Design and Analysis of Heterogeneous DSP/FPGA Based Architectures for 3GPP Wireless Systems*

Michael C. Brogioli, Manik Gadhiok and
Joseph R. Cavallaro
Electrical and Computer Engineering
Rice University
Houston, TX 77005
{brogioli, gadhiok, cavallar}@rice.edu

ABSTRACT

This paper shows how iterative hardware/software partitioning in heterogeneous DSP/FPGA based embedded systems can be utilized to achieve real-time deadlines of modern 3GPP wireless equalization workloads. By utilizing a well defined set of application partitioning criteria in tandem with SOC simulation tools, we are able to show a greater than six fold improvement in application performance and ultimately meet, and even exceed real-time data processing deadlines.

Keywords

Real Time Systems, Embedded Systems, FPGA, System Partitioning, Wireless Applications

1. INTRODUCTION

Often heterogeneous architectures are used in the embedded domain, consisting of one or more DSP cores, simpler microcontroller, and possible FPGA based compute engines providing low cost, high performance, reconfigurable functionality [6]. The DSP provides the flexibility and programmability, while the FPGA provides the performance of an all hardware solution.

Next-generation mobile wireless communications systems are a driving force behind the innovation in high–performance, low–cost hardware. Heterogeneous, system-on-chip architectures are being developed in order to meet the real–time requirements and end–user performance expected from these ubiquitous mobile devices. In designing these systems, task partitioning and hardware/software co–design present a challenge in alleviating computational bottlenecks in a traditional all software DSP based solution.

This paper shows how hardware/software partitioning between programmable DSPs and FPGA compute elements can be used to achieve real-time processing deadlines in modern 3GPP wireless equalization workloads. By using a software based SOC simulation environment, and well defined application partitioning criteria, we are able to easily prototype multiple hardware topologies in tandem with iterative application partitioning, and show impressive preliminary gains and trends in system performance in using heterogeneous DSP/FPGA system-on-a-chip architectures.

*This work was supported in part by Texas Instruments, Inc., Nokia Corporation, National Instruments, Xilinx, Inc., and by NSF under grants EIA-0224458, and EIA-0321266.

2. BACKGROUND

This section discusses the background of hardware/software co-design, and uses of FPGA based computing elements in embedded processing. As a case study, the channel equalization algorithm for next generation High Speed Downlink Packet Access (HSDPA) wireless standard [2] is considered, as well as the real time requirements and computational complexities.

2.1 Hardware/Software Partitioning in the Presence of FPGAs

Hardware/software codesign and system partitioning has been an emerging field for a number of years, and has taken many forms and targeted various types of workloads. In recent years, modern embedded systems have grown to be vastly more complex than a simple microcontroller and a small amount of local memory. This has resulted in a number of open ended problems with respect to how the designer should jointly specify hardware and software partitionings, and how system architects design their systems.

There have also been a number of uses of FPGA based computing elements as an attempt to provide high performance reconfigurable computing at both the fine grained ISA level, and coarse grained task level. Systems such as Chimaera and other fine grained uses of FPGAs have made attempts to provide extensible and reconfigurable ISAs in general purpose processors [7]. Typically these types of systems have tightly coupled a reconfigurable FPGA based ALU that contains a shadow copy of the host processor's register file. Other architectures employing FPGA based computational blocks have worked at a more coarse grained level, and make attempts to offload larger elements of computation in the input application onto an FPGA based array [9]. These systems are typically applied to partition tightly coupled nested loops of otherwise regular computation onto an FPGA, to enhance runtime performance. Rather than partition workloads at the fine grained, or even coarse grained loop level, this work shows how application partitioning at the application task level across multiple FPGAs can help achieve real-time deadlines in computationally bottlenecked embedded systems due to the vast amounts of instruction and data level parallelism that can be exploited by FPGAs versus traditional DSP based solutions.

2.2 Channel Equalization in 3G Wireless Systems

Code Division Multiple Access (CDMA) has been the driving force behind the third generation (3G) wireless cellular technology, and is used as the multiple–access technology of choice by many cellular standards (UMTS, CDMA2000). Extensions to these for data services have been standardized as the High Speed Downlink Packet Access (HSDPA) and its equivalent 1xEV-DV (Evolution for Data and Voice) [1, 2, 3].

