ELECTRONICS AND ELECTRICAL ENGINEERING

ISSN 1392 – 1215 *2008. No. 8(88)*

ELEKTRONIKA IR ELEKTROTECHNIKA

SYSTEM ENGINEERING, COMPUTER TECHNOLOGY

T 120

SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

FPGA based Packet Splitter Implementation Using Mixed Design Flow

S. A. Shinde, V. G. Shelake, R. K. Kamat

VLSI Laboratory, Department of Electronics, Shivaji University, Kolhapur – 416 004, India,
email: rkk_eln@unishivaji.ac.in, website: http://www.rkkamat.in

Introduction

Security is a major issue in today's communication networks. The performance pressures on implementing effective network security monitoring are growing fiercely due to rising traffic rates, the need to perform much more sophisticated forms of analysis, the requirement for inline processing, and the collapse of Moore's law for sequential processing. Given these growing pressures, it is time to fundamentally rethink the nature of using hardware to support network security analysis [1]. Many researchers have adopted different strategies for implementing the network security. The traditional approach for the network security is installing the intrusion detection software (IDS) per host. However this leads to many drawbacks. With the software IDS, it is not only harder to correlate network traffic patterns that involve multiple computers but also poses a challenge in coping up with the heterogeneous environment having mixed machine configurations and operating systems. It is also not very difficult to disable the IDS by the attackers. The literature survey reveals that, there is a growing upcoming trend to use Field Programmable Gate Arrays (FPGA) based customized Network On Chip (NoC) and Offload the software processing to hardware realizations. But it turns out to be a costlier solution and therefore hardware-software based hybrid solutions for the security scenario are widely discussed in literature [2].

However, irrespective of software or hardware domain implementations, the most important fundamental building block of the network security/monitoring appliance is the packet splitter. A successful implementation of packet splitter empowers the network administrator to gain valuable insight regarding the network. It entails the out of order packet flow due to incompatibility of the speeds and bandwidths. It facilitates the full analysis of the data harvested from the network for studying the performance issues. The interaction of the protocols can be studied with the separation of the packet headers and port numbers. An IP based smart appliance to be operated remotely using internet can be easily executed with the packet splitter as a core. Moreover, the packet splitter is the core for the firewalls and anti spammers and its implementation dictates the latency issues of these devices. This paper presents the design and implementation of an IP packet splitter for a TCP/IP protocol. The implementation is based on the Xilinx FPGA using mixed set of tools such as Handle–C, ModelSim and Xilinx Webpack.

The rest of the paper is organized as follows. We start with the basic justification for adoption of the design flow having mixed tools, and then gradually present the design from the theoretical framework of finite state machine (FSM), data path controller design and then actual implementation in Handle–C. Further the simulation and debugging issues are taken care by the ModelSim and the register transfer level model and FPGA usage statistics is presented using the Xilinx Webpack.

Optimizing the metrics of FPGA based NoCs by adopting mixed design flow

FPGAs have marked their entry in the network on chip domain due to various reasons. The main reasons are their ability to satisfy the real time constraints, increasing dynamism to cope up with the latency and throughput and faster design and prototyping cycle that enables robust design from testability point of view. FPGA based NoCs have been reported widely in the literature e.g. [4], [5], [6], [7], are commonly considered as a scalable solution for on-chip communication. However, it is also understood that there is no "one size fits all" NoC architecture [8], as different silicon systems have very different requirements from their NoCs. This is especially true in an FPGA based environment where the design metrics such as space, time are very stringent. In this paper we are proposing a novel framework for designing of the FPGA based NoC. A specific case study of the packet splitter is taken up.

The basis of our frame work is analyzing the basic packet splitting mechanism by using finite state machine (FSM). We gradually approach the actual implementation by theoretically transforming the FSM into the data path controller architecture which is the basis of the register transfer machine. Since the entire emphasis is on behavioral architecture, we choose Handle–C for implementation owing to its built in constructs to express the design in more abstract, behavioral form at the higher design abstraction.

