Reconfigurable System on Chip C-Based Design Methodology

[1] **P. R. Bokde,** [2] **P. N. Aerkewar,** [3] **P. M. Palkar**

[1, 2, 3] Assistant Professor,Electronics Department, PBCOE,
Nagpur (M.S.), India.

Abstract- Reconfigurable system is a promising alternative to deliver both flexibility and performance at the same time. New reconfigurable technologies and technology dependent tools have been developed, but a system-level design methodology to support system analysis and fast design space exploration is missing. In this paper, we present a System C-based system-level design approach..The main focuses are the resource estimation to support system analysis and reconfiguration modeling for fast performance simulation. The approach was applied in a real design case of a WCDMA detector on a commercially available reconfigurable platform. The run time reconfiguration was used and the design showed 40% area saving when compared to a functionally equivalent fixed system and 30 times better in processing time when compared to a functionally equivalent pure software design.

Keywords- **Reconfigurable system, WCDMA, System- C Based.**

1.Introduction

Reconfigurability is becoming an important issue in design of System-on-Chip (So C) because of the increasing demands of silicon reuse, product upgrade after shipment and bug-fixing ability. The reconfigurability is usually achieved by embedding reconfigurable hardware into the system. Any system whose sub-system configurations can be changed or modified after fabrication is called as Reconfigurable System. Reconfigurable computing (RC) is commonly used to designate computers whose processing elements, memory units, and/or interconnections can change function and/or (spatial) configuration after fabrication, prior or during the run-time of a particular program or part of a program. Reconfigurable computing is a computer architecture combining some of the flexibility of software with the high performance of hardware by processing with very flexible high speed computing fabrics like field-programmable gate arrays (FPGAs). The principal difference when compared to using ordinary microprocessors is the ability to make substantial changes to the datapath itself in addition to the control flow. On the other hand, the main difference with custom hardware, i.e. application-specific integrated circuits (ASICs) is the possibility to adapt the hardware during runtime by "loading" a new circuit on the reconfigurable fabric.

The Xilinx [1] and the Alter a [2] provide fine-grain FPGA platforms. They contain embedded processor cores, which make it possible to design a rather complex system in such FPGA platforms. However, the reconfigurability does not come free of costs. The power consumption, reconfiguration latency and related overhead are main concerns to system designers and tools. Reconfigurable computing is intended to fill the gap between hardware and software, achieving potentially much higher performance than software.

2. Related Work

There are several research works such as Garp, MorphoSys, and PipeRench, toward developing novel reconfigurable architectures and associated software development tools. However, these are architecture-centric design frameworks that do not specifically address system-level design issues. In the heterogeneous reconfigurable system that we are studying, the reconfigurable hardware is an add-on processing unit to the system, so we concentrate on how to extend the existing design flow and tools to support the design of RSoC. A few research works focus on the system-level issues of more advanced reconfigurable devices such as multi context devices or the devices that allow a task to be freely allocated at run time. These include context management, task relocation and on-line scheduling. In this work, we focus on more realistic reconfigurable technology, which allows run-time reconfiguration (RTR) but uses only single context devices. In addition, dynamic relocation is not taken into account since routing constraints restrict such flexibility in practice at the moment. Although this looks like a limitation of the reusability of our approach, our models and techniques are not bound to a particular reconfigurable technology and they can be easily extended in the future.

