Implementation of FPGA based Firewall Using Behavioral Synthesis

Rajanish K. Kamat1[†], Pawan K. Gaikwad[††] and Santosh A. Shinde1[†††]

Department of Electronics, Shivaji University, Kolhapur – 416 004, India

Summary
Behavioral design helps the designer to understand the design space and subsequently coming up with a design that meets all the constraints specifically in a field programmable gate array (FPGA) based design paradigm. In this paper we have reported a novel design framework for creation of behavioral design. We have examined the opportunities brought about by finite state machines and to harness them into a synthesizable register transfer level (RTL) architecture. We discuss a case study of packet parser its finite state machine (FSM), data path controller architecture and issues related to its Handel-C implementation.

Key words:
Firewall, FPGA, Handel C, Behavioral Synthesis, ASIC .

1. Introduction

A firewall is a dedicated appliance, or software running on a computer, which inspects network traffic passing through it, and denies or permits passage based on a set of rules. Its basic function is to regulate the flow of traffic between computer networks of different trust levels. Conventional firewalls operate at the network layer and their operation is based on stateful or non-stateful type. The former functions on the basis of information on the state of connections (for example: established or not, initiation, handshaking, data or breaking down the connection) as part of their rules (e.g. only hosts inside the firewall can establish connections on a certain port) [1]. The later type has packet-filtering capabilities however; it is unable to make more complex decisions as regards to the stage or level of communications between the hosts. This leads to less security and functioning more like a router from the packet filtering point of view. Conventional firewalls working on the principle of stateful analysis poses a typical tradeoff of security Vs latency. More tightly the security implementations lead to increased latency causing jamming and congestion over the network. In order to overcome the traffic speed bottleneck we are working on the packet filtering approach for firewall implementation [2]. The work is centered on a customized processor development with its architecture tuned to the intended filtering functions. The final prototyping will be achieved on a suitable FPGA target which will serve as a programmable semi-custom application specific integrated circuit (ASIC) to be deployed in between the networks. As mentioned above, building custom silicon in FPGAs leads to significant advantages such as rapid design cycle, early time-to-market, easy transition to structured ASICs and reduced non recurring cost of engineering (NRE) costs. Although the designers are choosing FPGA as a prototyping element, it is observed that the target FPGA is just treated as a black box. This makes the system designer to miss many opportunities to optimize the design to fit within the FPGA [3]. It again reiterates the optimization misconceptions regarding the FPGA based systems in comparison to the ASICs. The two most important design specifications namely spatial and temporal can be achieved in FPGAs too (as in ASIC) by adopting the behavioral design strategy prior to adopting the register transfer (RTL design). Behavioral design specifies abstract description of the operations to be performed to a RTL model with the details of how and when these operations are carried out [3]. However, the design community lacks know how connecting the behavioral design on one side and the efficient RTL through the existing Hardware Descriptor Languages HDLs on the other side. The present paper reports a design frame work that links the behavioral to RTL conversion using the FSM and data path controller architecture. The FSM is used here for partitioning the design into data path control architecture. The frame work developed is applied to the packet parser of the FPGA based firewall. The obvious advantages of adopting the framework for the given applications are presented in this communication.

The paper is organized in various sections. At the outset the underlying details of the behavioral methodology and datapath controller are presented. The problem statement of packet parser development and its utility in the FPGA is given. Subsequently the design flow from the FSM of the packet parser to its equivalent data path controller and implementation of one of the modules in Handel C is discussed at length.

Manuscript received June 5, 2010
Manuscript revised June 20, 2010

2. Behavioral Synthesis and Data path Controller architecture

The behavioral synthesis also known as the high level synthesis is the process of generating a register level design from an algorithmic behavioral specification. A program that models a chip's desired function is called as the behavioral or functional model [4]. There are three essential components of the behavioral synthesis model first the inputs or stimulus to the system, second a module library and third the spatial and temporal constraints.

The behavioral specification is generally written in a high level general purpose language like C or in a Hardware Description Language like VHDL or verilog. Our choice is Handel C [5], which is basically a variant of C oriented towards the behavioral modeling. It is a programming language designed for compiling programs into hardware images of FPGAs. A small subset of C, extended with a few constructs for configuring the hardware device facilitates generation of efficient hardware.

