www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 5 May 2016, Page No. 16613-16616

A Survey on Partial Reconfiguration Techniques

Mrs Sowmya Aithal [1]

[1]PG Student, Dr. Ambedkar Institute of Technology,

Bangalore, aithalsowmya@gmail.com

Abstract: The technology evolvement has led to incorporation of more and more features into the system. One such feature is the flexibility of changing the hardware and software at the time of fabrication. Such a feature is called reconfiguration. The reconfiguration allows the user to incorporate more control over the hardware and software even at final stages of the process. Such type of reconfiguration is possible in Field Programmable Gate Array (FPGA). There are various classifications of reconfiguration based on application of reconfiguring a device usually FPGA. In this paper we go through the overview of partial reconfiguration and various techniques employed to achieve it.

Keywords: Dynamic Partial Reconfiguration, FPGA, Partial Reconfiguration, Reconfiguration.

I Introduction

The reconfiguration is the key factor in today's world of technology as it gives us control over the system at the manufacturing level. The reconfiguration will allow adapting to the system changes.

The Application Specific Integrated Circuit (ASIC) is very fast and efficient when it is designed to do a specific task. The hardware cannot be changed or reconfigured after fabrication. The microprocessor or microcontroller is software controlling hardware and it has more latency in computation as it needs minimum 4 cycles to do any operation. The reconfigurable device like FPGA is one where software generates hardware and so is flexible. The Mask Programmable Gate Array (MPGA) is also one of the reconfigurable devices like FPGA. MPGA are transistor arrays which can be configured into logic by metal interconnections, made at the time of fabrication.

The reconfiguration is classified into static and dynamic reconfiguration. Static reconfiguration is also called Compile time reconfiguration. It is the simplest form and most commonly used type of reconfiguration. Here the reconfigurable resources are loaded with the respective reconfigurations. When the operation begins the resources will remain there until the operation finishes. Here the hardware resources remain static throughout the operation. So it is aptly called static reconfiguration.

The dynamic reconfiguration is also called Run time reconfiguration. It is one commonly used type of reconfiguration. Here it uses dynamic allocation of resources at run time. This increases the performance and use of optimized circuits which are loaded and unloaded dynamically during the operation. So in this way the flexibility of the system is maintained and functional density is increased.

II Partial Reconfiguration

Partial Reconfiguration is the process of reconfiguring logic of part of the system while other parts are operating normally. The partial reconfiguration is effective as it allows the designer to move or change fewer devices and thus reduce power and improve system upgradability.

To program the FPGA we will have to change the bit-file. The bit–file can be greater than 1MB so while reconfiguration we have to alter the bit-file so it is too hectic job so we go for partial reconfiguration.

There are two types of partial reconfiguration based on the functionality. One is static partial reconfiguration and other is dynamic partial reconfiguration.

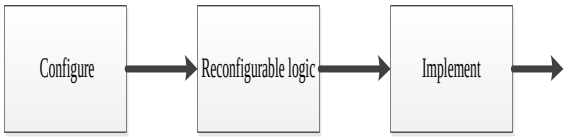

Figure 1: Static Partial Reconfiguration

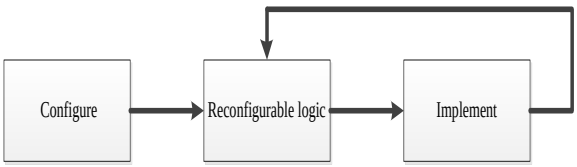

Figure 2: Dynamic Partial Reconfiguration

In static partial reconfiguration, the device is not active and it becomes active after the reconfiguration process as illustrated in figure 1.

Dynamic partial reconfiguration (DPR), also known as active partial reconfiguration. Here the alteration can be done when rest of FPGA is running. It is carried out to allow FPGA to adapt to the reduced power, higher efficiency and resource utilization. DPR is very useful in critical environments where the devices need to continue operating when some of the regions are redefined.

There are 2 basic styles of DPR on single FPGA. The difference based partial reconfiguration and module based partial reconfiguration.

Difference based partial reconfiguration[1] is used when small change is made to the design. This method is widely used in Look Up Table (LUT) for changing the equations or in the dedicated memory. Here the partial bitstream contains only the difference between the old content and new content of an FPGA. The switching configuration of a module from one implementation to other is very quick as bitstream are very small.

