Prototyping and Application Development Framework for Dynamically Reconfigurable DSP Architectures

Steffen Köhler, Martin Zimmerling, Martin Zabel, and Rainer G. Spallek
Institute of Computer Engineering, Department of Computer Science
Technische Universität Dresden, D-01062 Dresden, Germany
{stk, zimmerling, zabel, rgs}@ite.inf.tu-dresden.de

Abstract: This work introduces a prototype evaluation framework for a class of digital signal processor (DSP) architectures, which provide increased instruction-level parallelism (ILP), flexibility and scalability through a coarse-grain reconfigurable data-path. The proposed framework differs from the previous approaches in the fact that it provides several views to achieve a common systematic way to evaluate DSP architectures and its dynamic reconfiguration in the application development process. Furthermore, we applied the framework concept to the ARRIVE [5] architecture, which can be characterized by an enhanced RISC microprocessor with a tightly coupled reconfigurable ALU array and a vector load/store unit. These units implement coarse-grain reconfigurable structures by means of switchable contexts. In contrast to previous work, context activation is performed event-driven according to the instruction pointer of the RISC microprocessor. The synchronous operation of the context-controlled functional units enables an ILP comparable to complex VLJW/SIMD processors, without introducing additional instruction overhead. As a result of the ARRIVE FPGA prototype evaluation, we show detailed quantitative performance and utilization results related to the ALU array geometry, memory transfer bandwidth and the number of configuration contexts. Additionally, we discuss significant test and debug issues of coarse-grain data-path reconfiguration and its implementation trade-offs.

1 Introduction

The data processing for today’s media and telecommunication applications in conjunction with low power requirements can be efficiently provided by highly parallel digital signal processors (DSP). Due to a recognizable trend towards rising implementation and mask costs caused by the reduction of the semiconductor structural size [1], the per-chip cost increases dramatically considering low-production-volume application-specific DSPs. This cost trap can be escaped by flexible architectures adaptable to a wide range of applications also allowing to keep up with future algorithmic advances. Coarse-grain data-path reconfiguration has shown a promising way towards high functional density and flexibility achievements in DSP functional unit design [6]. Targeting a particular application for a highly-optimized reconfigurable DSP architecture, efficiency is more difficult to achieve compared to an entirely hard-wired microprocessor, since hardware reconfiguration introduces an additional dimension to the application development process. The prototyping
of reconfigurable microprocessors therefore introduces additional complexity because it is not possible to consider the functional unit behavior as given and unchangeable. The basic idea is to combine the flexible and well-known instruction flow oriented programming model with the adaptability of the underlying hardware to achieve performance gains, improve the efficiency of computation or the cost/performance ratio. The prototyping framework introduced in this paper attempts to find a systematic approach to design and test of coarse-grain reconfigurable data-path-enhanced DSP cores.

The rest of this paper is organized as follows. Section 2 outlines prototyping-related design goals for our considered class of embedded coarse-grain reconfigurable DSP architectures. In section 4, we describe the ARRIVE prototype architecture, which consists of an ARM7 microprocessor coupled tightly to coarse-grain reconfigurable functional units. Section 3 introduces the prototyping concept and environment. Section 5 describes the results achieved. Section 6, finally, draws some conclusions and shows potentials for further work.

2 Reconfigurable DSP Architectures

For an efficient prototype framework implementation, the definition of a unified design space model is important, since reconfigurability always introduces an additional dimension into the design process as compared to an entirely hard-wired microprocessor. The design space model shown in Fig. 1 reflects the relationship between chip area, production volume, application flexibility and performance issues already mentioned in section 1. Our
main application focus is the low-power embedded DSP domain where a high degree of parallelism is required to solve performance problems at comparatively low clock speeds. The application specificity normally found in dedicated microprocessors or ASICs in this domain can be partially eliminated through reconfiguration. An implementation of this feature essentially trades additional hardware effort and reconfiguration time for a gain in flexibility. The programming model is more complex and raises the demand for new software development tools. The hardware-related parameters showing the suitability of a particular coarse-grain reconfigurable DSP architecture can be summarized by the following four parameters:

**Flexibility** is expressed as the variety of executable functions and its achieved execution performance.

**Code-density** can be characterized as the number of program/configuration bits in relation to the number of controllable pay-operations. A high code density can be achieved when the number of configuration bits is kept low.