The second component of the behavioral model i.e. module library consists of storage units like registers, memories or FIFOs, execution units like adders and multipliers, and interconnect units like multiplexers and buses. Overall, the behavioral model provides valuable information like the constraints such as area, clock speed, power and the data dependcy or the temporal model of execution. However, from the synthesis point of view, the component-level environment of interest is register transfer architecture. In this architecture, the components are specific hardware manipulations of control and data with connections showing the flow for the desired algorithm. The behavioral synthesis is a process of constructing the register transfer model from its behavioral counterpart by adopting a process called as binding. Binding comprises of two sub processes namely scheduling and allocation. Scheduling information is obtained from the behavioral model and the system constraints. This is used so as to optimize the final RTL in terms of delay and power. The allocation of the RTL interms of FPGA resources are done so as to share them based on the utility and idleness information obtained from the temporal analysis. Traditionally, the output of the synthesis system consists of two interacting components namely data path and controller . With the emergence of the data path controller architecture, the constraints get effectively captured and the mapping of behavioral description to hardware results into the high performance and space efficient machines. The design flow for mapping the system specification to the final architecture is shown in figure 1.

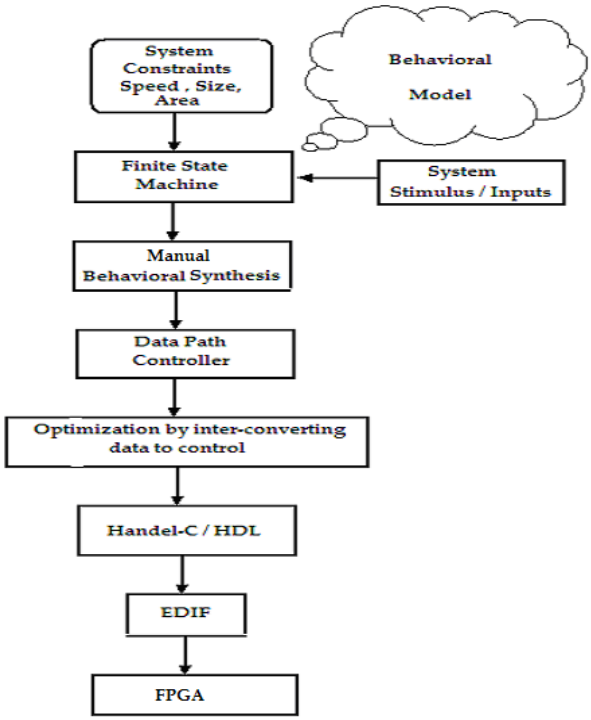

Fig. 1: Design flow for mapping the system specifications to FPGA

3. Top Level Model of Input Packet Parser:

The main objective of the development reported in this paper is a FPGA based firewall processor for high degree of traffic selectivity, by avoiding the usual performance penalty associated with IP level firewalls. Figure 2 shows the top level diagram of the FPGA based packet processor for implementation of the firewall. An incoming packet received from network interface 1, is selectively filtered and either passed to the network interface 2 or rejected. The database of good and bad IPs and valid port numbers is stored in the flash memory and updated on the fly with the reconfiguration feature of the FPGA.