This paper assumes the HSDPA downlink transmission with single transmit and single receiving antenna (Single Input Single Output: SISO system). The chip rate of 3.84 MChips/sec is assumed [3] which determines the real-time processing requirements on the receiver end. Current research has indicated that channel equalization is the most complex part of the receiver baseband signal processing chain and continues to be a bottleneck for both SISO and MIMO CDMA-based systems [8, 11, 10].

3. SOC SIMULATION INFRASTRUCTURE

All simulations and DSP/FPGA system-on-a-chip (SOC) prototyping was done using the Spinach SOC simulation environment for prototyping heterogeneous embedded architectures [4, 5]. Spinach is a library of composable, user configurable software modules for rapidly prototyping simulators for embedded SOC architectures. Spinach is built upon the Liberty Simulation Environment (LSE) developed at Princeton University, creating a SystemC like infrastructure where all module communication is transaction based at the bit–level, on clock cycle boundaries within the simulated system. Spinach is intended for modelling architectures that include heterogeneous computing environments, as well as varying system topologies and timing semantics.

In designing an SOC architecture with Spinach SOC, users prototype the system with a library of existing simulator software modules that have a 1:1 mapping to hardware components. The library of simulator module building blocks contains bit-true, cycle accurate Texas Instruments DSPs, MIPS microcontrollers, and interconnect modules such as busses, bridges and crossbars with user definable bandwidth, throughput latencies, and arbitration policies. Memory modules such as SRAM, DRAM, memory controllers, caches and cache controllers are user configurable, and simulate bit true cycle accurate contents for processors, instruction/data memory, etc. FPGA modules are designed such that clock rates, gate counts and functionality are user definable. FPGA modules have support for internal dual ported RAM arrays that can be targeted by intelligent DMA engines for data transfer. While the functionality implemented by an FPGA module is user definable, timing information and gate counts can be verified on real hardware, and configured via high level parameters at runtime. This functionality is discussed further in Section 5.

Intra–module communication between simulated hardware components occurs over abstracted hardware-like module I/O ports, upon discrete clock cycle boundaries at the bit level. For instance, in a DMA transaction from local on-chip data memory to FPGA dual ported RAM arrays, transactions on the simulated memory bus occur st 32/64/128/256 bit resolution upon discrete clock cycle boundaries, where one device attached to the bus wins arbitration each clock cycle and transmits a maximum of the aforementioned bit width data. All data is coupled with timing information, and thus overall system performance between DSP and FPGA for a given system topology can be quantitatively measured. Further information on usage, functionality, and model of computation for Spinach can be found in [4, 5].

4. METHODOLOGY

When partitioning software in a heterogeneous system comprised of reconfigurable FPGA compute engines, programmable DSPs or programmable host processors, a number of criteria must be evaluated to determine which application level task should execute in software on the host processor or in hardware on FPGA.

4.1 Hardware/Software Partitioning Criteria

Spatial locality of data and the ability to access data efficiently is of great importance to performance in offloading to hardware based FPGA implementation. DMA data transfer times can effect performance, as well as data layout in memory and possible preemptive reordering. DMA overhead penalties versus computational gains must be accounted for. Signal processing applications also contain large amount of ILP (instruction level parallelism) and DLP (data level parallelism). ILP/DLP can exceed the DSPs compute and I/O resources creating software bottlenecks. FPGAs coprocessors can alleviate this with large numbers of parallel functional units and large local block RAM arrays for local data storage and computation. Finally, algorithms that are implemented in FPGA are often computationally intensive, exploiting ILP and DLP beyond that of the host DSP. While this can alleviate computational bottlenecks in the system, the programmability and flexibility of a software solution is lost. Often it may be desirable to keep application functionality in software on the DSP, for the sake of programability and flexibility at the cost of performance.

