

Application Specific Processor Core Construction from C Code

Jelena Trajkovic (jelenat@ics.uci.edu)
CECS/ University of California, Irvine
Irvine, CA 92697-3425

Advisors: D. Gajski, A. Veidenbaum
Estimated graduation: September 2008. NOT
presented at ASP-DAC or DATE PhD Forum

Introduction: Application specific processor cores are being used to address the demand for high performance, low area and low power consumption in modern embedded systems. However, the design of such cores is non-trivial and the problem is further exacerbated by the size and complexity of the application as well as the short time to market. In this dissertation, we present a methodology and combination of automation and design techniques for construction of embedded processor cores from application C code.

Although the problem of C based processor design has been around for a while, its adoption in the industry has been limited. One factor is the scalability of the solutions. The other is the controllability of the design process. Our contribution is that the proposed techniques are not limited by the size of the C program and they provide choice of interactive design for controllability. Furthermore, the processor designs produced by our technique are close to manually created designs in performance and area.

We present processor core generation in two parts. The first part addresses the design of the datapath that meets performance constraints while maximizing the utilization of function units. We present a heuristic based automatic datapath refinement algorithm as part of an interactive design process. The second part presents techniques for design of energy efficient pre-fetch buffers that reduce overall power consumption of the core memory subsystem.

Datapath Construction: Fig. 1 shows the steps of proposed C-to-datapath construction technique. We start by characterizing the C code with data types, operators, variables, parallelism, loops and dependencies. We developed heuristics for mapping the code characteristics to hardware components (functional units, storage elements, etc.) and structure (pipeline, connectivity, etc.) and create the Initial Datapath (DP). The initial datapath is used for compiling and profiling the given application code. The compiled code is then analyzed and the architecture undergoes possibly several iterations of refinement and estimation during the datapath optimization step.

The estimation algorithm, shown in Fig. 2, is used to quantify the performance impact of

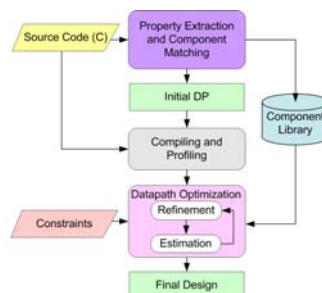


Figure 1: Datapath extraction technique

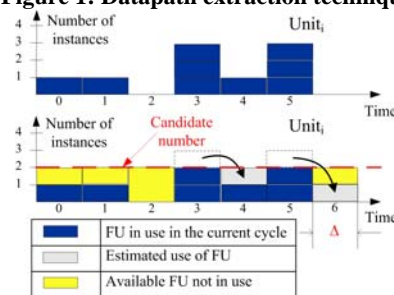


Figure 2. Estimation algorithm - example

varying the number of instances of a component in the datapath. First, a utilization graph (top of Fig. 2) is created for each component per basic block. Then, the number of instances of the component is reduced to candidate number of units, which are two in this example. As it can be seen, with only two units, there is one extra operation in cycle three and in cycle five that can not execute in the originally specified cycle. The estimation algorithm finds a consecutive cycle that has an available component (such as cycle four and cycle six) and uses it for delayed execution of the extra operation. By modeling the delayed execution the tool estimates the execution overhead (Δ) caused by the use of the candidate number of units. The computed overhead is compared to the overhead specified by the constraints and the candidate number of units is incremented or decremented in order to meet the constraint. The same approach may be applied for using component utilization and power dissipation as constraints.

After the specified constraints have been satisfied, all the components and structures are instantiated to create a synthesizable RTL datapath netlist. The size and performance of the generated datapath can be controlled by designer constraints.

For the experiments presented in Table 1, the

designer specified the register file configuration (number of ports) and pipeline configuration (non-pipelined or pipelined). The tool then balanced out remaining components and connections such that performance and utilization are maximized. The baselines for the comparison are manual designs that were developed by computer engineering PhD students who started from the same C code. Controllers for all designs were produced by the same, third party tool that generates statically-scheduled horizontally-microcoded controller. Large input data sets were used for profiling.

