RAPID PROTOTYPING OF MULTIMEDIA CHIP SETS

Mohamed Ben-Romdhane *Marius Vassiliou* *Lan-Rong Dung*

Rockwell Science Center
1049 Camino Dos Rios
Thousand Oaks, CA 91360

ABSTRACT

We have developed a rapid prototyping environment for Multimedia applications. The design environment is based on efficient hardware and software reuse, abstraction, design parameterization, and automation. The design methodology maintains a flexible boundary between hardware and software by eliminating hardware fabrication from the design loop. A reusable Hardware/Software library for video compression has been developed to support the design methodology. We present a case study involving the design of an H.263-based video decoder. This case study illustrates the efficiency and flexibility of the design methodology.

1. INTRODUCTION

This paper describes a rapid prototyping environment for multimedia applications. The need for such a tool is triggered by a shrinking time-to-market window and a rapidely evolving and highly competitive market. The rapid prototyping tool is based on virtual prototyping, abstraction, design reuse, parameterization, and automation. Virtual prototyping allows us to test and validate the system as a simulatable model prior to fabrication. Abstraction is implemented through encapsulation and reuse. It relieves the designer from dealing with low level details. Design reuse is considered at the functional level. It provides ease-of-use and contributes significantly to the shortening of design turnaround times. Parameterization introduces the flexibility required for retargeting reusable components and/or design architectures to a change in the specifications. Automation is introduced, when possible, to help shorten the design process and lessen the design complexity, and to assist in investigating a larger number of design alternatives.

2. OVERVIEW OF DESIGN METHODOLOGY

The design methodology discussed here consists of four abstraction Levels: (1) Algorithm Development, (2) Performance Modeling, (3) Virtual Prototyping, and (4) Low-level Implementation. Figure 1 illustrates these design phases. We have added Levels 2 and 3 to our conventional design methodology in order to bridge the gap that exists between Algorithm Development and Low-level Implementation. The design methodology begins at the Algorithm Development level. Here, the performance of the target algorithm is validated. The algorithm is represented as a

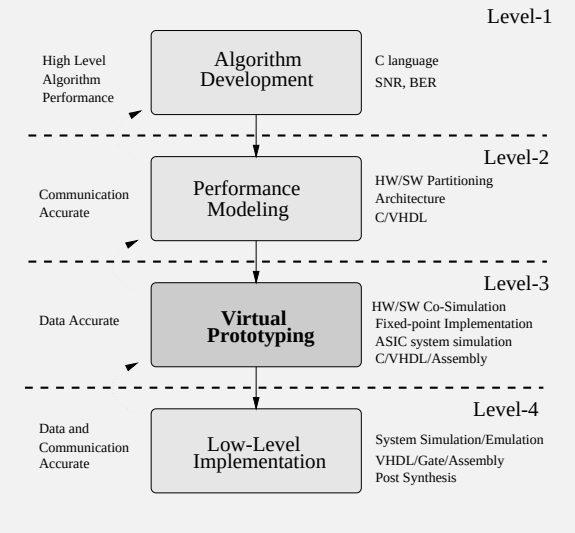

Figure 1: General View of the Design Methodology

block diagram of communicating Macro-Blocks (Software Tasks) written in C. The next design phase, Performance Modeling, benchmarks the execution of the algorithm over a set of new and existing hardware architectures. This design phase assists the designer in making early implementation decisions such as: Hardware/Software partitioning, choice of processor, memory type and size, and communication protocol. Functionality refinement and implementation issues (i.e., Hardware/Software Cosimulation, Fixed-point optimization) are addressed in the subsequent design stage, Virtual Prototyping. This abstraction level, which is the main focus of this paper, captures the necessary detail about the system, and validates its implementation prior to fabrication. By eliminating hardware fabrication from the design loop, Virtual Prototyping maintains a flexible boundary between hardware and software. This results in an efficient Hardware/Software integration that can easily and quickly accommodate late design changes. The final design phase before fabrication is the Low-Level Implementation phase. This is the post synthesis stage where the final validation of the system is done through gate-level simulation and emulation.

