Research Article

DYNAMICALLY CO-SYNTHESIS OF H/W & S/W AND OPTIMIZATION IN RECONFIGURABLE EMBEDDED SYSTEM

Sunil Kr. Singh [1], R. K. Singh [2], M. P. S. Bhatia [3]
Address for Correspondence
[1] Ph.D, Research scholar, Uttarakhand Technical University, Uttarakhand, INDIA
[2] Professor, Uttarakhand Technical University, Uttarakhand, INDIA,
[3] Professor, Netaji Subhash Institute of Technology (NSIT), New Delhi.INDIA
E Mail anujsunilsingh@yahoo.co.in

ABSTRACT
Field Programming Gate Array (FPGA) play an important role in reconfigurable computing. Reconfigurable computing has been used to build reconfigurable embedded system. Reconfigurable embedded System is dynamically changing in the hardware circuit at runtime with the reconfigurable characteristic of Programmable Logic Devices like FPGA, to give the system advantages in both hardware and software. A general embedded system has high performance with minimum flexibility but due to fast development of technology, embedded system need more performance with more flexibility. Reconfigurable computing devices like FPGA promise to meet both the needs i.e. flexibility and performance. Reconfigurable hardware can provide a flexible and efficient platform for fulfilling the device area, cost, performance, and power requirements of many embedded systems. In this paper, we try to presents an overview of reconfigurable computing in embedded systems in terms of runtime hardware & software optimization, co- synthesis process and its benefits. We also try to solve some of design issue, tool and method for designing of Reconfigurable Embedded System.

KEYWORD: Reconfigurable Computing, ASIC, FPGA, HW/SW partitioning, Dynamic reconfiguration, Scheduling, HW/SW co-synthesis.

I. INTRODUCTION

With the uninterrupted progress of the electronics and the computer technologies, many emerging application in the fields of multi-media, (mobile) communication, networking, computing, consumer electronics etc. require high performance, flexibility and wide variety of functionalities after the system utilization. That's why we try to present in this paper that system architecture has been a emerging topic for research and development and much has change over last few years of the history of modern high-performance embedded system. Flexible system must function in rapidly changing environment in multiple mode of operation. For such system, efficient hardware architecture must mach with algorithm to maximize performance and minimize resource. So that structurally adaptive reconfigurable architecture can meet both these need, achieving high performance with changing algorithm[3].

Reconfigurable computing device, like FPGA (Field programmable Gate Array) allow the implementation of architecture that change in response to the changing environment. These reconfigurable hardware components are already being used in combination with traditional processor to deliver novel ways of implementing application. So FPGA technology shows great promise computational system. Thus the target systems are built on a heterogeneous computing platform: including reconfigurable H/W, ASIC and general purpose processor.

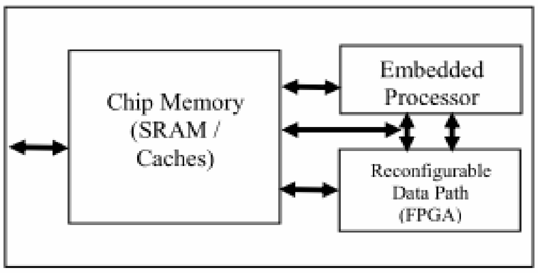


FIG. 1: General structure view of Dynamic reconfigurable Embedded System

II. RELATED RESEARCH WORK

Research related area in the field of Reconfigurable Computing is rapidly advancing for scientific and high performance multimedia applications. While today's Field programmable Gate Array (FPGA) technology shows great promise for implementing reconfigurable computational systems, their capabilities in certain areas (such as floating point arithmetic) cannot equal other technologies. For this reason, efficient system architectures must encompass an heterogeneous mix of the best technologies. The target systems are built on a heterogeneous computing platform: including configurable hardware, ASIC and general purpose processors and FPGA based upon partial reconfigurable SRAM[3,5,6]. The primary difficulty in this approach lies in system design. A designer must now maintain a set of different system architectures, which exist at different times in the system's lifetime, and map these architectures onto the same group of resources. The designers must manage the behaviour of the system, determining the operational modes of the system, the rules for transitioning between operational modes, and the functional properties within each operational mode. In addition, the system must make efficient use of the resources, enabling the designer to minimize the envelope of hardware required to support the union of all operational modes. Currently system design tools are insufficient to manage this complexity[12]. In this paper, we try to propose some basic tool for designing RES.