**Functional density** is the average contribution of a transistor to the calculation of the computational results. A high functional density can be achieved when the (reconfigurable) functional units are highly-utilized by the application.

**Performance/Parallelism** determines the number of operations per clock cycle. As already mentioned, a coarse-grain reconfigurable unit can exploit more application-level parallelism than common instruction-level functional units.

Finally, it has to be noted that the balance between computation and data transfer has to be assured to avoid bottlenecks during application execution. In this work, we focus on single-core architectures, which may contain programmable and reconfigurable components.

### 3 Prototyping Framework

The prototyping of the reconfigurable microprocessors introduces some requirements to the test environment:

**Flexibility.** The development process often requires frequent changes to the “device under prototyping”, the prototyping engine must be able to change the configuration contexts quickly. This fact sets some constraints to the design of the hardware and software interfaces between target prototype and the evaluation host.

**Scalability.** The amount of logic resources in middle-range FPGA devices does not yet suffice for the implementation of wide (128 or 256 bit) DSP data paths. For this reason the system must allow accurate performance estimates for processors with wider data paths based on the data for 16 or 32 bit.

**Observability.** The current state of the processor must be observable for both the hardwired and the reconfigurable parts. Due to this fact, a hardware interface with additional registers and a software protocol to control this interface are required.
Usually, multi-FPGA engines are used for the prototyping of complex digital systems. Such engines consist of several FPGA or CPLD devices, communication chips, dedicated SRAM and DRAM modules, clock management and interfacing circuitry. The main disadvantage of these boards (in addition to the fact that they are very expensive) is the necessity to split the design over several reconfigurable chips. For the prototyping of embedded DSP cores a single-FPGA board suffices in most cases. For this work Altera’s Stratix EP1S80 DSP Development Board [4] was chosen as the basic equipment because of the high DSP performance of the Stratix FPGA family. The board uses a JTAG interface to configure the EP1S80 FPGA from the host workstation. Although the JTAG interface could be used for debugging the circuitry implemented on the FPGA, it is very complicated to access the internal processor state. Hence, a dedicated hardware component based on the RS232 port of the development board was designed for debugging and control purposes. The corresponding software environment is running on the host workstation and provides all functions required to control the execution flow and system state. A dedicated user interface (Fig. 2) provides the following functionality:

- Load the instructions and the configuration contexts of a particular application
- Start/stop the operation of the core, flush the execution pipelines
- Single stepping/clocking
Figure 3: ARRIVE Architecture Overview

- Read/writemodify the register files, memories and internal states
- Non-intrusive application debugging
- Performance and resource monitoring
- Hardware breakpoints/watchpoints

As the implemented architecture can be viewed from different perspectives (instruction flow, data flow, functional unit interaction, pipeline operation, resource access etc.), the prototyping framework gains access to virtually any particular hardware item via a dedicated debug interface access path. It is obvious that the hardware effort for such a flexible interface might exceed the available resources, thus limiting the functionality of the prototype system. Hence, the set of functions has to be chosen carefully.

4 The ARRIVE Architecture

The ARRIVE architecture extends an ARMv4 RISC core [7] by two coarse-grain reconfigurable functional units (RFUs) and one hard-wired functional unit as illustrated in Fig. 3. These units are dedicated to DSP application acceleration, including support for arithmetic parallelism, data-path parallelism and hardware-based zero-overhead loop unrolling (hardware loops). In detail, the ARRIVE architecture includes a multi-context reconfigurable ALU array (RALU), a multi-context vector load/store unit (VLSU) and a hard-wired control-flow manipulation unit (CFMU). The RFUs share the lower eight mode-
independent registers of the ARM core through a separate read/write port, which provides an efficient datapath-coupling in case of a simultaneous operation of all units. For performance reasons, the RFUs are also connected to a local register file not accessible from the ARM core. To avoid bottlenecks, all local registers can be used in parallel. From the processing model’s view, the ARRIVE architecture can be considered a highly parallel VLIW DSP. With the exception of the hardware loop management in the CFMU, ARRIVE is not an ISA extension. The processing operations of the RFUs are only specified within the configuration contexts. As the number of frequently executed instructions inside inner loops is low, ILP can be easily mapped to a limited number of configuration contexts. The configuration context change is performed cycle-wise according to the ARM instruction pointer. A context assignment table is implemented in the context configuration manager (CCM), which is capable of changing the configuration context on a matching instruction pointer value.