The Electronic Design Interchange Format (EDIF) bitstream generated with the Handle–C compilation implements the design successfully in the Xilinx Virtex II FPGA. However, prior to the implementation it is of utmost importance to simulate and debug the design. It is also required to check the fitting specifications in the FPGA paradigm for proper selection of the device. Therefore, the design flow incorporates third party tools for gaining various advantages of simulation, testing and debugging. One of the feature of Handle –C design suite to model the program listing interms of VHDL is exploited for simulation, debugging and RTL synthesis. The VHDL model is then simulated and debugged using the ModelSim from Mentor Graphics. The synthesis view in RTL is also derived by parsing the VHDL listing through the Xilinx Webpack. The paper thus illustrates a mixed design flow that moves from the theoretical frame work i.e. manual behavioral synthesis to Handle C implementation and then to the actual RTL synthesis.

Developing the FSM

The FSM model of the packet splitter is shown in Fig. 1. The key for the implementation is extraction and storage of the following attributes namely, source IP address, destination IP address, source port, destination port, and layer 3 protocol type. The packet count of the incoming TCP/IP packet is extracted at the outset for checking its size. The packet size (ranging from 40 to 1500 bytes), is then set as the counter so as to extract the entire packet till the same is stored in a RAM. The version of the internet protocol (either IPV6 or IPV4) is checked from the first byte of the extracted packet data stored in the RAM. The checksum is done so as to eliminate the corrupted packets going to the forward mechanism. The various attributes of the incoming packets are then stored in temporary memory module for passive monitoring of the traffic without disrupting the flow to the forward mechanism.

Data path controller

The FSM model can be manually synthesized towards its RTL by categorizing the functionalities into data and control sections as shown Fig. 2. The data section includes loadable registers and regular arithmetic and logical functions, while the control sections include random logic and state machines. The data path architecture comprises of packet capture module for inputting the packet, forward-drop mechanism to output the module and memory module to store the temporary processed packet. The controller architecture consists of the checksum calculation and dropping out the corrupted packets accordingly, version checker and layer 3 protocol checker. The data path controller architecture provides insights regarding the further implementation of the modules in Handle–C.