III. RECONFIGURABLE COMPUTING SYSTEM:

Conventional computing for the execution of algorithm have primary two methods, one is using Von-Neumann computing in which programming of microprocessor/ microcontroller using S/W. the second method is making application specific processor or integrated circuits. The first one is more flexible solution with performance degradation . in the later method the system has been designed for particular application , may not be cost effective to modify to add more feature. So the use of reconfigurable computing system (RCS) which fill the gap between H/W (ASIC/ASIP) & S/W (Microprocessor) approaches. Because conventional computing system have fixed H/W and variable algorithm (S/W) to implement a system but In RCS, it has both H/W as well as algorithm (S/W) are variables[7][13].

With this approach, reconfigurable computing systems have some basic properties and model which are given as:

A) PROPERTIES OF RCS:

Reconfigurable configurable computing system normally consisting of a matrix of programmable computational units with a programmable interconnection network superimposed on the computational matrix. The main characteristics that have in RCS are

1. Distributed computation,
2. Configurable data path
3. Spatial computation
4. Distributed control and scheduling

B) DESIGNE MODELS OF RECONFIGURABLE COMPUTING

There are some basic model can be used for simulating the architecture of RCS, so that a designer carefully analyze the need of method and component used in design of target system. Some are given as

1. Behavioural modelling
2. Algorithm / Structural Modelling Mesh model
3. Resource Modelling
4. Partially Reconfigurable model
5. Multi-context modelling

6. Constraint modelling
7. Parallel Model

IV. CONFIGURATION & SYNTHESIS IN EMBEDDED SYSTEM:

A basic model configuration process generates hardware architecture specifications, software modules, process / schedule tables, communications maps, synthesizable hardware specifications, and a run-time Configuration Manger for dynamic adaptation to changing environments. The synthesis process attempts to optimize hardware/software architectures for user- cost such as weight, power, algorithmic accuracy and flexibility[14].

At this point, the synthesis procedure can generate the actual runtime artifacts. From the behavioral models, a set of tables is produced for the Configuration Manager. The state based behavior is defined in the Behavior Models. These models are transformed into a compact state table. The table contains next state equations for each operational mode. The interfaces to internal and external events are generated to provide the state transition variables to the state machine. These tables and variable interfaces are executed directly by the configuration manager. The synthesis process for hardware, software and both are given as[8,12]:

A) HARDWARE SYNTHESIS

For each configurable component (FPGA), a design specification is generated. This design specification includes a hardware design file for each component for each mode. The design for a component*mode is specified in structural VHDL. The VHDL design incorporates computational components from the design library, which can contain user defined VHDL behavioral descriptions and vendor-supplied Intellectual Property (IP) modules. These modules are glued together using components from a standard interface runtime library, which is part of the Runtime Environment described later. These interfaces connect computational components on the same chip with simple FIFO's and asynchronous handshaking interfaces. These interface components manage the physical hardware resources (pins and wires), buffer data, and multiplex multiple logical communications across a single set of wires. When required, data format conversions are also supplied.

B) SOFTWARE SYNTHESIS

For the general-purpose RISC/DSP components, a set of software specifications is generated. These specifications provide the information needed by the Runtime Environment to enact the desired computational behavior. The Runtime Environment requires several categories of design files:

- Software Load Modules contain executable modules that are downloaded to the processors in the system. The system can generate a common load module that contains the superset of all executable functions (if memory is sufficient) or it will generate a customized module for each of the processors in the system. The customized module is clearly more memory-efficient.
- Real-time schedules contain the list of processes and their priorities. A unique schedule is generated for each processor and for each mode of operation.
- Communication maps describe the information flow between processes. These "streams" can perform communication between two modules on the same processor, or they can transport data across the network, through intermediate processors, and to a remote process anywhere in the system.

C) H/W & S/W SYNTHESIS

Interfaces between software modules and hardware modules/data sources/sinks are automatically inserted during the synthesis process. These interfaces perform the "care and feeding" of hardware interfaces, converting complex communication protocols into simpler hardware compatible protocols. The interfaces also multiplex multiple logical streams over a single physical port and perform data conversion functions [13].

The result of the synthesis and post processing is a complete executable system, ready for deployment. The deployment is performed in concert with the Runtime Environment.

V. DYNAMIC RUNTIME RECONFI-GURATION

The dynamic runtime reconfiguration must support implementation platforms with the following attributes:

- **Heterogeneity:**

Optimizing the architecture for performance, size, and power requires that the most appropriate implementation techniques be used. Implementations will require software (implemented on RISC and DSP processors), configurable hardware on FPGAs, and a mix of ASIC components.

- **Low overhead / High Performance:**

The runtime environment must minimize overhead, since overhead results in extra hardware requirements.