4.2 Channel Equalization Workload

The workload for the mobile receiver's equalization application can be decomposed into multiple tasks consisting of: channel estimation, covariance matrix computation and making matrix into circulant matrix form, Fast Fourier Transform and Inverse Fast Fourier Transform, Fininte Impulse Response (FIR) Filtering, and Despreading/Descrambling of the received signal to recover user information bits.

The channel estimation block takes in the received data and generates channel estimates based on the training sequence. This estimate drives the covariance matrix computation block, which generates the covariance matrix and approximates it with a circulant matrix. The FFT is then applied to the channel estimates and to the first column of the circulant matrix. Reordering is performed on the FFT output to compute the frequency–domain representation of the filter coefficients. Next, an Inverse FFT is performed on these values and the resulting filter coefficients are used for filtering the received data. Finally, despreading and descrambling are done to compute the user information bits. In partitioning this workload across either software programmable DSP or across configurable FPGA based computation engines, we consider partitioning at the granularity of these tasks. Channel conditions can change dramatically from slow to fast fading environments with differing numbers of channel paths, and thus different algorithms for channel estimation and modifications of each algorithm should be employed [8]. Due to this flexibility, channel estimation is to remain in a programmable software based DSP implementation for these initial case studies, even though there exists significant parallelism in candidate algorithms.

4.3 System Modelling

Table 1 lists SOC system parameters used in simulation. The host DSP is the Texas Instruments TMS320C6201 fixed point programmable DSP, operating at 200MHz. There are 256–bit on-chip busses to program and data memory, with 64KB of addressable content each for these experiments. On–chip data memory

controller also performs routing and arbitration of memory references to peripheral components consisting of DMA engines, memory mapped registers (MMRs), FPGA based compute engines and other peripherals. Peripheral busses are 32–bits with uniform clock rate. Bridging logic is used to glue 256-bit memory busses to 32-bit peripheral busses, performing bus packet segmentation/assembly at minimum single cycle throughput latency per 32-bit bus transaction. MxN crossbars are stackable with round-robin arbitration policy and single cycle throughput latency assuming arbitration winner. Resulting system topology and block diagram are discussed in detail in Section 5.

Simulation Parameters	*Value*
DSP Architecture	Texas Instruments TMS320C6201
System Clock Rate	200MHz
Instruction Memory Bandwidth	256b on–chip
Data Memory Bandwidth	256b on–chip 32b off–chip
FPGA I/O Bandwidth	32 bits per clock cycle
DMA I/O Bandwidth	32 bits per clock cycle bidirectional
Bus Arbitration	Round Robin, single cycle minimum throughput latency

Table 1: Baseline Simulation Parameters

Data transfers to and from the FPGA compute fabrics are performed via the on-chip DMA engines. DMA engines are programmed via MMRs with host DSP control and synchronization management. It is these DMA engines, operating at 32-bit data width resolution that perform high speed block data transfers from local DSP data memory to the dual ported RAM arrays found locally on the FPGA fabrics. Data transfers occur on discrete clock cycle boundaries, at the resolution of the on-chip DMA engines. Each 32-bit packet must pass through appropriate on-chip crossbar and bridging logic with associated latency and bus arbitration effects.

The channel equalization firmware, as described in Section 2.2 was all compiled using Texas Instruments Code Composer Studio Version 2.10.0, at level three optimization with aggressive inlining and loop unrolling used. All computation in the channel equalization workload was optimized to be 16–bit fixed point, in effect optimizing the workload for DSP performance by maximizing the functional unit utilization of the TMS320C6201 architecture. While performance critical kernels are often coded in assembly, low level optimized C code was used in these studies. Though performance of assembly may be slightly better in some cases, the overall trends in performance gain are the same due to bottlenecks in DSP computational resources rather than software efficiency.

When partitioning the application across one or more FPGAs, application functionality is manually offloaded to FPGA simulation modules and implemented by the user. Timing information and gate counts are obtained via prototyping in hardware, or via the use of C to RTL offline tools. FPGA compute functionality in simulation is then modelled via low level C code modelling RTL behavior within the FPGA simulator module. Timing and latency information is set via high level user parameters in simulation, as automated RTL to C code migration into the simulation infrastructure is not yet supported.