3. PERFORMANCE MODELING

This is a front-end design phase that maps algorithms to candidate architectures, and allows early decisions about architecture tradeoffs, design feasibility, validation of architectural concepts, and implementation details. This design phase is described in detail in [2]; This section provides only a brief outline. Experience shows that Performance Modeling can determine over 70% of the final product, and its manufacturing and maintenance costs. It allows designers to start with an executable specification and derive rapid architecture exploration, requiring the least amount of functionality. This is made possible by a performance modeling library of Processors, ASICs, Memory modules, DMA units, and communications components (buses, PCI, VME, etc.). In the Performance Modeling phase, the designer can effect quick architecture exploration, Hardware/Software partitioning, communication modeling and tuning, and task scheduling.

The Performance Modeling phase captures system performance results, such as:

- processor utilization,
- memory size,
- bus congestion and activity,
- interrupt overhead,
- Gantt chart,
- efficiency of partitioning and scheduling.

This design phase helps designers make rapid architectural tradeoffs and decisions about the processor of choice, the tasks that will run in hardware versus those that will run in software, the memory size, the DMA speed, the bus protocol, and other important implementation details. The current practice for performance evaluation is a pen-and-paper approach that can take up to several months without providing the required accuracy or tradeoff flexibility. Also, a pen-and-paper approach, cannot predict issues such as interrupt and bus arbitration overhead. Our proposed methodology runs communication-accurate simulations with the least amount of functionality. It provides a valuable and accurate insight to architectural exploration and hardware/software partitioning.

4. VIRTUAL PROTOTYPING

The Virtual Prototyping design phase relies on core reuse of hardware and software blocks. We have developed a hardware and a software library of reusable multimedia cores to support the Virtual Prototyping Engine (VPE) described in Figure 2. Using the VPE, a Virtual Prototype of the application is quickly assembled by pulling reusable blocks from the available libraries. The hardware library is written in RTL-VHDL, while the Software library is in C. The partitioning of hardware and software tasks is derived from the previous design stage: Performance Modeling. Because our main goal is to develop Multimedia Chip Sets, the focus of the VPE is to: (1) exercise the hardware (Chip Sets) in the system, (2) tailor its functionality and datapath to the specifications of the application, (3) derive its exact cost,

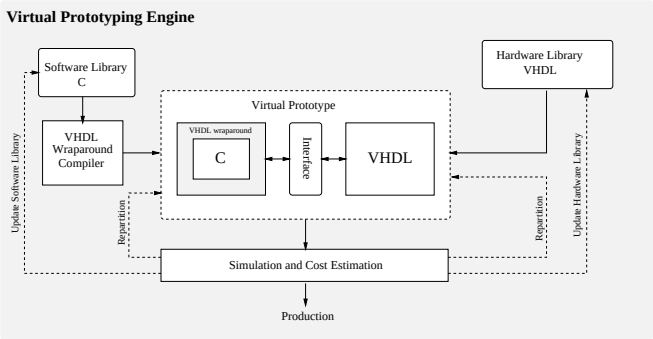

Figure 2: Virtual Prototyping Engine (VPE)

and finally (4) to synthesize it into Application Specific Integrated Circuits (ASICs). The simulation of a Virtual Prototype is run under a unified VHDL simulator. Software tasks written in C are automatically wrapped around in VHDL to run under the same VHDL simulator (e.g. QuickHDL). The wraparound compiler uses the foreign language interface of the VHDL simulator. This approach results in fast Hardware/Software (VHDL/C) co-simulation. After verifying and tuning the Virtual Prototype to meet the performance/cost specifictions, the hardware tasks are synthesized and the software tasks are compiled to the target processor.

4.1. REUSABLE LIBRARY

The hardware library is designed to take advantage of those VHDL features that make reuse easy without sacrificing performance. The reusable components are designed at the RTL level with a parameterization of all datapath aspects. Reusable components are characterized by their domain, structure, and feature. The domain specifies the target algorithm, the structure defines the architecture, and the feature states the variations of implementations for a fixed domain and structure [1].