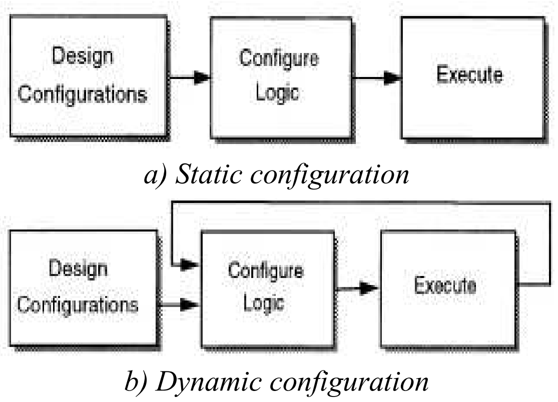
- **Hard Real-Time:**

The target systems have significant real-time constraints.

- **Reconfiguration:**

The execution environment must allow hardware and software resources to be reallocated dynamically. During reconfiguration, the application data must remain consistent and real-time constraints must be satisfied.

All the above issues must be addressed at multiple levels of system configuartion. At the lowest level, the hardware must be capable of reconfiguration. As in shown figure: 2.



a) Static configuration

b) Dynamic configuration

Figure 2: A general view of system configuration

Software-programmable components, such as DSP's and RISC processors, have excellent inherent hardware support for reconfiguration, since software has the ability to change system function by changing memory contents. Internal CPU hardware structures are designed to restrict dangerous conditions that could damage hardware. FPGA's are an unrestricted collection of gates, switches, and connectors. This process must be provided by a cooperation of the design process and the runtime infrastructure.

VI. HARDWARE ARCHITECTURE & APPLICATION OPTIMIZATION

A) Architecture optimization

Reconfigurable system is a promising alternative to deliver both flexibility and performance at the same time. Technology-dependent tools and high-level abstract supporting tools have been proposed to solve the various design problems at different abstraction levels. However, a complete overview of how to integrate them into a single design flow is missing. In this work, we use a real case study to demonstrate our design target architecture optimization flow of Run Time Reconfigurable (RTR) systems.

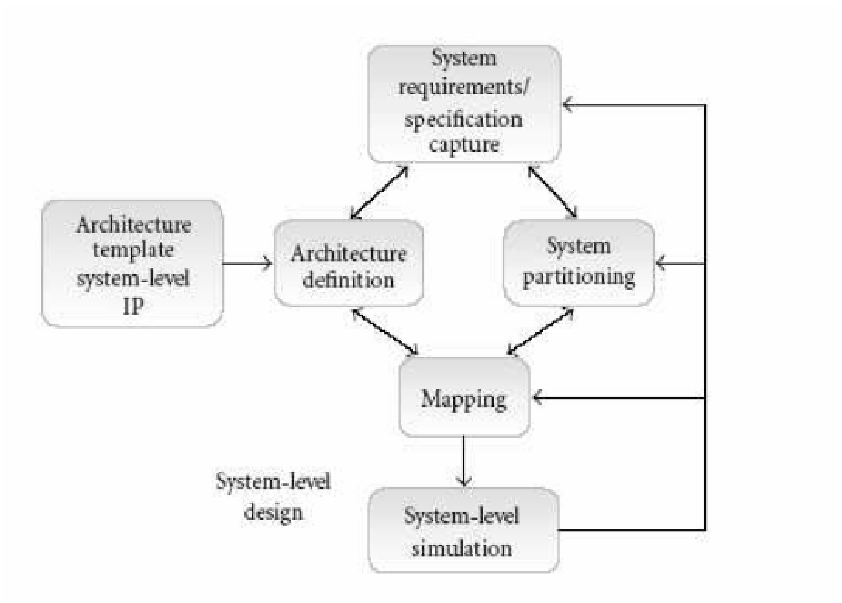


Figure 3: A general View of system-level design flow

The above design flow is divided into some basic system-level design and implementation-level design. The process at system-level design is to make various partitioning decisions and evaluate system performance. At implementation level, executable code and HW net list are generated using technology-dependent tools. A general view of the system-level design flow is given in above Figure 3. The some basic following new features are identified in each phase when reconfigurability is taken into account.

(i) ***System requirements/specification capture*** needs to identify requirements and goals of reconfigurability (e.g., flexibility for specification changes and performance scalability).

(ii) ***Architecture definition*** needs to model the reconfigurable technologies of different types and vendors at abstract level and include them in the architecture mode.

(iii) ***System partitioning*** needs to analyze and estimate the functions of the application for SW, fixed HW, and reconfigurable HW. The parts of the targeted system that will be realized on reconfigurable HW must be identified. There are some rules of thumb that can be applied. If an application has roughly several same sized hardware accelerators that are not used at the same time, these accelerators can be implemented onto DRHW. If an application has some parts in which specification changes are foreseeable or there are foreseeable plans for new generations of the applications, it may be beneficial to implement it onto DRHW.