5. RESULTS

Table 2 shows the runtime profiling data for the workload discussed in Section 4.2, executing on the base case single TMS320C6201 DSP. Based on the data in Table 2, and the criteria mentioned in Section 4.1, FIR Filtering, FFT, and Despread and Descramble computations were chosen to be offloaded to FPGA implementation. Channel Estimation, as mentioned in Section 4 was not considered due to the desire for software flexibililty.

Program Task	*Percentage Runtime*
Channel Estimation:	7.06%
Covariance Matrix:	0.34%
Circulation of Covariance Matrix:	0.19%
FFT:	9.90%
IFFT:	4.06%
FIR Filtering:	73.31%
Despread/Descramble:	3.97%

Table 2: Application Profile Data: All Algorithmic Parts Executed on DSP

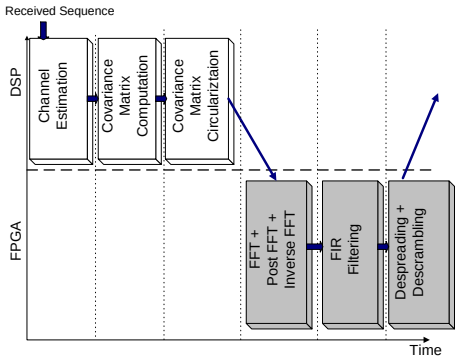

Figure 1: DSP/FPGA Partitioning

Figure 1 illustrates the major computational blocks in the channel equalization algorithm as previously depicted in Section 4.1 now partitioned to execute either in software on the host DSP, or in hardware via an FPGA based implementation. This partitioning was achieved via an iterative hardware–software design approach in which the input application was profiled and analyzed, and various bottlenecks in the application were iteratively offloaded to an FPGA based implementation.

Figure 2 illustrates the final SOC topology modelled in simulation whereby FIR Filtering, FFT/IFFT and Despread/Descramble are offloaded to FPGAs with all other functionality existing in software on the host DSP. While a single FPGA could be used to implement multiple blocks of computation, we chose to model the worst case scenario and maximum data transfer requirements by putting each block in a separate FPGA. Table 3 shows the FPGA parameters for cold start computational latency assuming data residing in local FPGA RAM arrays, as well as the best case DMA data transfers across chip interconnect. DMA transfer times are often worse during program runtime due to non-deterministic conflicts in crossbar arbitration policies for bus access.

Figure 5 shows the clock cycle runtimes of processing one incoming frame of data for each hardware/software partitioning mod-

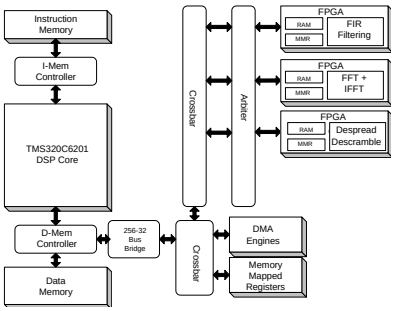

Figure 2: SOC Block Diagram

FPGA Coprocessor	*Compute Cycles*	Data Transfer Cycles
FIR20	20*512 samples	38
FIR35	35*512 samples	38
FFT/IFFT	128	128
DD	1536	276

Table 3: FPGA Cold Start Compute Latencies and Best Case Data Transfer Times

elled. We can see that two implementations of FIR filtering in FPGA were chosen, one with increased parallelism and lower compute latency and one with a longer cold start compute latency. Looking at this figure, it can be seen that as increasing amounts of computation are offloaded to FPGA, performance gains increase. In the fully partitioned case with FIR Filtering, FFT and Inverse FFT as well as Despreading Descrambling offloaded, a 6.21 fold increase in application performance is obtained.