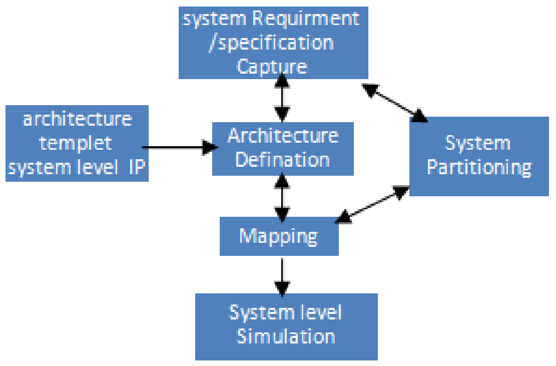
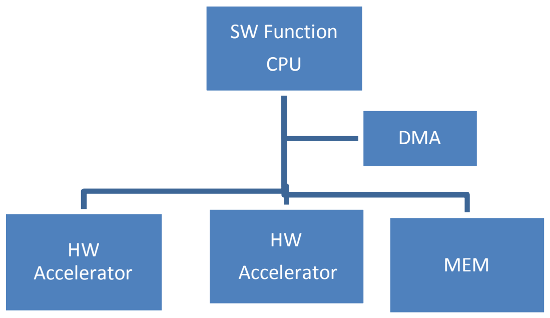

system Requirment /specification Capture
architecture templet system level IP
Architecture Defination
System Partitioning
Mapping
System level Simulation

Figure 1. A generic system-level design flow

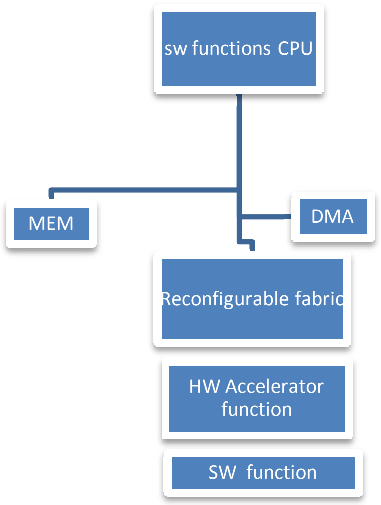
3. System C-based Design Methodology

A generic view of the system-level design flow is depicted in Figure 1. The following new features are identified in each phase when reconfigurability is taken into account :

- **System Requirements** and Specification Capture needs to identify requirements and goals of reconfigurability.
- **Architecture Definition** needs to model the reconfigurable resources at abstract level and include them in the architecture models.
- **System Partitioning** needs to analyze and estimate the functions of the application for software, fixed hardware and reconfigurable hardware.
- **Mapping** needs to map functions allocated to reconfigurable hardware onto the respective architecture model.
- **System-Level Simulation** needs to observe the performance impacts of architecture and reconfigurable resources for a particular system function.



(a)



(b)

Figure 2. (a) An initial fixed SoC architecture and
(b) a modified architecture using reconfigurable Hardware

In the SystemC-based approach, we assume that the design does not start from scratch, but it is a more advanced version of an existing device. The initial architecture is often dependent on many things not directly resulting from the requirements of the application. The company may have experience and tools for certain processor core or semiconductor technology, which restricts the design space and may produce an initial hardware software (HW/SW) partition.

The way that the SystemC-based approach incorporates dynamically reconfigurable parts into architecture is to replace SystemC models of some hardware accelerators, as shown in Figure 2(a), with a single SystemC model of reconfigurable block, as shown in Figure 2(b). The objective of the SystemC-based extensions is to provide a mechanism that allows designers to easily test the effects of implementing some components in the dynamically reconfigurable hardware. The provided supports in the SystemC approach include:

- ➢ Estimation and analysis support for design space exploration and system partitioning [17]
- ➢ Reconfigurability modeling using standard mechanisms of System C and a transformation tool to automatically generate System C models of the reconfigurable hardware.

3.1. Definitions

The terms and concepts specific to the System C based approach used in the rest of the paper are defined as follows:

- ❖ **Candidate Components** :- Candidate components denote those application functions that are considered to gain benefits from their implementation on a reconfigurable hardware resource. The criterion is that the task should have two features in combination: flexibility and high computational complexity

- ❖ **Dynamically reconfigurable fabric (DRCF):** The dynamically reconfigurable fabric is a system level

- ❖ concept that represents a set of candidate components and the required reconfiguration support functions, which later on in the design process will be implemented on a reconfigurable hardware resource.

- ❖ **DRCF component :-** The DRCF component is a transaction-level SystemC module of the DRCF. It can automatically detect reconfiguration request and trigger the reconfiguration process when necessary.