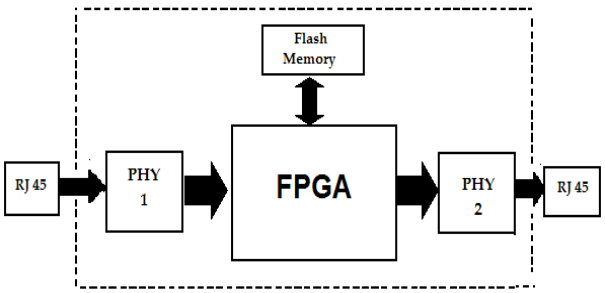

Fig. 2: Top Level Schematic of the FPGA based Firewall

4. Designing Finite State Machine Model of the packet processor:

The behavioral model of the packet parser is designed interms of its FSM implementation. The same is shown in figure 3. The functionality of the FSM depends on the rule base comprising of the IP addresses and port numbers. As shown in the FSM, the incoming packets are stored in a flash RAM and then parsed through the packet splitter for extraction of the source IP address and the port number form the IP header. The validity of the source IP address and the port number is then compared with the IP tables and port numbers stored in rule base. The IP tables stored are of two types viz. a good one and other is bad to be rejected. The incoming IP is primarily compared with the BAD IP table and a decision regarding further transmission or dropping is taken. The IP tables are updated routinely as and when a new bad IP is detected. The built in intelligence based on the statistical mechanism ensures the fixing of bad IP at considerably less depth which is an essential requirement for less latency.

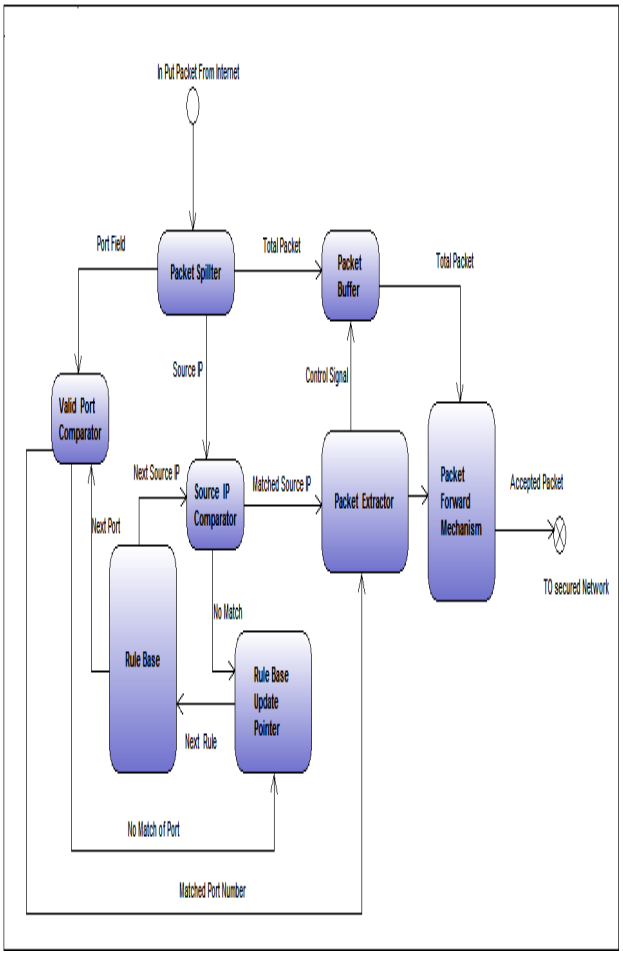

Fig. 3: FSM model of the packet parser

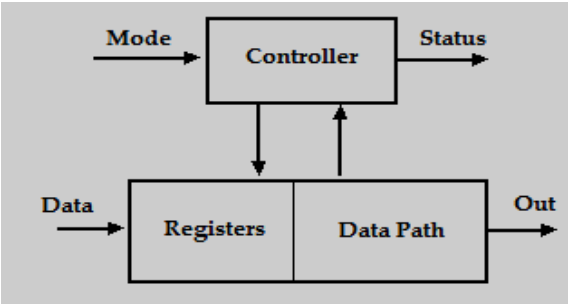

Fig. 4: Basic architecture of a data path controller

4. FSM to data path controller architecture:

The FSM model of the packet parser resembles somewhat to the Mealy machine wherein the primary outputs are a function of both the primary inputs and states. The FSM model can be manually synthesized towards its RTL by categorizing the functionalities into data and control sections as shown in table 1. The data section includes loadable registers and regular arithmetic and logical functions, while the control sections include random logic and state machines.