The Reconfigurable ALU Array is composed of coarse-grain context-controlled processing elements (PEs). Horizontal routing is provided through dedicated busses. The inputs and outputs of the PEs are connected to these busses via configurable switches. Vertical routing is maintained only through the PEs. The PEs can be configured to execute 48 different arithmetic, logic or shift functions, including a bypass operation. Additionally, an idle operation is implemented for routing and power-save reasons. Each PE also includes a pipeline data register. A designated carry logic can be utilized to chain the PEs horizontally into an ALU operation over operands of multiples of their processing width. The inputs of the top row are directly connected to the source register bus whereas the outputs of the bottom row interface to the destination register bus. The size of the array is currently assigned - but not limited to - 8×8 8-bit clustered PEs or 4×4 16-bit non-clustered PEs, as it was assumed most suitable for our benchmark applications. Other array architectures may also be advantageous for different application domains.

The Vector Load/Store Unit (VLSU) provides a flexible and scalable interface to dedicated local memories. It enables parallel data load/store of the ARM core registers or the RALU/VLSU local registers through the source and destination register busses. Through special address registers the VLSU supports typical DSP memory transfers including pre-/post-increment/-decrement, write-back and offset address modes.

The Control Flow Manipulation Unit (CFMU) can directly modify the address register of the ARM core, thus influencing the instruction fetch process. Eight dedicated stack locations are available to support the mapping of nested loops with fixed bounds.

The Context Configuration Manager (CCM) matches the value of the ARM address register in the instruction fetch phase with the entries in the context mapping table. Its entries define the relation of ARM PC addresses to the contexts to be activated accordingly. In each clock-cycle, the ARM fetch address register is compared against all CCM address entries simultaneously. In case of a match, the specified RALU/VLSU context numbers are transferred to the RFUs through separate context configuration busses. Hence, the context numbers directly address the RALU/VLSU operations stored in the local configuration memories. The configuration memories are pre-loaded by the ARM core on a per-application basis. According to the ARM7 pipeline, the context activation process is implemented as a triple-stage pipeline, achieving activation of the RFU functions exactly
in the execution-stage of the ARM core-pipeline

Through its separated control-flow-, data-processing- and transfer-oriented reconfigurable functional units, the ARRIVE concept can be considered a load/store architecture, which is inherently compiler-friendly. Thus, the configuration contexts can be obtained by the application of sophisticated code generation methods similar to RISC and VLIW microprocessors [2, 3]. The ARRIVE architecture is implemented as a template model, where the overall data-path width, the size and the operations of the PEs, the layout of the RALU and its cluster size, the VLSU memory size, the number of local registers, the number of contexts and finally the size of the CCM table can be tailored to the application requirements. To obtain an optimal architecture instance for a particular application, we developed a parametrizable VHDL RTL model, which is evaluated using our proposed FPGA-based prototyping framework.

5 Experimental Results

In this section, we discuss the mapping of six significant DSP algorithms onto the ARRIVE prototype architecture. Our benchmark applications include a FIR/IIR filter, an in-place complex radix-2 FFT, a half-rate 16-state Viterbi decoder, a turbo decoder (half-rate, two
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ARM 8-bit PE</th>
<th>ARM 16-bit PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIR Filter</td>
<td>9.5</td>
<td>0.25</td>
</tr>
<tr>
<td>IIR Filter</td>
<td>59.7</td>
<td>2</td>
</tr>
<tr>
<td>FFT Radix-2</td>
<td>99608</td>
<td>5100</td>
</tr>
<tr>
<td>Viterbi Decoder</td>
<td>212</td>
<td>11</td>
</tr>
<tr>
<td>Turbo Decoder</td>
<td>4648</td>
<td>216</td>
</tr>
<tr>
<td>DCT</td>
<td>5213</td>
<td>136</td>
</tr>
</tbody>
</table>

