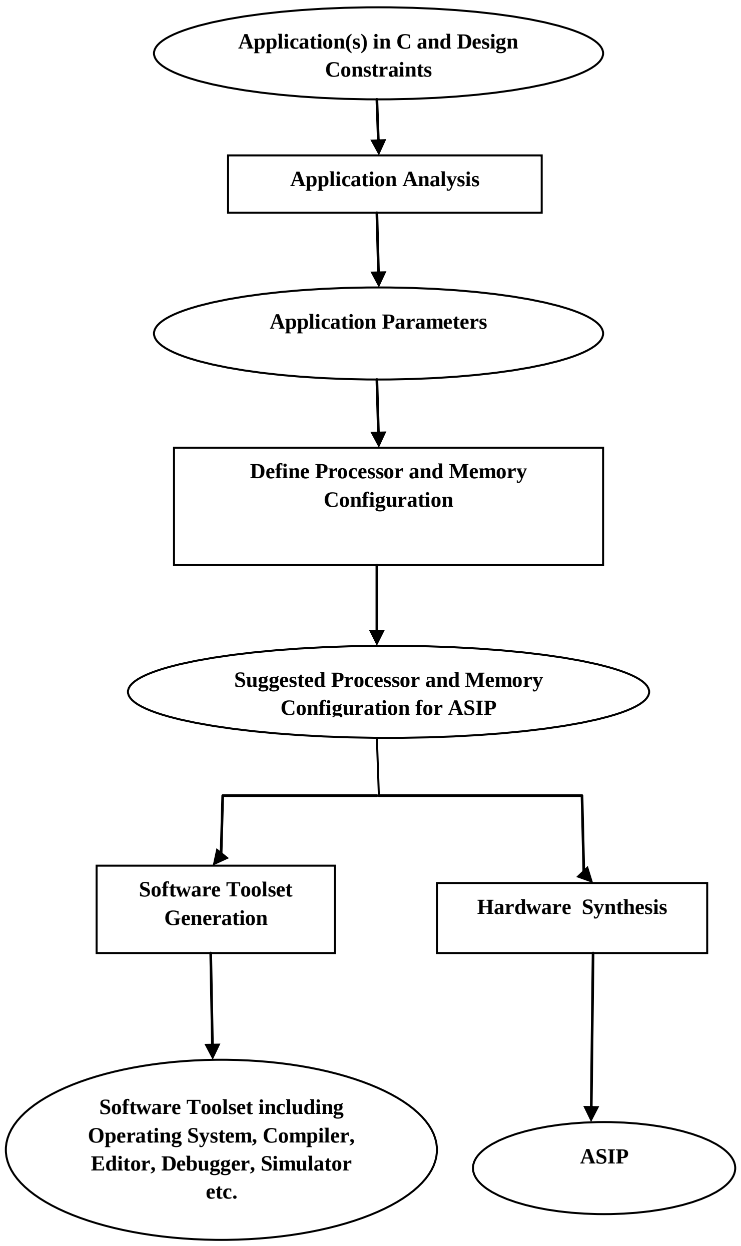
Application(s) in C and Design Constraints
Application Analysis
Application Parameters
Define Processor and Memory Configuration
Suggested Processor and Memory Configuration for ASIP
Software Toolset Generation
Hardware Synthesis
Software Toolset including Operating System, Compiler, Editor, Debugger, Simulator etc.
ASIP

Fig. 5. ADA: Applications Define ASIP [New Technique]

In this Section, a novel ASIP design technique is presented. The idea is simple but its implementation is challenging. After sometime, when the existing techniques will be absolutely unsuitable due to reasons already mentioned in the paper, this new technique will play a vital role in the coming years. Proposed technique is shown in Figure 5.