(iv) ***Mapping needs*** to map functions allocated to *reconfigurable* hardware onto the respective architecture model. The special concerns at this step are the temporal allocation and the scheduling problem. Allocation and scheduling algorithms need to be made either online or offline.

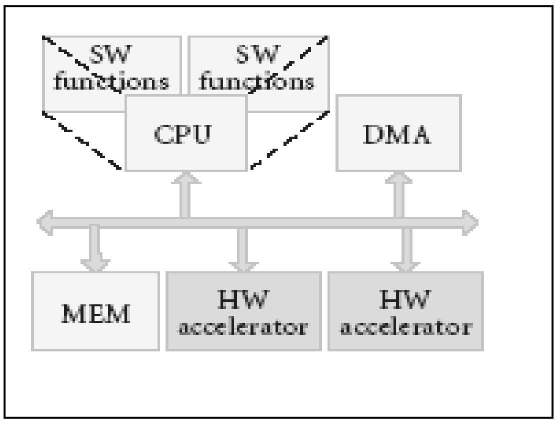


Figure 4: A general view of initial fixed architecture

(v) ***System-level simulation*** needs to observe the performance impacts using reconfigurable resources for a particular system function. The effect of configuration overhead should be highlighted in order to support designers to perform system analysis or design space exploration.

The reconfigurable components are mainly used as coprocessors in System on Chips. The new architecture is redefined partly based on the existing architecture and partly using the system specification as input. The initial architecture is often dependent on many things not directly resulting from the requirements of the application. Therefore, the initial & final architectures and the HW/SW partition are often given at the beginning of system-level design (figure 4, 5)[9][12].

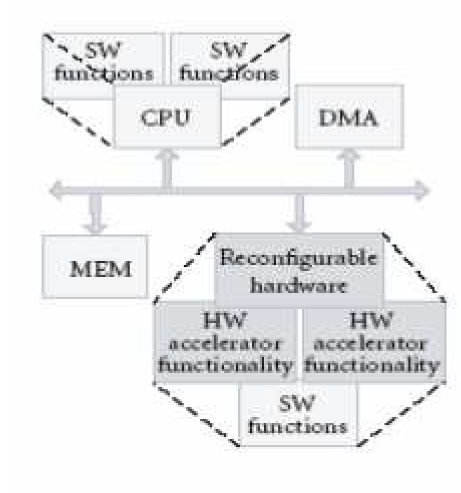


Figure 5: A modified view of reconfigurable system level design flow

B) APPLICATION OPTIMIZATION

Generally FPGA has used over the ASIC design flow for the embedded system because run- time or dynamic reconfiguration is of special interest among all the researcher because it provides a performance and cost advantage over a load – time configuration. The processor can be an embedded RISC like ARM or MIPS, generally such configuration use an open- source processor LEON, which is compatible with SPARC V8 and has same performance as RISC. The main advantage of an open source core is that to provide design flexibility, i.e. provide interface between FPGA and processor. The FPGA has direct access to main memory. It compile C code to the FPGA easier the other communication models[7][6].

Generally, there is a particular border between H/W and S/W design. Software is generally programmed in a sequential mode but hardware is developed in parallel mode. In dynamic reconfiguration, the gap between H/W and S/W is burned by FPGA. We try to manually explain application optimization in RES through transformation on C source code. For this purpose we make use of FPGA features to improve performance and reduce area.

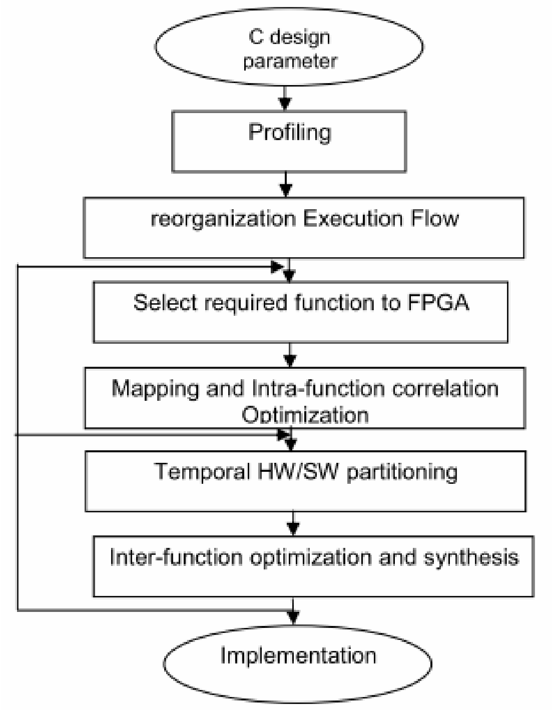

C design parameter
Profiling
reorganization Execution Flow
Select required function to FPGA
Mapping and Intra-function correlation Optimization
Temporal HW/SW partitioning
Inter-function optimization and synthesis
Implementation

Figure 6: Application optimization Design flow in reconfigurable embedded system.