- ❖ **DRCF template**: The DRCF template is an incomplete SystemC module, from which to create the DRCF component.

3.2. Estimation approach to support system Analysis

System analysis is mainly the phase to make HW/SW partitioning and the initial architecture decision. In the design of reconfigurable SoC, system analysis also needs to focus on studying the trade-off of performance and flexibility. The estimation approach focuses on a reconfigurable architecture in which there is a RISC processor, an FPGA-type RTR hardware unit, and a system bus as a communication channel. It starts from function blocks represented using C-language and it can produce the following estimates for each function block: software execution time in terms of running the function on the RISC core, mappability of the function and the RISC core [18], execution time in terms of running the function on the RTR hardware, and resource utilization of the RTR hardware.

The framework of the estimation approach is shown in Figure 3.

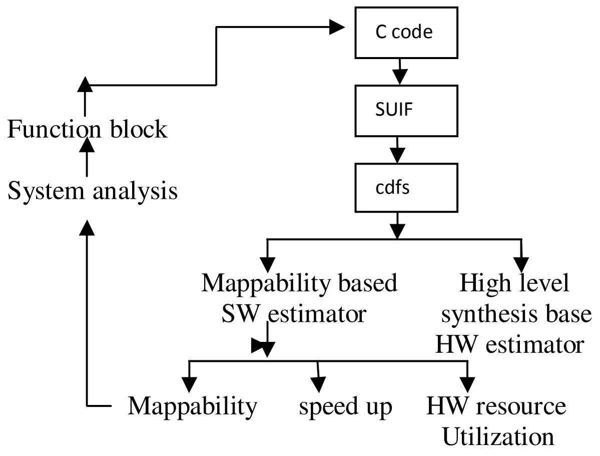


Figure 3. The estimation framework

The starting point is the functional description given in ANSI-C language. The designer decides the granularity of partitioning by decomposing the algorithm down to function blocks. A single function block may then be assigned to SW, RTR hardware or a fixed accelerator. Each of the function blocks will be individually studied and the set of estimation information will be fed into the system-level partitioning phase. In the estimator, a SUIF-based front-end preprocessor is used to extract Control-Data Flow Graphs (CDFG) from the C code. Then some well-known highlevel synthesis tasks are carried out to produce the estimates. A modified version of Force-Directed Scheduling (FDS) is used to estimate the hardware resources required for the computation units and the storage units of the tasks. The current estimator targets a Virtex2-like FPGA in which the main resources are LookUp-Tables (LUTs) and multipliers. The ultimate goal of the estimation approach is to make candidate component selection, which is an application-dependent procedure. In current design framework, the selection is carried out manually based on designers' experience and design constraints. When global resource saving is an issue, the resource estimates are important inputs. Current approach does not include automated tools to support the power and the task dependence analysis.

3.3. Link to Low-Level Design

The low-level design is divided into detailed design and implementation design. The output of the detailed design is the intermediate representation of the system, in which SW is represented as C or assembly code and HW is represented as RTL-HDL code. The implementation is the phase where binary code for SW, bit stream for RTRHW and layout for ASICs are generated. In our approach, automatic code generation for low level design is not provided and designers should manually or using other tools to transform the SystemC representation of the reconfigurable system to low-level code such as for SW implementation and VHDL code for HW implementation. methodology. In our work, we used the Dynamic Circuit Switching (DCS)-based technique to carry out the cycle accurate co-simulation between the functions mapped onto the RTR hardware and the functions mapped onto the static part of the system. A VHDL module for each of the function mapped onto the RTR hardware is manually created. Multiplexers and selectors are inserted after the outputs of the modules and before the inputs of the modules. They are automatically switched on or off according to the configuration status. In the cycle accurate simulation model, the reconfiguration is modeled as pure delay.

4. A WCDMA Detector Case Study

We selected a WCDMA detector design case to validate the SystemC-based approach. We targeted on an RTR-type of implementation and the implementation platform was the VP20FF1152 development board from Memec Design group [23], which contains one Virtex2P XC2VP20 FPGA [1].