Table 1: Data and Control sections of the packet parser

Control	***Data***
Packet splitter	Rule Base
Valid port comparator	Packet buffer
Source IP comparator	Packet extractor
Rule base update pointer	Packet Forward mechanism

With the identification of the data path and controller architectures, the basic datapath controller architecture of figure 4 emerges out as shown in figure 5. With this exercise the implicit hardware becomes clear. The basic controller architecture has a 'IF-Else' type of construct in Handel C and is used for checking the conditions, comparing the IP addresses, port numbers etc,. The basic datapath modules can be broken down to set of registers for data and multiplexers for the path or routing part. For instance one of the possible implementations of the rule base could be an amalgamation of set of 32 bit registers due to the 32 bit IP addresses and 16 a set of bit registers for port numbers. The packet buffer will be simply a FIFO pipeline and the packet extractor will be logic block to extract the desired packet out of order from the FIFO. Packet forward mechanism will be again a set of registers which serves as a pipeline to the secured network interface..

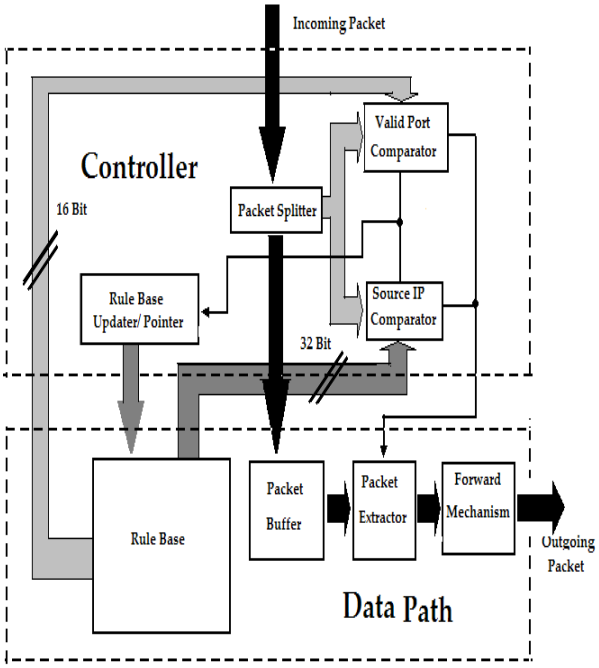

Fig. 5: FSM Categorized into the Data Path controller using manual Behavioral Synthesis

The controller implementation has two obvious options micro coded and hardwired
Looking at the paramount importance of reducing latency, the hardwired style of implementation looks appropriate for the firewall implementation. However, an intelligent design alternative is even a processor less implementation wherever possible to maximize the throughput as described in the next point.

5. Dealing with the behavioral HDL dialects:

The Handel –C language used for the coding the above datpath controller architecture, offers significant advantages over the conventional HDLs such as VHDL or Verilog which employ the basic bottom-up approach for implementing the hardware functionality into the circuit structure. The Handel-C synthesis flow is more abstract and closely corresponds with a typical software flow. The provision of add-on extensions required to describe hardware such as flexible data widths, parallel processing and communications between parallel threads effectively the inherent concurrency. However, the behavioral synthesis subsets of either the HDLs or the Handel-C are not as standardized as the RTL subsets. A clever design methodology can be used to resolve the spatial and temporal issues. As an example, the implementation of source IP comparator is done by using the EX-OR logic and enabling a flag indicating a good or bad IP as shown in figure 6. This alleviates the complicated execution of the controller part in a much efficient spatial implementation. By merely checking the status of the flag, the decision regarding the passage or blocking the packet is taken.

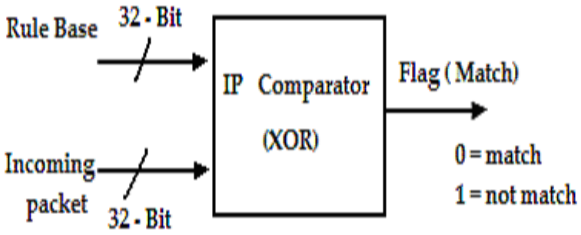

Fig. 6: IP Comparator using Ex-OR Logic

```
If (I R== IF)   /* the I R represents 32-bit IP address
                   stored in the memory and the IF represents
                   the 32-bit IP address of the Input packet.*/
{
Match = 0;}
Else
{ match =1;}
```