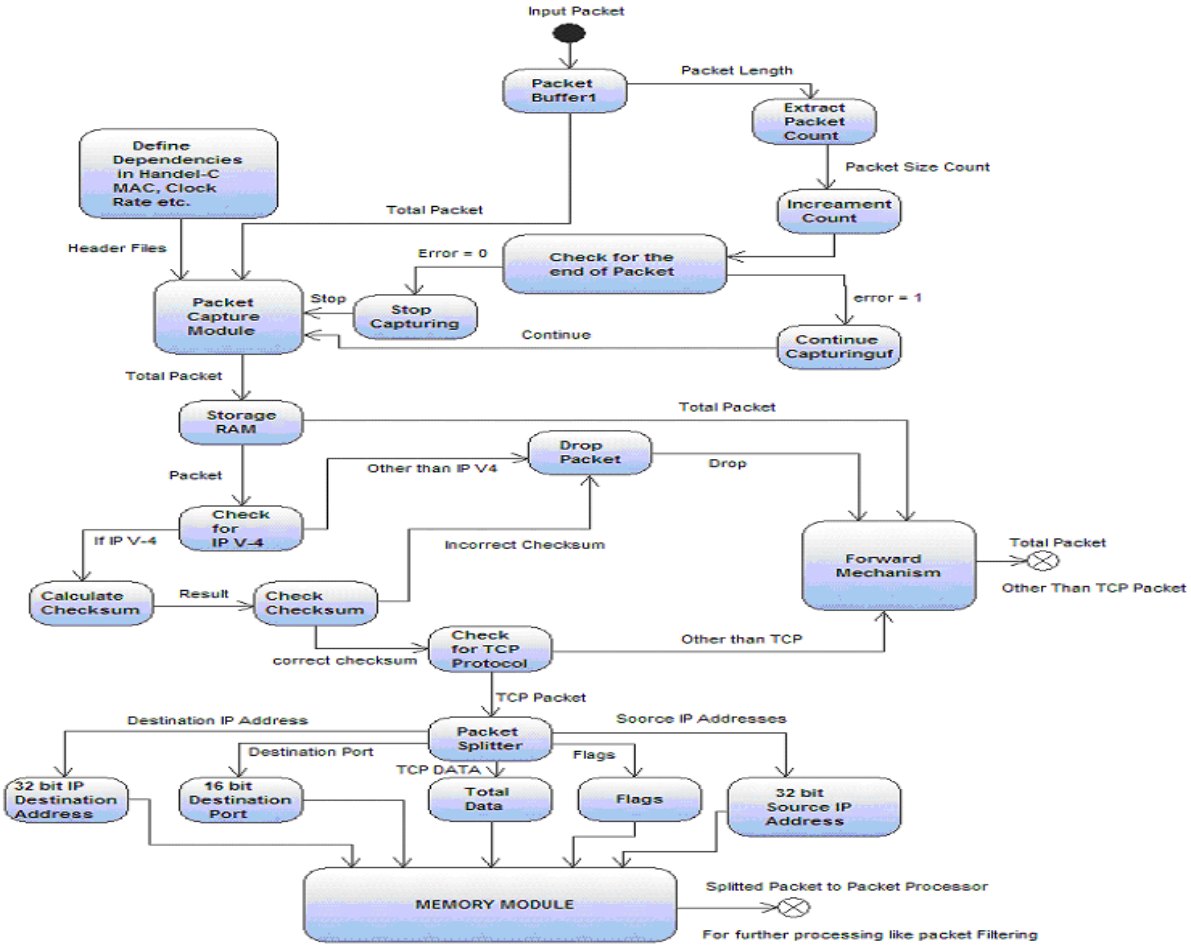

Fig. 1. FSM model of the Packet Splitter

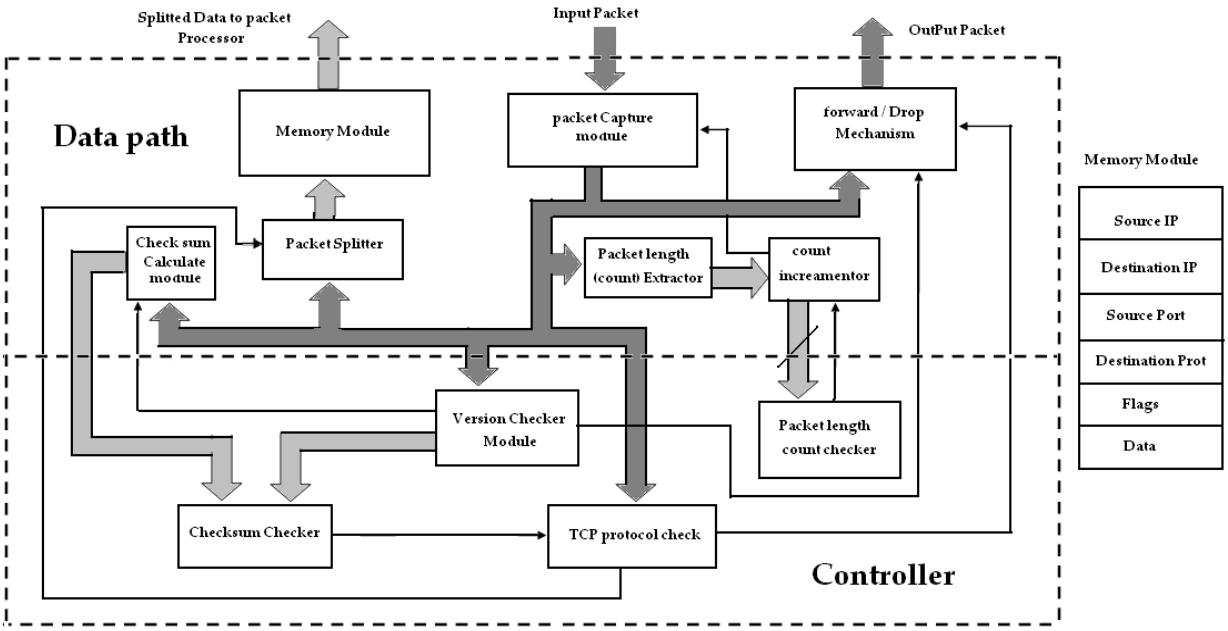

Fig. 2. Data path Controller model of the packet splitter

Handle–C implementation

There are several good reasons for choosing Handle–C. First and foremost is its ability to express the design at very high abstract level. Secondly it has built in Platform Abstraction layer (PAL) that features readymade macros required for the design of the packet splitter. The headers such as Ethernet, console, RAM, display are declared and invoked as and when required in the main program. This kind of leveraging of predefined functionality to access peripherals via APIs leads to fast prototyping and early development time. The concept is illustrated with the handle C program listing of the main macro for reading the packet and storing it in a RAM:

```
macro proc ReadPacket (ConsolePtr, Ethernet)
{
   unsigned 1 Error;
   unsigned 16 Type;
   unsigned 48 Destination, Source;
   unsigned 11 Count, Counter;
   unsigned 8 Data;
static ram <unsigned 8> Temp[1500];
unsigned index;
        / * Attempt to read ethernet packet */
   par
   {
      PalEthernetReadBegin (Ethernet, &Destination, &Source,
                  &Type, &Count, &Error);
      Counter = 0;
   }
   / * If read was successful, display MAC addresses, then packet data. */
   if (Error == 0)
   {
      do
      {
         PalEthernetRead (Ethernet, &Data, &Error);
index=0;
       }
  par
{
         Temp[index]=Data;    //Store packet in RAM
         output! Temp[index];  //Channel Interface for output file
         index++;
        }
            Counter++;
      while (Counter != Count); //stop when all bytes are stored
PalConsolePutChar (ConsolePtr, '\n');
      PalEthernetReadEnd (Ethernet, &Error);
   }
   else
   {
      delay;
   }
}
```

Modeling in VHDL, Co-simulation and Synthesis