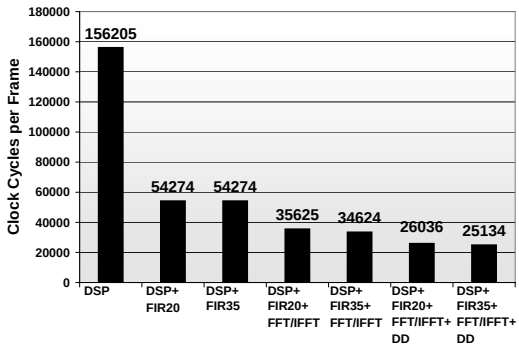

Figure 3: Normalized Runtime vs. System Partitioning

To achieve 3.84MChips/sec data rates of the HSDPA standard, 512 data-chip frames must be processed every 260 ns, or 26104 clock cycles (at 200MHz clock rate) for uncoded base-band processing, which entails channel estimation, covariance matrix computation and circularization, FFT + IFFT, FIR Filtering and Despread / Descramble. In the fully partitioned case, we can see that the system now more than exceeeds the real-time computational requirements to achieve HSDPA data rates. Furthermore, over 40% of the total runtime is spend with the DSP in idle polling mode, waiting for FPGA computation or data transfers to complete.

6. CONCLUSIONS AND FUTURE WORK

This paper shows how iterative hardware/software codesign can be an effective form of designing heterogeneous DSP/FPGA based embedded architectures for meeting real-time system deadlines in modern 3GPP wireless systems. By following a well defined set of partitioning criteria, and iteratively partitioning application software between programmable DSP implementations and hardware based FPGA implementations, significant improvements in real-time system performance can be achieved. As a case study, modern 3GPP wireless channel equalization workloads are analyzed. Greater than 6.2x improvements in performance are obtained over the traditional DSP based software implementation and real-time system deadlines are met and data rates are more than exceeded.

7. REFERENCES

[1] 1xEV-DV Evaluation Methodology (Rev.26). Third Generation Partnership Project Two (3GPP2), May 2001.

[2] 3GPP Technical Report 25.848, Physical layer aspects of UTRA High Speed Downlink Packet Access, version 4.0.0, March 2001.

[3] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and channel coding (TDD), version 4.2.0, March 2002.

[4] M. Brogioli and J. Cavallaro. Modelling heterogeneous DSP-FPGA based system partitioning with extensions to the spinach simulation environment. In *Asilomar Conference on Signals, Systems, and Computers*, October 2005.

[5] M. Brogioli, P. Willmann, and V. Pai. Spinach: A Liberty-Based Simulator for Programmable Network Interface Architectures. In *Languages Compilers and Tools for Embedded Systems 2004*, pages 100–110, June 2004.

[6] T. J. Callahan and J. Wawrzynek. Instruction Level Parallelism for Reconfigurable Computing. In *Proc. 8th Intl. Workshop on Field-Programmable Logic and Applications*, Sept 1998.

[7] J. Cong, Y. Fan, G. Han, A. Jagannathan, G. Reinman, and Z. Zhang. Instruction Set Extension with Shadow Registers for Configurable Processors. In *FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays*, pages 99–106, New York, NY, USA, 2005. ACM Press.

[8] A. de Baynast, P. Radosavljevic, and J. Cavallaro. Chip Level LMMSE Equalization for Downlink MIMO CDMA in Fast Fading Environments. In *Globecom*, November 2004. Accepted.

[9] J. eun Lee, K. Choi, and N. D. Dutt. An Algorithm for Mapping Loops Onto Coarse-Grained Reconfigurable Architectures. In *LCTES '03: Proceedings of the 2003 ACM SIGPLAN Conference on Languages, Compilers, and Tool for Embedded systems*, pages 183–188, New York, NY, USA, 2003. ACM Press.

[10] Y. Guo, J. Zhang, D. McCain, and J. Cavallaro. Efficient MIMO Equalization for Downlink Multi-Code CDMA: Complexity Optimization and Comparative Study. In *Globecom*, volume 4, Dallas, Texas, November 2004.

[11] P. Radosavljevic, J. Cavallaro, and A. de Baynast. ASIP Architecture Implementation of Channel Equalization Algorithms for MIMO Systems in WCDMA Downlink. In *IEEE Vehicular Technology Conference*, volume 3, September 2004.