Module based partial reconfiguration[1] is used when we have to reconfigure large blocks of logic. Reconfigurable modules are the distinct portions of the design which need to be reconfigured. For any reconfigurable module some specific properties and specific layout need to be laid out so we need to plan the use of partial reconfiguration in FPGA.

Another technique is Early Access Partial reconfiguration (EAPR) [1]. Here the DPR system is used in EAPR flow. The gray and Johnson counter are reconfigured dynamically on Xilinx Virtex4- FX12 FPGA chip. The output has four LEDs used to demonstrate the counter. Figure 3 illustrates the proposed architecture. The system consists of LED control module and two partial reconfiguration modules (PRM A1, PRM A2) which are placed on same partial reconfiguration design. LED control module which enables LEDs is a static module.

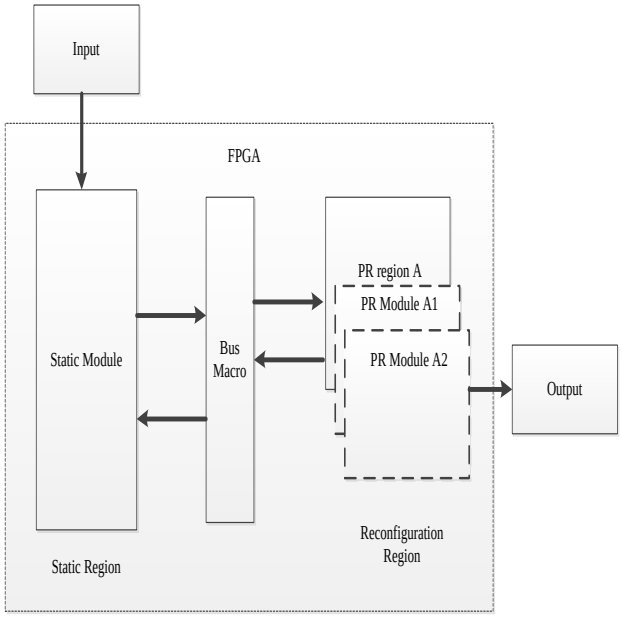

Figure 3: Architecture of the design

EAPR Design Flow is shown in figure 4. Here in first step of Hardware Descriptive Language Synthesis where the top design, base design and PRM design.

In Top Design the top module which has clock instantiations, I/O instantiations, partial reconfiguration instantiations, signal instantiations and all macro instantiations are done.

In Base Design the static modules are present and are static throughout the reconfiguration. This module does not contain clock or reset.

In PRM Design also clock is not included in the module but can be referenced through the top module. In this design we have 2 PRM's one of Gray Counter another Johnson Counter.

The Second step is Set Design Constraints. Here the constraints are area group, reconfiguration mode, timing constraint and location constraints. The constraint area group defines which modules in top module is static and which are reconfigurable. The reconfiguration mode constraint is applied to specific group to be reconfigurable module. Location constraints are set for each pin, clock pins and all bus macros.

The Third step is Implement Base Design where translate, map, place and route is implemented. Before implementation constraints file should be created.

The Fourth step is Implement PRM's where each PRM is implemented separately and follows translate, map, place and route as in base design implementation.

The Final Phase is Merge where complete design is built based on base design and each PRM's. Here many partial bit-streams are generated for each PRM's and full bit-streams are created initially to configure FPGA.

The EAPR technique is used for small blocks like counters or multipliers etc, but for real time signals or video or speech we cannot use the EAPR technique. The below mentioned technique is used in such cases.