The Handle–C implementation is modeled interms of VHDL, which is then simulated using ModelSim Version 6.3. Xilinx Virtex II (device x2v250) FPGA is used for the implementation owing to its SRAM based LUTs that gives optimum fitting. The RTL synthesis report reveals the following information regarding the packet splitter implementation in Virtex II FPGA:

Logic Utilization:

Number of Slice Flip Flops: 411 out of 3072 (13%); number of 4 input LUTs: 653 out of 3072 (21%);

Logic Distribution:

Number of occupied Slices: 513 out of 1536 (33%); number of Slices containing only related logic: 513 out of 513 (100%); number of Slices containing unrelated logic: 0 out of 513 (0%); total Number 4 input LUTs: 675 out of 3072 (21%); number used as logic: 653; number used as a route-thru: 22; number of bonded IOBs: 35 out of 172 (20%); IOB Flip Flops: 32; number of GCLKs: 1 out of 16 (6%); total equivalent gate count for design: 7867; additional JTAG gate count for IOBs: 1680; peak memory usage: 71 MB.

Conclusion

The paper reports successful implementation of FPGA based packet splitter using design flow that comprises of mixed tool sets from different vendors. The FSM lays the sound functional model packet splitter for manual behavioral synthesis towards the data path

controller architecture. The data path controller architecture breaks the functionality into two architecturally diverse parts – one that has data intensive implementation and other with control orientation. The Handle–C chosen for coding the data path control architecture poses various advantages such as algorithmic expressiveness and higher level of abstraction, fast development cycle and fast prototyping. The simulation through ModelSim ascertains the timing and debugging at ease. The VHDL modeling through the Xilinx Webpack presents the RTL model and gives valuable device utilization information.

The implementation of reported FPGA based NoC has number of potential applications. It is a very useful device for in-depth study of the network behavior. The device is core for security implementations such as firewall or anti-spammer. Further it can be used for the effective billing, AS peer monitoring and network traffic engineering and analysis.