Fig. 7: Handel C Implementation of Ex-OR Logic

5. Summary and Conclusion:

There is a constant quest for optimization of the three essential attributes namely speed, power and area in an FPGA based semicustom ASIC implementations. Since the basic architecture of the existing FPGAs has many practical constraints such as limited design size, speed degradation with large systems, relatively high power budgets, PCB (printed-circuit-board) requirements, design and verification requirements etc. Inspite of these shortcomings, designers are constantly stepping forward towards FPGAs as a truly viable production vehicles rather than the ASIC-prototyping tools. It is observed that the designers miss the targeted specifications because they go for the readymade push button synthesis tools. Instead they should go for a semiautomatic synthesis process having a room for manual synthesis to achieve the optimized specifications. This paper has reported such a new design framework to optimize the design specifications in FPGA domain. The framework has been applied to a packet parser which is a core element of the FPGA based firewall.
The design flow comprises of deriving the FSM model of the packet parser. It is further categorized into the data

path and controller to test against the implementation tradeoff of speed Vs Area. As the design problem is from computer networking domain, there is paramount importance to the latency. One tends to obtain it by implementing more parallelism by putting more hardware. However, it not only leads to increased power, but sometimes more multiplexer based routing resources introduces the parasitic and results into worst timing model. The analysis also reveals hardware sharing by employing the scheduling algorithm. As an example the packet splitter and source IP comparators are scheduled on As-Soon-As-Possible (ASAP) basis. On the other hand the packet extractor and buffer are scheduled on As-Late-As-Possible (ALAP) basis. This clears the data dependcy and temporal model of the system interms of clock cycles. The designer can also take up the other issues such as whether to go for a FIFO memory or pipeline approach for the implementation of packet extractor. The pipelining features ranks of the memory elements to reduce the clock cycles at the cost of added latency. The FIFO memory reduces the latency but at the cost of less intelligence preventing the out of order execution and difficulty in updation of the IP base.

Finally we put forth few remarks regarding the choice of the tools for implementation. There are two high-level synthesis alternatives namely the behavioral compilation or simplified way to express parallelism. The behavioral compilers pose a pushbutton approach, completely automating the design flow with little room for the manual intervention. This leaves no scope for the designers to optimize the design. On the other hand a high-level approach such as the Handel-C methodology supplemented by prior analysis of the design problem using the FSM and data path controller achieves the desired specifications still working at the higher or abstract level of the design. With the adoption of the reported framework, designers can multifold their benefits with the inherent features of Handel-C based compilers such as partitioning and synthesis of entire systems including multiple clock domains, control logic, and datapath.

References

[1] L. Qiu, G. Varghese, and S. Suri. “Fast Firewall Implementations for Software and Hardware-based Routers.” Proceedings of 9th International Conference on Network Protocols (ICNP’2001), November 2001.

[2] L. Qiu, G. Varghese, and S. Suri. “Fast Firewall Implementations for Software and Hardware-based Routers.” Proceedings of 9th International Conference on Network Protocols (ICNP’2001), November 2001.

[3] Wolf Wayne, “FPGA based System Design”, PHI Publications, 2006

[4] Wander O. Cesário, Zoltan Sugar, Rodolphe Suescun and Ahmed A. Jerraya, “Overlap and frontiers between behavioral and RTL synthesis”, Draft Version on web retrieved from www.tima.imag.fr/SLS/documents/flex.pdf

[5] Handel-C Manual, www.celoxica.com

[6] Steven M. Rubin, Computer Aids for VLSI Design, Second Edition, Addison-Wesley VLSI Systems Series, 1994

[7] http://www.ieice.org/eng/shiori/mokuji.html

Dr. R.K. Kamat is working as Reader in the Department of Electronics, Shivaji University, Kolhapur, India. He has published 30 research papers and successfully guided two Ph.D. students.

Mr. Pawan K. Gaikwad is working as Lecturer and Head of the Department of Electronics, Willingdon College, Sangli, India. His Ph.D. work is based on development of FPGA based ECG and pulseoximeter.

Dr. Santosh A. Shinde has completed his Ph.D. on “Development of programmable ASIC for circumventing Spam”. He is active researcher in the area of VLSI Design.