DOI: 10.18535/ijecs/v5i5.47

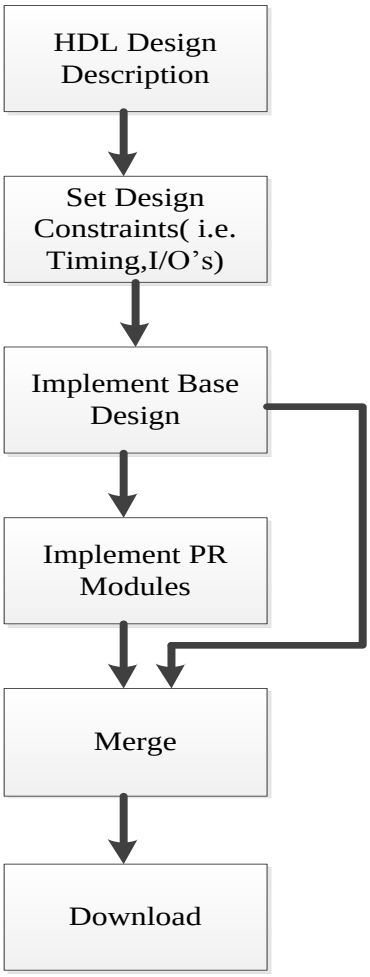

Figure 4: EAPR design Flow

Another technique for partial reconfiguration is for video processing[2]. Here Speed Efficient Dynamic Partial Reconfiguration Controller (SEDPRC) is used. Noise is present in an image while generation, distribution or in display. The image noise appears in different forms based on their place of generation so we need different kinds of filters to be used while processing an image. So having variety of filters is useful but only one type of filter is used at a time. Now DPR can be exploited to accommodate different types of filters. The filters are deployed in slots at run time based on their requirement. The configuration is shown in figure 5.

The reconfiguration controller can choose filters from filter library and deploy them to slots at run time. By this method we can have larger number of available filters than normal number which fit into the target device. If a video filter is getting swapped then the remaining pre/post processing and audio processing block become optional. In the same way audio processing can be swapped without stopping video processing.

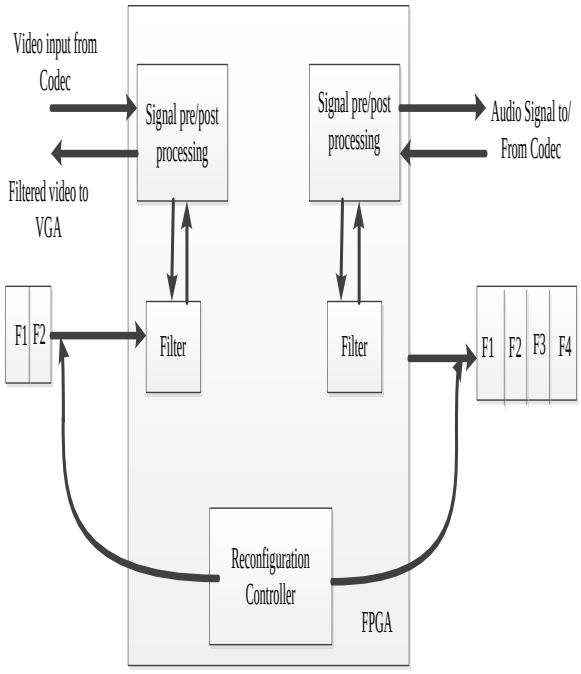

Figure 5: Overview of the dynamic partial reconfigurable audio/video signal processing system.

Here parallelism is exploited in FPGA's by making video and audio processing blocks running in parallel on the device and in the process changing the functionalities of the system without stopping the working ones.

The reconfiguration controller should be fast enough to achieve filtering without losing data. In video processing the filters should be swapped after the completion of first frame processing and before the arrival of the second frame.

The other technique for partial reconfiguration[3] is use of Internal Configuration Access Port (ICAP). ICAP is Xilinx primitive which provides internal access to the configuration logic of the FPGA from within FPGA fabric. The ICAP interface is where the configuration data can be dynamically loaded into the configuration memory of FPGA at runtime. It is possible to read back of the configuration data from configuration memory or to read the status registers of the configuration logic with the ICAP interface.

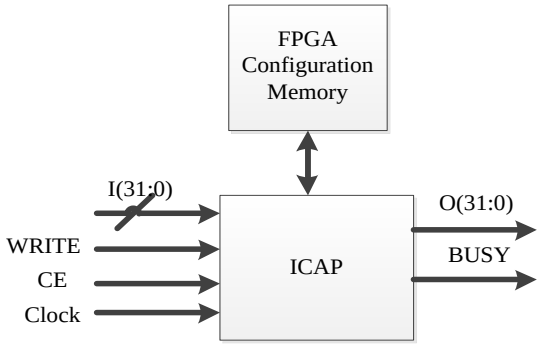

Figure 6: Xilinx ICAP Primitive

ICAP interface consists of separate data ports for reading (output) and writing (input) configuration data. For

Virtex 4 the configuration data ports are 8 or 32 bits wide, for Virtex 5 or Virtex 6 the configuration data ports are 8, 16 or 32 bits wide. The chip enable (CE), write enable (WRITE) signals are the inputs along with the Clock. The output signal is busy/ready (BUSY). The BUSY signal is low during WRITE operations and high during read operations. The WRITE signal writes when it is low and reads when high. At rising edge of the clock the configuration data is written to the device, if ICAP port is enabled. So the controlling of writing configuration data is done by use of clock or enable input.