Acknowledgement

This work is supported in part by the DST-SERC Fast Track Project for Young Scientist Grant SR/FTP/ETA-14/2006 entitled "Development of FPGA based open source soft IP cores for parameterized microcontroller design" to Dr. R.K. Kamat

References

1. **Vern Paxson, Krste Asanovic, Sarang Dharmapurikar, John Lockwood, Ruoming Pang, Robin Sommer, Nick Weaver.** Rethinking Hardware Support for Network Analysis and Intrusion Prevention // USENIX First Workshop on Hot Topics in Security (HotSec). – Vancouver, B.C. – July 31, 2006.
2. **Saraswathi Sachidananda, Srividya Gopalan, Sridhar Varadarajan.** Hardware-Software Hybrid Packet Processing for Intrusion Detection Systems. – Springer. – 2005. – Vol. 3802/2005.
3. **David V. Schuehler, John W. Lockwood.** TCP Splitter: A TCP/IP Flow Monitor in Reconfigurable Hardware // IEEE Micro. – January/February 2003. – Vol. 23, No. 1. – P. 54–59.
4. **Bertozzi D., Jalabert A., Murali S., Tamhankar R., Stergiou S., Benini L., de Micheli G.** NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip. –2005.
5. **Bolotin E., Cidon I., Ginosar R., Kolodny A.** QNoC: QoS Architecture and Design Process for Networks on Chip // JSA. – Feb 2004.
6. **Goossens K., Dielissen J., Radulescu A.** A Ethereal Network on Chip: Concepts, Architectures, and Implementations // IEEE Design and Test of Computers. – September/October 2005.
7. **Moraes F., Calazans N., Mello A., Möller L., Ost L.** Hermes: an Infrastructure for Low Area Overhead Packetswitching Networks on Chip, Integration // VLSI Journal. – Oct. 2004.
8. **Roman Gindin, Israel Cidon, Idit Keidar**. NoC-Based FPGA: Architecture and Routing. Accessed at: http:// www.ee.technion.ac.il/matrics/papers/NoC-Based%20FPGA.pdf. Retrieved on March 1, 2008.

Received 2008 05 20

S. A. Shinde, V. G. Shelake, R. K. Kamat. FPGA based Packet Splitter Implementation Using Mixed Design Flow // Electronics and Electrical Engineering. – Kaunas: Technologija, 2008. – No. 8(88). – P. 15–18.

Design and development of a FPGA based packet splitter using Handle–C DK4 design suite is reported. A mixed design flow comprising of the integration of tools from third party has been adopted for the simulation, testing, debugging and generation of the RTL model. The reported packet splitter core has lot of potential applications in passive monitoring of the networking setup, security appliances etc. Ill. 2, bibl. 8 (in English; summaries in English, Russian and Lithuanian).

С. А. Шинде, В. Г. Шелаке, Р. К. Камат. Разделитель пакетов данных на основе FPGA // Электроника и электротехника. – Каунас: Технология, 2008. – № 8(88). – С. 15–18.

Описывается новый разделитель пакетов данных, для создания которого использован пакет программ «HandleC DK4». В проекте были использованы инструменты разных производителей. Этими инструментами проведено моделирование, испытание, отлаживание и генерирование модели RTL. Представленный разделитель пакетов данных имеет много потенциальных возможностей применения в пассивном контролировании сетевой установки, программ безопасности и т. д. Ил. 2, библ. 8 (на английском языке; рефераты на английском, русском и литовском яз.).

S. A. Shinde, V. G. Shelake, R. K. Kamat. FPGA pagrįstas duomenų paketų skaidytuvas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2008. – No. 8(88). – P. 15–18.

Pristatomas FPGA pagrindu sukurtas duomenų paketų skaitytuvas, sukurtas naudojant projektavimo įrankių sistemą „Handle–C DK4". Projektuojant tarpusavyje buvo derinami skirtingi įvairių gamintojų įrankiai, skirti RTL modeliui imituoti, testuoti, klaidoms aptikti ir joms šalinti. Aptartojo paketų skaidytuvo branduolys gali būti naudojamas atliekant pasyviąją informacinio tinklo stebėseną saugumo sistemų įtaisuose ir t.t. Il. 2, bibl. 8 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).

DOI: 10.5755/j02.eie.11225