3.1 Application Analysis

Application or a set of applications need to be analyzed, putting more efforts. One can get advantage of small number of applications which is given to user while designing ASIP. Even short time to prototype and short time to market constraints allow us to perform detail analysis of applications. Though this step is already a part of ASIP design technique but it is not being used in a manner it is supposed to be used

Application to be analyzed statistically (without running it, just analyzing application and its code), and dynamically by running it on the host machine as there are many parameters which can be purely extracted from the application only without getting any details of the target architecture. Some parameters which can be extracted directly from the application and has important role in deciding processor configuration are as follows.

- P1 number and types of operation performed and proportion to the total number of operations .
- P2 data types used and the sites required to store them in memory.
- P3 register needs (locally and globally)
- P4 parallelism available in the applications
- P5 pattern of operations executed frequently
- P6 memory reference trace

3.2 Define Processor and Memory Configuration

Parameter P1 will decide the kind of operations to be performed by the ALU of the proposed ASIP. This is in conjunction with parameter P5 will help in deciding the instructions and corresponding hardware to perform this operation will be provided in the ALU . Parameter P2 in conjunction with P6 will decide almost all the features of memory to be synthesized. Parameter P3 will decide the structure and size of the register file. Parameter P4 will decide the parallelism in the proposed ASIP. If the parameters list is closely observed, it seems that all the parameters except P3 (register needs) hardly depends on the processor architecture. Interestingly, this parameter can also be found out without knowing details of proposed ASIP architecture. Following sub section describes how that is done.

3.3 Estimating the Register Needs

We have earlier studied how to register needs can be extracted. Estimates generated by our proposed approach were validated using standard tool sets for vast range of processors eg. a RISC processor (ARMTTDMI), a VLIW (Trimedia TM-1000), and a processor with register windows (LEON). Due to limitation of the space and the work is already reported [17], is not being reproduced the same here.

The register needs estimated in this way help in many folds. One obvious use is in deciding the size and structure of register file for the desired ASIP. Other advantages are also significant. Once the register file is decided, it helps in deciding the instructions as a typical RISC instruction format includes three register addresses. If our study reveals the fact that there would be some 'spare' register as hardware synthesis allows taking number of registers as power of two. These 'spare registers' and specially 'addresses of these spare registers' can be used in many ways. Few registers can be used as special purpose registers like status registers. Some 'spare register addresses' can be used to simplify coprocessor interface. Such a study is already reported in the literature [20].

3.4 Software Toolset Generation

This step generates software toolset required to make the ASIP useable. Software toolset include essential software like Operating System, Compiler, Editor, Debugger, and Simulator etc. It is important to note the major difference between the existing approaches and the proposed approach. In future, it is proposed to generate software toolset only for the suggested ASIP. In contrast to this, the existing approaches are expecting compilers and simulators for each and every configuration of the possible architecture design space.

3.5 Hardware Synthesis

ASIP is synthesized using standard synthesis tools. For this a synthesizable hardware description of the desired ASIP is provided.

4. CONCLUSION

Interest in ASIP design has increased significantly recently. In this study, the existing ASIP design methodologies have been studied and the kinds of challenges faced by them are identified. These methodologies are soon going to become unsuitable to meet out upcoming challenges, so a novel a novel ASIP design technique is proposed in this study in which each possible architecture configuration is not evaluated and judged for suitability. A suitable architecture is directly suggested. Many more parameters from the applications will be considered in future so most of the ASIP configuration may be decided easily. When the ideal situation will be reached, complete ASIP configuration will be decided without design space exploration.

5. REFERENCES

[1] Jain M.K., Balakrishnan M., and Kumar A., 2001, "ASIP Design Methodologies: Survey and Issues", In Proceedings of the IEEE/ ACM International Conference on VLSI Design. (VLSI 2001), pages 76-81.

[2] Gour Deepak, Jain M.K., 2011, "ASIP Design Space Exploration: Survey and Issues", International Journal of Computer Science and Information Security, Vol. 9, No. 3, 2011, pages: 141-145.

[3] Jain M.K., and Gour Deepak, 2012, "Comparison between the Simulator and Scheduler based approach of Design Space Exploration for Application Specific Instruction set Processor", International Journal of Computer Applications, IJCA, ISSN: 0975-5887, Vol. 43, No. 5, April 2012, Pages 14-19.