The reusable library is designed to quickly prototype video compression codecs. In order to eliminate errors due to code modification by designers, the components are considered as black boxes accessed only through their entities. The library also provides the necessary information about the functionality, the implementation cost, and the use of the components in the system. Figure 3 illustrates an example of a parameterized ASIC core. An original image is run through a parameterized Discrete Cosine Transform (DCT) model. The transformed image is then decode by a parameterized Inverse Discrete Cosine Transform (IDCT) model. The DCT/IDCT models are then tuned to optimize the fixed-point perfomance of the algorithm. Figure 3 shows the improvement in quality of the decoded picture when the data precision is increased from 6 to 8 bits. The DCT/IDCT models can also be tuned for coefficient, accumulation, scaling, and internal RAM precisions. The library also provides a variety of DCT/IDCT architectures for a wide range of image and video compression algorithms.

Figure 4 illustrates the architecture of a high perfor-

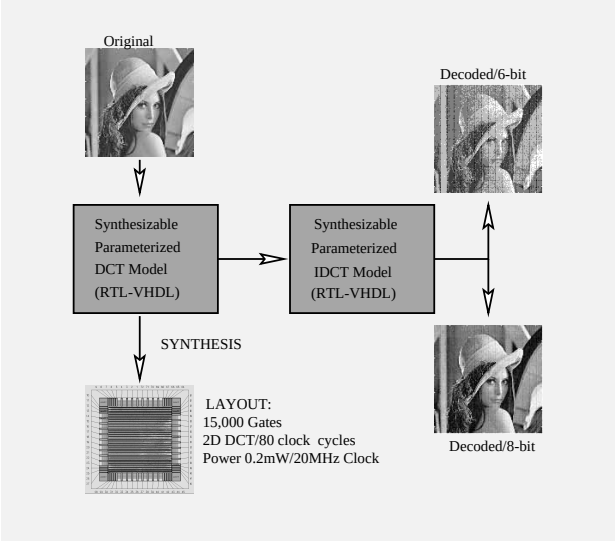

Figure 3: Parameterized ASIC Cores

mance DCT reusable component that executes a two dimensional 8x8 DCT block in 80 clock cycles. The original 8x8 DCT matrix is decomposed into two 4x4 matrices (M1 and M2). The architecture is based on three units: Execution Unit (EU), RAM Unit (RU), and Address Generator Unit (AGU). The computation of the DCT is executed in two phases. In the first phase, the data are read in pairs from the input ports and fed to the pipelined EU (a new pair of data is accepted every clock cycle). The output of the EU that implements matrix multiplications M1 and M2 is written to the 2-ported RU (based on the WRITE addresses generated by the AGU). In the second phase, the data stored in the RU are read in pairs (based on the READ addresses generated by the AGU) and recycled to the EU. The output of the EU is then directed to the output ports of the chip to yield the DCT of the original data. This DCT component can be tailored to different applications through its tunable datapath precision.

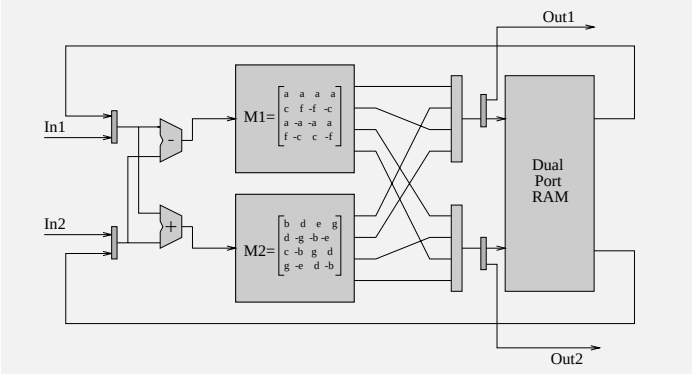

Figure 4: DCT Architecture

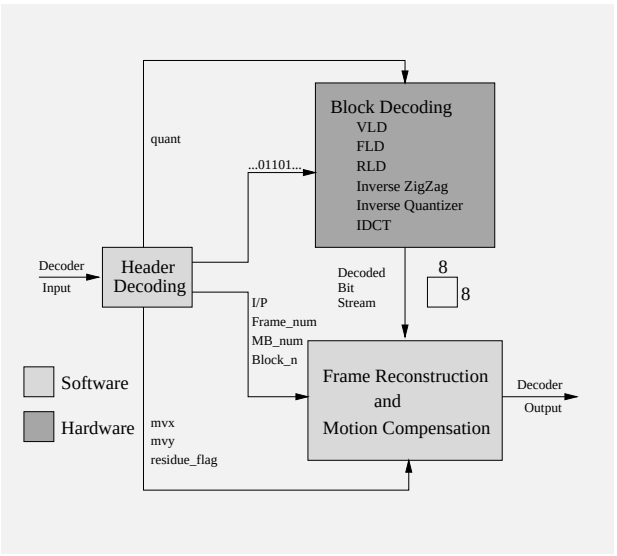

Figure 5: Task partitioning for the H.263 decoder implementation.