4.1. System Description

The whole WCDMA base-band receiver system is depicted in Figure 5. The case study focuses on the detector portion of the receiver and a limited set of the full features were taken into account. The detector case used 384 kbits/s user data rate without handover.

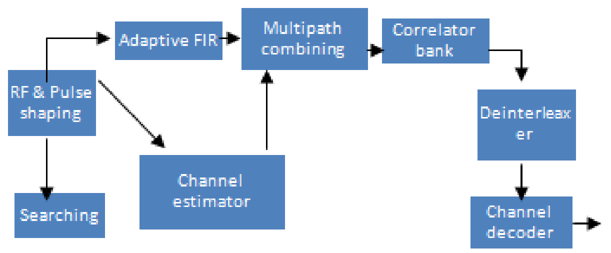


Figure 5. The WCDMA base-band receiver System

The detector contains an adaptive filter, a channel estimator, a multi-path combiner and a correlator bank. The adaptive filter is performing the signal whitening and part of the matched filtering that are traditionally implemented with the RAKE receiver. The channel estimator module calculates the phase references. In the combiner part, the different multi-path chip samples are phase rotated according to the reference samples and combined. Finally the received signal is de-spread in the correlator bank. When compared to traditional RAKE based receiver concepts, this WCDMA detector achieves 1 - 4 dB better performance in vehicular and pedestrian channels.

4.2. System-level Design

The design started from the C-representation of the system. It contained a main control function and the four computational tasks, which lead to a simple system partition that the control function was mapped onto SW and the rest onto RTR hardware. The estimation tool was used first to produce the resource estimates. The results are listed in Table 1, where LUT stands for look-up table and register refers to word-wide storages. The multiplexer refers to the hardwired 18x18 bits multipliers embedded in the target FPGA.

Based on the resource estimates, the dynamic context partitioning was done as following. The channel estimator was assigned to one context (1387 LUTs), and the other three processing blocks were assigned to a second context (1078 + 463 + 287 = 1828 LUTs). This partition resulted in both balanced resource utilization and less interface complexity compared to other alternatives. A SystemC model of a fixed system was then created, which had two purposes in the design. The first was to use its simulation results as reference data, so the data collected from the reconfigurable system could be evaluated. The second purpose was to automatically generate the reconfigurable system model from it via the transformation tool.

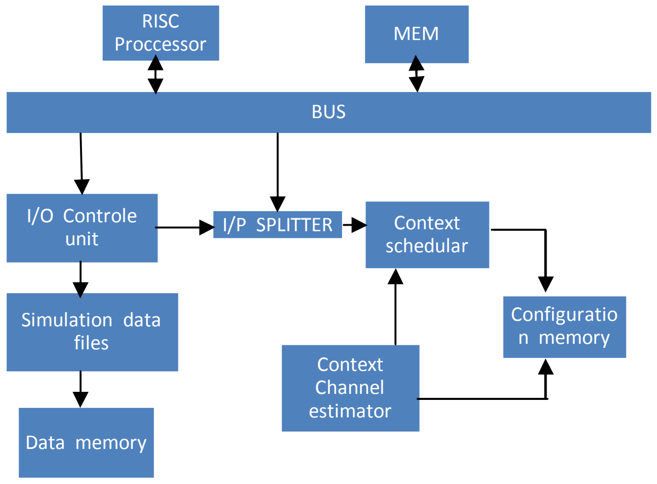


Figure 6. Reconfigurable system model of the WCDMA detector

In the fixed system, each of the four detector function was mapped to an individual hardware accelerator, and pipelined processing was used to increase the performance. A small system bus was modeled to connect all of the processing units and storage elements. The channel data used in the simulation was recorded in text files, and the processor drove a slave I/O module to read the data from the file. The SystemC models were described at transaction level, in which the workload was derived based on the estimation results but with manual adjustment. The transformation tool was used to automatically generate the reconfigurable system model, which is depicted in Figure 6, from the fixed model. There configuration latency of the two dynamic contexts was derived based on the assumption that the size of the configuration data was proportional to the resource utilization, the number of LUTs required. The total available LUTs and size of full bit stream were taken from the Xilinx XC2VP20 datasheet. Some accurate approaches can be used to derive the reconfiguration latency. For example, the latency is related only to the region allocated to the dynamic contexts. In the current work, these have not been studied.