[4] Halambi A., Grun P., Khare A., Ganesh V., Dutt N., Nicolau A,, 1999, "EXPRESSION: A Language for Architecture Exploration through Compiler/Simulator Retargetability", In Proceedings of the Design Automation and Test in Europe (DATE), March 1999, pages 485–490.

[5] Pees S., Zivojnovic V., Meyr H., 1999, "LISA- Machine Description Language for Cycle Accurate Models of Programmable DSP Architectures", In Proceedings of the Design Automation Conference (DAC), June 1999, pages 933–938.

[6] Hoffmann A., Kogel T., Nohl A., Braun G., O. Schliebusch O., Wahlen O., Wieferink A., Meyr H., 2001, "A Novel Methodology for the Design of Application-Specific Instruction-Set Processors (ASIPs) Using a Machine Description Language", In IEEE Transactions on Computer Added Design of Integrated Circuits and Systems, 20(11), November 2001, pages 1338–1354.

[7] Radhakrishnan S., 2006, "Customization of application specific heterogeneous multi pipeline processors", In Proc. EDAA 2006, pp. 746 – 751.

[8] Yoonjin Kim, Mary Kiemb, Kiyoung Choi, 2005, "Efficient Design Space Exploration for Domain-Specific Optimization of Coarse-Grained Reconfigurable Architecture", In Design, Automation and Test in Europe, 2005. Proceedings, 7-11 March 2005, page(s): 12 – 17.

[9] Bossuet L., Gogniat G., and Philippe J.L., "Communication-Oriented Design Space Exploration for Reconfigurable Architectures", In EURASIP Journal on Embedded Systems, Volume 2007, Article ID 23496.

[10] Pasricha S., and Dutt N., "A Framework for Memory and Communication Architecture Co-synthesis in MPSoCs", In Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Volume 26, Issue 3, March 2007 Page(s):408 – 420.

[11] Kenshu Seto and Masahiro Fujita, "Custom instruction generation for configurable processors with limited numbers of operands", IPSJ Transactions on System LSI Design Methodology Vol. 3, 57-68, Feb 2010.

[12] Jain M.K., and Ramnani V., 2013, "Design Space Exploration for a Custom VLIW architecture", International Journal of Computer Applications, IJCA, ISSN: 0975-5887, Vol. 61, No. 8, January 2013, Pages: 31-34.

[13] Tensilica Inc., http://www.tensilica.com.

[14] Altera Corp., http://www.altera.com.

[15] Xilinx Inc., http://www.xilinx.com

[16] Gupta T.V.K., Sharma P., Balakrishnan M., Malik S., 2000, "Processor evaluation in an embedded systems design environment", In Proc. VLSI Design 2000, pages 98-103, January 2000.

[17] Jain M.K., Balakrishnan M., and Kumar A., 2002, "An Efficient Technique for Exploring Register File Size in ASIP Design", IEEE TCAD, Vol. 23, Issue 12, December 2004. Pages: 1693-1699.

[18] Jain M.K., Balakrishnan M., and Kumar A., 2003, "Exploring Storage Organization in ASIP Synthesis", In Digital System Design, 2003. Proceedings. Euromicro Symposium, 1-6 Sept. 2003, pages: 120 – 127.

[19] Jain M.K., 2011, "A Multi Layer Technique for Performance Estimation for ASIP Design Space Exploration", International Journal of Advanced Research in Computer Science, IJARCS, Vol. 2, No. 4, August 2011, Pages 648-653.

[20] Jain M.K., Balakrishnan M., and Kumar A., 2005, "Integrated On-chip Storage Evaluation in ASIP Synthesis", Proceedings of the IEEE Eighteenth International Conference on VLSI Design with Fourth International Conference on Embedded System Design, VLSI 2005, 3-7 January, 2005, pages: 274-279.