Table 1: Benchmark Performance Overview

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CCM Entries</th>
<th>CFMU Entries</th>
<th>RALU Contexts</th>
<th>VLSU Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIR Filter</td>
<td>13</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>IIR Filter</td>
<td>11</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>FFT Radix-2</td>
<td>19</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>FFT Radix-2 16-bit</td>
<td>26</td>
<td>6</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Viterbi Decoder</td>
<td>29</td>
<td>5</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Turbo Decoder</td>
<td>29</td>
<td>5</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>DCT</td>
<td>51</td>
<td>2</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 2: ARRIVE Context and Mapping Table Allocation

parallel concatenated convolutional codes with 8 states) and a DCT [8, 9, 10, 11, 6]. Except for the turbo decoder, all algorithms are implemented on an RALU composed of 64 8-bit PEs organized as four clusters, each containing an array of $4 \times 4$ PEs. To provide efficient data transfer capabilities, two VLSUs with separate 32-bit data-busses to local memories are available. For comparison, an alternative non-clustered $4 \times 4$ 16-bit variant of the RALU was evaluated. The FPGA synthesis results of the 8-bit RALU based ARRIVE model indicate a total requirement of 30270 LUTs, whereas the 16-bit ARRIVE model consumes 19570 LUTs. These values include the expenses for the ARM microprocessor (approx. 4050 LUTs), but do not consider the additional hardware effort for the multipliers since they are mapped entirely to the dedicated DSP blocks. Although the analysis of the overall critical RALU path indicates an achievable clock-frequency of 12 MHz, it is not completely relevant for our considered benchmark applications, which can be run at 20 to 25 MHz. Further clock-frequency enhancements can be achieved through an increased RALU pipeline depth. The logic cell consumption and clock frequency estimation of the ARRIVE components was determined using Altera’s Quartus II software.

Some detailed information on the overall RALU utilization is provided in Fig. 4, whereas the number of required execution cycles compared to a stand-alone operation of the ARM microprocessor can be obtained from Table 1. In Table 2, we finally draw some attention to the number of allocated configuration contexts and CCM mapping table entries.
6 Conclusion

In this paper, we have presented a prototyping framework for the ARRIVE architecture, which is an extension of a standard RISC microprocessor. Spatial computation is performed by tightly-coupled context-controlled coarse-grain reconfigurable functional units, which are synchronized to the microprocessor core in a pseudo-VLIW manner. To examine the utilization of this architecture and to prove the increase in ILP, we analyzed the mapping of six significant DSP algorithms.

Due to application constraints, we encountered a moderate utilization of the arithmetic, logic and shift units in the RALU and a significant count of routing operations. On the contrary, the memory bandwidth shows good utilization. It is interesting to note that at least in the DSP domain an 8-bit RALU architecture does not pay off in contrast to its 16-bit counterpart. When referring to 16-bit PEs, the achievable performance gain is definitely higher than the increase in logic consumption when compared to a pure ARM microprocessor, resulting in a higher functional density. This aspect is also valid for the comparison to state-of-the-art VLIW processors, due to the decrease in instruction memory size, simplification of the instruction decoder and replacement of special-purpose hard-wired functional units with a horizontally and vertically scalable RALU. The vertical enlargement of the RALU is, however, not desirable as it complicates the datapath-routing, pipelining and scheduling process dramatically.

It has also turned out that the debug interface consumes only 10 to 15% of the total hardware resources to provide access to all register-files, data/instruction memories and configuration contexts. Most of the consumed logic cells are dedicated to flexible register file access. It is interesting to note that the implementation of a configuration context read-back mechanism is very expensive in terms of logic cell consumption since a large amount of multiplexers/busses is required in this case. Thus, a configuration context read-back capability is not available within the current framework.

Having shown the advantages of reconfigurable architectures by the example of the ARRIVE prototype, some further steps are needed. As the results obtained from the FPGA prototype are imprecise in some manner, we plan to re-evaluate the chip area consumption as well as the achievable clock frequency more accurately by an ASIC circuit synthesis in the near future.
References