Table 1. Experimental results

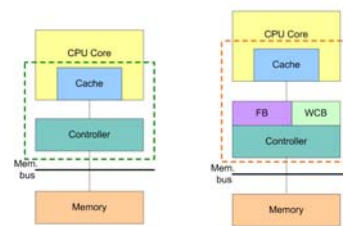
| Bench | Speedup [%] | Area [%] | GenT[s] |
|-------------|-------------|----------|---------|
| bdist2-non | 14.2 | 23.8 | 0.2 |
| bdist2-pipe | 8.4 | 25.2 | 0.8 |
| dct32-non | -0.6 | 28.1 | 1.3 |
| dct32-pipe | 21.4 | 28.8 | 2.3 |
| Sort-non | 1.5 | 1.3 | 0.1 |
| Sort-pipe | 0.0 | 1.4 | 0.1 |
| Mp3-non | 23.5 | 29.3 | 15.6 |
| Mp3-pipe | 27.6 | 30.0 | 42.6 |

Our experimental results show that even for large industrial scale applications, like Mp3 decoder, with over ten thousand lines of C code, our automatically generated datapath is up to 28% faster and within 30% of area of manual design. Compared to months of manual design, automatic generation produces comparable quality datapath in order of seconds.

Prefetch Buffer Design: Another embedded core design problem we addressed was the large energy consumption in DRAMs (dynamic random access memory), exceeding that of the data cache and even that of the processor. In this section, we present and evaluate a scheme for reducing the energy consumption of SDRAM (synchronous DRAM) memory access by techniques that take advantage of SDRAM energy efficiencies in bank and row access.

The proposed application specific memory subsystem is shown in Fig. 3. b). Energy saving is achieved by using small, cache-like structures in the memory controller: Fetch Buffer (FB) to prefetch an additional cache blocks on SDRAM reads and Write Combine Buffer (WCB) to combine block writes to the same SDRAM row. Access to the data in FB is significantly cheaper in terms of time and energy consumption, while write combining leads to gains *without any penalty*. Separate fetch and write buffers are advocated for simultaneous use of read and write combining. Prefetching or write combining can be powered down individually to better tune them to a given application or an application

region. In order to utilize power down, compiler analysis, profiling or user directives could be used.



a) Traditional b) Application specific
Figure 3. Memory subsystem design

The results quantify the SDRAM energy consumption of MiBench applications and demonstrate that a significant reduction in memory energy consumption and delays can be achieved by read prefetching and write combining. The latter also reduces the overall execution time which in turn minimizes the static energy consumption. Even with small size buffers, 256B/512B for prefetching and 128B for write combining, an average 23% DRAM energy reduction is achieved. The energy-delay (ED) product is improved, on an average, by over 40%. The cycles per instruction (CPI) are reduced by 26%, on an average.

Conclusion and Future Work: Our datapath design technique provides two key advantages to designers: it is applicable and scalable for any size of C code and it is controllable allowing fine tuning of the designs. For future work, we are enhancing the datapath refinement algorithm to optimize the pipeline configuration for a given C code. We use code analysis to estimate data, structural and control hazards that may occur as a result of pipelining. We also analyze loops to predict the performance impact of adding pipeline stages. Our prefetch buffer design technique provides significant energy savings while it requires simple hardware customization that is suitable for embedded systems. We plan to evaluate the impact of technology scaling on the efficiency of the proposed approach.

Bibliography:

1. J. Trajkovic, D. Gajski 'Automatic Datapath Generation for Custom Processors,' **Best Paper Award**, In Proc. of IESS (June 2007) ISBN: 978-0-387-72257-3
2. J. Trajkovic, A. Veidenbaum, A. Kejarawal 'Improving SDRAM Access Efficiency for Low-Power Embedded Systems,' To Appear in ACM TECS
3. J. Trajkovic, M. Reshadi, B. Gorjiara, D. Gajski 'A Graph Based Algorithm for Data Path Optimization in Custom Processors,' In Proc. DSD (Sept. 2006) ISBN:0-7695-2609-8