Due to dynamic reconfiguration, the optimization is divided into two stages. The first stage in intra-function optimization. Main techniques include specify word length, simple SIMD etc. the intra-function optimization will not affect other parts of the system. In the second stage, after the temporal partitioning, we get the knowledge of different functions that how they are grouped into one configuration and how they are perform inter-function optimization (figure:6). The main technique includes resource, localizing memory access, pipelining and parallelizing among functions. So by using two stage optimization, we get more performance and use small area than a sequentially written C program can provide. As a c based methodology, various design steps are performed on c level, like, profiling, mapping, optimization and restructuring execution flow. Its also reduce the debugging and verification work[5][12].

VII. CONCLUSION:

The main advantage of Reconfigurable Embedded system is the combined flexibility and performance. However, implementing RES does require extra efforts in the various design stages, from the general View of system-level design flow to modified view of reconfigurable system level design flow, In this paper, we discuss the design issue of Reconfigurable Embedded System (RES) by using reconfigurable computing. High performance with limited resources needs application-specific architectures (ASIC), while flexibility requires adaptive capabilities. We also present a design flow of application optimization for Reconfigurable systems. However, some key step of our design issue, tools and method for designing of Reconfigurable Embedded System are still done manually, and the demonstrator is under development. We are going to concentrate on different design issue. The main challenge is to develop such system also to develop a compiler which compile the selected and optimize application code for the target reconfigurable embedded system. Our goal is to provide the basic design issue for reconfigurable embedded system architecture.

REFERENCE:

1. Villasenor, J., Mangione—Smith, W., “Configurable Computing”, Scientific American, June, 1997.
2. S. Edwards, L. Lavagno, E.A. Lee, A. Sangiovanny- Vincentelli, “Design of Embedded Systems: Formal Models, Validation, and Synthesis”, *Proc. of the IEEE*, vol. 85, no. 3, March, 1997, pp. 366-390.
3. Sunil. Kr. Singh, Dr. M.P.S. Bhatia, Dr. Rajni Jindal, Design issue of reconfigurable embedded system, in *Proceeding of the International conference on advance computing technologies, December 2008*
4. Xilinx , Inc., Virtex- E 1.8V field programmable Gate Array, Feb 2001.
5. R. Ernst, J. Henkel, and T. Benner, “Hardware–software cosynthesis for microcontrollers,” IEEE design Test Comput., vol. 10, pp. 64–75, Dec.1993.
6. K. M. GajjalaPurna and D. Bhatia, “Temporal partitioning and scheduling for reconfigurable computing,” Proc. IEEE Symp. FPGAs for Custom Computing Machines, pp. 329–330, 1998.
7. K. Bondalapati andV. Prasanna. “Reconfigurable Computing systems in Proc. IEEE, vol.90, no.7, July 2002, pp.1201-1217.
8. K. Chatta and R.Vemuri, “Hardware–software codesign for dynamically reconfigurable architectures,” in Proc. of FPL’99, Glasgow, Scotland, Sept. 1999.
9. Juanjo Noguera and Rosa M. Badia, “HW/SW Codesign Techniques for Dynamically Reconfigurable Architectures”, IEEE Transactions on VLSI Systems, Vol. 10, NO. 4, august 2002, pp 399-415.
10. Xilinx, “Virtex platform datasheet,” May 2007, http://www.xilinx.com. ,Septembe 1999.
11. Bondalapati andV. Prasanna. “Reconfigurable Computing systems in Proc. IEEE, vol.90, no.7, July 2002, pp.1201-1217
12. Juanjo Noguera and Rosa M. Badia, “HW/SW Codesign Techniques for Dynamically Reconfigurable Architectures”, IEEE Transactions on VLSI Systems, Vol. 10, NO. 4, August 2002, pp 399-415.
13. Andr´e DeHon and John Wawrzynek, “Reconfigurable Computing: What, Why, and Implications for Design Automation”, technical report.
14. Sandeep Neema: “Constraint based System Synthesis”, Technical Report, Department of Electrical and Computer Engineering, Vanderbilt University, 1999.