4.2. DESIGN ENVIRONMENT & CASE STUDY

Figure 6 illustrates the VPE. We used this design environment to quickly prototype a variety of video compression algorithms. The case study considered in this section illustrates the prototyping of a video decoder based on the H.263 standard. The partitioning of the decoder is illustrated by Figure 5, and is derived from the Performance Modeling design phase (see Figure 1). The block decoding tasks are allocated to hardware due to their computational intensity. On the other hand, H.263 protocol oriented tasks are allocated to software. The VPE of Figure 6 provides a configurable plug-and-play environment to prototype the H.263 video decoder based on the partitioning of Figure 5. The Graphical User Interface provides a flexible way to choose the test bench and to set the hardware and software design parameters. Using this flexibility, one can quickly tune the Virtual Prototype to investigate performance-cost trade-offs. Figure 6 shows the improved quality of the decoded sequence on the right. This improvement is obtained at the cost of increasing the internal precision of the decoder by 2 bits, and decreasing the scaling factor inside the DCT by a single bit.

5. SUMMARY

This paper describes a rapid prototyping environment for Multimedia chip sets. The design methodology relies on efficient hardware and software reuse, abstraction, design parameterization, and automation. The design methodology eliminates hardware fabrication from the design loop and provides a flexible environment to model and tune Virtual Prototypes. A reusable Hardware/Software library for video compression was developed to support the design methodology. The design of a fully-functional video

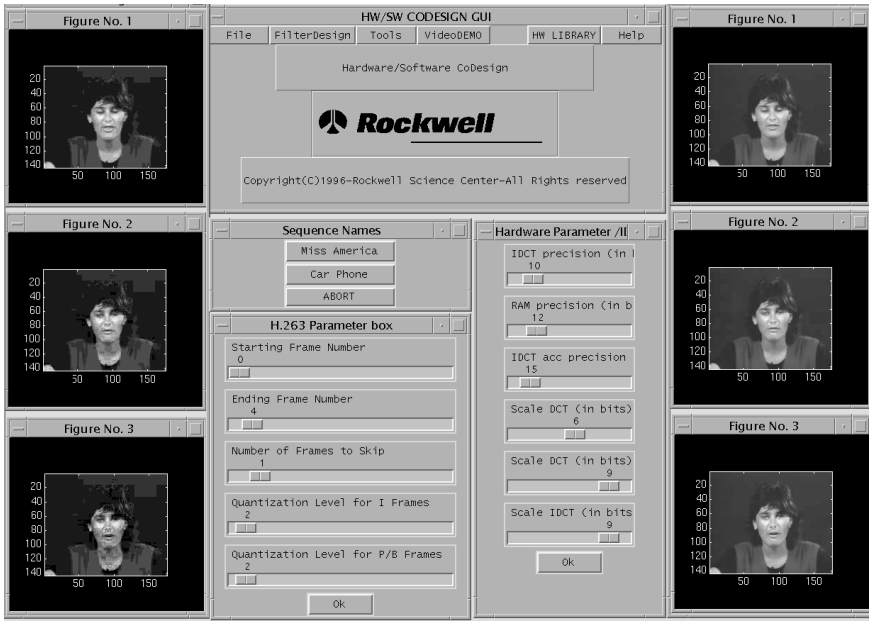

Figure 6: Rapid Prototyping Environment

coder/decoder within the proposed environment has demonstrated the efficiency and flexibility of the approach.

6. REFERENCES

1. Mohamed Ben-Romdhane, Vijay Madisetti, and John Hines, "Quick-Turnaround ASIC Design in VHDL, Core-Based Behavioral Synthesis", *Kluwer Academic Publishers (KAP),* June 1996.

2. Lan-Rong Dung, Mohamed Ben-Romdhane, and Marius Vassiliou, "IP-Based Architecture Exploration," *DesignCon 99*, Feb. 1-4, Santa Clara, CA.