Partial reconfiguration when need to be done for various combinations then we cannot use the above techniques so the next type is LUT based partial reconfiguration[4].

Xilinx FPGA's allow some LUTs to function as Shift Registers at the same time have LUT functionality, referred to as SRL's. This dual functionality allows shifting the configuration bits, defining the LUT content and the behavior without going for normal FPGA procedure like ICAP.

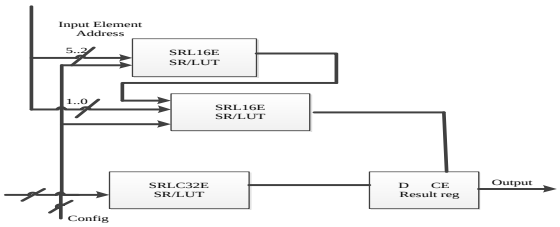

Figure 7: SRL based FU implementation.

The block diagram shown in figure 7 has SRL based Functional Unit (FU). For the address comparator part which is 6 input one, we need to implement two 4-input SRL named SRL16E. Both the LUTs check for the equality of the Most Significant Bits (MSB) and Least Significant Bits (LSB) of the input and the configured bit. The first of the LUTs check for LSB other for MSBs. The second LUT implements the AND gate to combine the results and then enable register to load the result from the function part. The implementation of the function part is done using one single 5 input LUT called SRLC32E as its data width is tailored to its size. The configuration lines are needed for serial shifting of configuration data into LUTs. This type of partial reconfiguration is fast but has communication overhead, the latency is dependent on the largest SRL size (i.e 32 cycles.).

III Conclusion

The various partial reconfigurations are learnt and various techniques employed in dynamic partial reconfiguration is discussed with each technique used for specific application or purpose. The partial reconfiguration plays an important role in the field of technology based on FPGA.

References

[1] Dynamic Partial Reconfiguration in FPGAs by Wang Lie, Wu Feng-yan, Dept. of Computer Science & Electronic Information ,Guangxi University Nanning, China in Third International Symposium on Intelligent Information Technology Application IEEE ,2009

[2] High Speed Dynamic Partial Reconfiguration for Real Time Multimedia Signal Processing by S. Bhandari, S. Subbaraman, S. Pujari, F. Cancare, F. Bruschi, M. D. Santambrogio and P. R. Grassi in 15th Euromicro Conference on Digital System Design IEEE, 2012.

[3] High Speed Partial Run-Time Reconfiguration Using Enhanced ICAP Hard Macro by Simen Gimle Hansen, Dirk Koch and Jim Torresen in IEEE International Parallel & Distributed Processing Symposium,2011.

[4] Lookup Table Partial Reconfiguration for an Evolvable Hardware Classifier System by Kyrre Glette, Paul Kaufmann in IEEE Congress on Evolutionary Computation (CEC), 2014.

[5] Performance Evaluation of Hybrid Reconfigurable Computing Architecture over Symmetrical FPGA by Sunil Kr. Singh, R. K. Singh, M. P. S. Bhatia, in International Journal of Embedded Systems and Applications (IJESA),2012.

[6] Reconfigurable Computing Architecture Survey and introduction by Ali Azarian, MahmoodAhmadi in IEEE, 2009.

[7] Partial and Dynamic Reconfiguration of FPGAs: a top down design methodology for an automatic implementation by Florent Berthelot, Fabienne Nouvel in Emerging VLSI Technologies and Architectures, IEEE, 2006.

[8] A Physical Resource Management Approach to Minimizing FPGA Partial Reconfiguration Overhead by Heng Tan and Ronald F. DeMara in IEEE, 2006.

[9] Dynamic Fault Recovery Using Partial Reconfiguration for Highly Reliable FPGAs by Gehad I. Alkady, Nahla A. El-Araby, M.B. Abdelhalim, H.H. Amer, A.H. Madian in 4th Mediterranean Conference on Embedded Computing, 2015.

[10] Modern Fault Tolerant Architectures Based on Partial Dynamic Reconfiguration in FPGAs by Martin Straka, Jan Kastil, Zdenek Kotasek in IEEE, 2010.