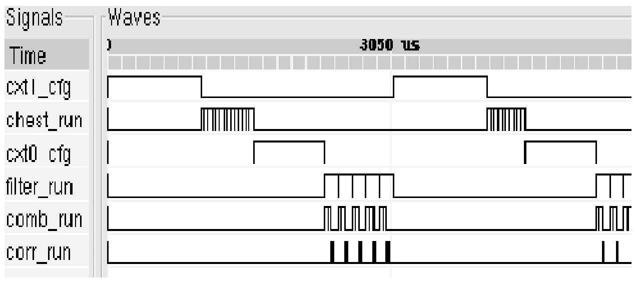


Figure 7. Simulation waveform shows the reconfiguration latency

The performance simulation showed that the system required two reconfigurations per processing each slot of data. When the configuration clock was running at 33 MHz and the configuration bit-width was 16, the reconfiguration latency was 2.73 ms and the solution was capable of processing 3 slots of data in a frame.

4.3. Detailed Design and Implementation

Vendor-specific tools were used in the system refinement and implementation phases. The task was divided into interface refinement, configuration design, SW design, and RHW design. Xilinx modular design flow [1] was used in the low-level reconfiguration design at the implementation phase.

Table 1. HW synthesis results

Functions	LUT	Multiplier
Adaptive filter	553	8
Channel estimator	920	0
Combiner	364	4
Correlator	239	0

One of the two PowerPC cores in the FPGA was used and C code was manually generated out of the SystemC code. The SW/HW interface was mapped to register based communication and the direct memory access was used in the SW side. The RTL-VHDL code was manually generated for the four computational tasks, and the Xilinx IP cores were used for other peripherals. The synthesis results are listed in Table 2. When considering the estimation, the results are over-estimated at about 55% in average. The main reasons are the assumption of fixed length computation and the technique of mapping multiplexers directly to LUTs.

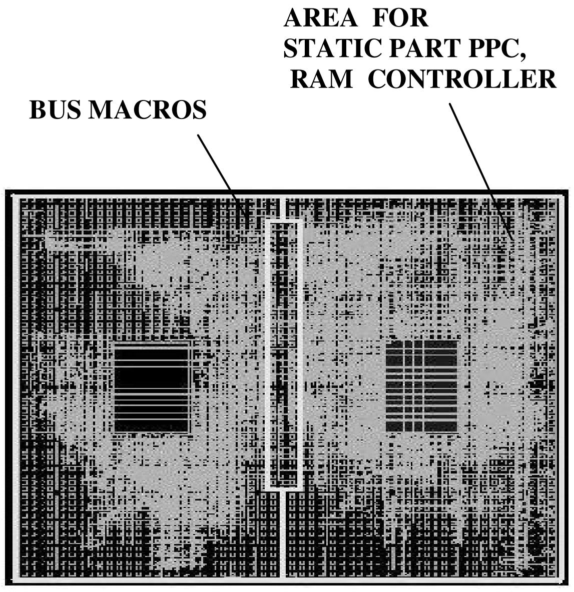


Figure 8. Routed Design of One Dynamic Context And The Static Context

4.4. Comparison with Other Approaches

In addition to the implementation of the dynamic reconfiguration approach, a fixed hardware implementation and a pure software implementation were made as reference designs. In the fixed-hardware implementation, the processing blocks were mapped onto static accelerators and the scheduling task was mapped onto SW that ran on the PPC core. The resource requirements were 4632 LUTs (24% of available resources), 55 Block RAMs (62%) and 12 Block Multipliers (13%). The system was running at 100 MHz. The execution time for processing one slot of datawas 1.06 ms. Compared to the fixed reference system, the dynamic approach achieved almost 50% resource reduction in terms of the number of LUTs, but at the cost of 8 times longer processing time. Everything was running in a single PPC core and data were entirely stored in internal BRAMs.

Table 2. Resource Utilization in Xilinx XC2VP20

	LUT	BRAM	MUL	Reg/bt	PPC
Static	1199	41	0	1422	1
Dynamic	1534	7	12	1855	0
Total	2733	48	12	3277	1

5. Advantages

- The main advantage of the SystemC-based approach is that it can be easily embedded into a SoC design flow to allow fast design space exploration for different reconfiguration alternatives without going into implementation.
- Implementation level is time consuming .
- The potential benefit of using the run-time reconfiguration approach is obviously the significant reduction of reconfigurable resources.
- Compared to a completely fixed implementation, the reduction of LUTs can be up to 50%.
- The main advantage of the approach is that it can be easily embedded into a So C design flow to allow fast design space exploration for different reconfiguration alternatives without going into implementation details.

6. Disadvantages

- The commercial off-the-shelf FPGA platform caused limitations on the implementation of runtime reconfiguration.
- The estimation approach and the DRCF modeling approach have shown their usefulness by providing reasonably accurate results without going into low-level implementation.
- The current approach does not address the power issue ,especially power variance associated with the reconfiguration.

7. Conclusions

The SystemC-based approach to support design of RSoC is described in this paper. The main focus is how to enable fast design space exploration at the system level. The estimation approach to support system analysis, the reconfiguration modeling technique and a tool to support automatic SystemC code generation are provided. Our models work at the transaction level, so the system performance can be quickly evaluated in simulation but without losing too much information details from the system architecture. A design of WCDMA detector design case has been carried out and it has been fully implemented in a real reconfigurable platform. The design case has validated that the SystemC-based approach is very useful in providing the supports to RSoC design at the system level.

References

[1] Xilinx, www.xilinx.com
[2] Altera, www.altera.com
[3] PACT XPP technologies, www.pactcorp.com
[4] QuickSilver Technologies, www.qstech.com
[5] Triscend, “A7 Field Configurable System-on-Chip datasheets”, www.triscend.com (2004)
[6] Motorola, www.motorola.com
[7] A. Pelkonen, K. Masselos, M. Cupak, “System-Level Modeling of Dynamically Reconfigurable Hardware with SystemC”, Proc. IPDPS’03, 2003, pp. 174-181
[8] Y. Qu, K. Tiensyrjä, K. Masselos, “System-level modeling of dynamically reconfigurable co-processors”, Proceedings of the 14th International Conference on FPL (LNCS 3203), 2004, pp. 881-885

[1]**Mr. P. R. Bokde** working as Assistant Professor in Department of Electronics at Priyadarshini Bhagwati College of Engineering, Nagpur. He is presently pursuing Ph. D. from Nagpur University, Nagpur.He has completed M. Tech. (VLSI), B.E. (ETRX) from Nagpur university, Nagpur. He has presented many National and International papers and attended National & International conferences. He has a special interest in Communication, Signal Processing and VLSI.

[2]**Mr. P. N. Aerkewar** working as Assistant Professor in Department of Electronics at Priyadarshini Bhagwati College of Engineering, Nagpur. He is presently pursuing Ph. D . from Nagpur University, Nagpur.He has completed M. Tech. (VLSI), B.E. (ETRX) from Nagpur university, Nagpur. He has a special interest in Digital Electronics, Signal Processing and VLSI.

[3]**Mr. P. M. Palkar** presently working as Assistant Professor in Department of Electronics at Priyadarshini Bhagwati College of Engineering, Nagpur. He is pursuing Ph. D. from Nagpur University, Nagpur.He has completed M. E. (Digital Electronics) from Amaravati University and B.E. (Electronics) from SGGS, Nanded, He has a special interest in Digital Image Processing ,Digital Communication and VLSI.