Characterization of Power-Aware Reconfiguration in FPGA-Based Networking Hardware

Sándor Plósz, István Moldován, László Kántor, Tuan Anh Trinh

Characterization of Power-Aware Reconfiguration in FPGA-Based Networking Hardware

Sándor Plósz[1], István Moldován[1], László Kántor[2], Tuan Anh Trinh[1]
Budapest University of Technology and Economics
Dept. of Telecommunications and Media Informatics [1], Aitia International[2]
Budapest, Hungary
{plosz, moldovan, trinh}@tmit.bme.hu
lkantor@aitia.ai

Abstract. Dynamic reconfiguration of FPGA in the networking hardware device is a feature which can be exploited in numerous networking applications. By reconfiguration we can change either the functionality, performance or even energy consumption of an area on the FPGA. This property can be exploited in a number of ways, in different application areas. However, the question of performance and power consumption trade-off in these situations is still an open issue. In this paper we address this issue by investigating different use cases and introducing a general approach of algorithmic optimization of power consumption based on dynamic reconfiguration. Our findings are supported by extensive SystemC/TLM hardware level simulations.

Keywords: Networking; hardware; reconfiguration; power optimization; FPGA.

1 Introduction

Handling high speed traffic requires high performance, which is a term quite often competing with low power consumption. The challenge of high-performance, yet low-power systems is to find an optimal balance of shutting down resources that are not in use. To shut down and wake them up when needed may not result in lower cost (power) comparing to not even shutting them down in the first place. Optimization for low power with high performance is a key requirement during the definition and design of state-of-the-art communication systems. With the ever-increasing demand of faster data transmission currently the 10 Gigabit per second Ethernet (10 GbE) service is becoming a requirement on the provider side network. When data arrives to a system at 10 Gigabit per second rate, the time is very limited for analysing or handling it. To optimize data processing at many different levels, tasks should be distributed and made parallel.

The Scalopes C-Board was created to overcome these issues featuring high processing power but also efficient resource usage. For high-speed packet processing FPGAs are used and the board was extended with a general-purpose processor for management functions. The four Virtex-5 FPGAs used in the Scalopes C-board support dynamic partial reconfiguration. This feature allows multiple design modules to time-share physical resources. Partially reconfigurable modules can be swapped on-the-fly, even while the base design continues to operate. The FPGA technology helps building a multi-purpose hardware, a simple firmware switch enables switching between applications much faster than if we needed to switch the whole device.

In this paper we investigate the applicability of networking system reconfiguration. We use a simulation tool to investigate the performance trade-off and the usefulness of reconfigurability according to different metrics.

In Section 2 we introduce what dynamic, partial reconfiguration is really about, how it works and how this feature can be exploited in communication systems. In Section 3 we present different algorithms for this use-case and describe the parameters which affect their operation. In Section 4.1 the realization of the simulation scenarios are presented along with the simulation environment. Results of the simulations are presented in Section 4.2. Finally Section 5 concludes the paper.

2 Dynamic, partial reconfiguration

Reconfiguration is a mechanism that makes it possible to change the configuration of a reconfigurable hardware, such as an FPGA. The reconfiguration can be full or partial: during partial reconfiguration only a part of the FPGA is reconfigured while the other part remains unchanged. This method supposes well defined interface points to the unchanged part. Dynamic reconfiguration means that partial reconfigurations are done on-the-fly, triggered by events related to the operation of the whole system. This method requires sophisticated management of interface points in the system to preserve process states. Dynamic reconfiguration implies that reconfiguration should be fast and should have limited impact on the performance of the whole system.

2.1 Hardware level reconfiguration mechanism

Partial reconfiguration requires that reconfiguration bitstream files that store the hardware reconfiguration option are available in advance. Usually they are stored in the DDRAM memory. Each FPGA has a DDRAM memory connected, and besides the communication related functions they can also contain reconfigurable IP (Intellectual Property) cores.

Bitstream-based partial configuration can be done using the ICAP (Internal Configuration Access Port) that is already loaded in the initial configuration stream. A Partial Reconfiguration (PR) controller is required to initiate the reconfiguration. The job of a PR controller is to retrieve the partial bitstream from memory, then to deliver it to a configuration port. An internal PR controller loads partial configuration bitstreams through the ICAP interface. As with any other logic in the static design, the internal partial reconfiguration control circuitry operates without interruption throughout the partial reconfiguration process.

The speed of reconfiguration depends on many factors. The most trivial is the speed of the source interface (PCI-express), the ICAP component which writes, and loads the frames to the FPGA configuration RAM, and the size of the cores. The C-Board design has a 125Mhz/64bit user data bus on the PCI-express interface. This results an approx. 8 Gb/sec throughput in theory. Practically, data throughput is around 7.2-7.6 Gb/sec depending on header/payload size and PC system configuration.

The necessary amount of transferred data also determines the overall performance. Differential core sizes vary depending on application complexity from 40-100 Kbytes to 1-2 Megabytes (almost full sized cores). Unfortunately, except some marginal cases binary file compression does not increase performance, due to the time consuming decompression process.

Above all, the real bottleneck is the operation speed of the ICAP controller. The component can operate in x8, x16, and x32 input/output bus width-modes. Input clock rates theoretically vary from 200 KHz to 200 MHz by Virtex5 devices. During our further investigation we have considered a XPS_HWICAP structure amended with a DMA controller in a Virtex FPGA introduced in [6] where the authors state an average reconfiguration speed of 82.1 MB/s achieved by simulation using this structure.

2.2 Applications of hardware reconfiguration in networking

In the networking area there are numerous applications for the reconfigurable hardware. Dynamic reconfiguration of hardware, however, is a new topic of interest. Dynamic reconfiguration has been used for several networking applications in the literature such as in firewalls [1], intrusion detection systems [3], on-demand configured cryptographic and multimedia accelerators [2], enhanced packet processors [4] and for reducing the power consumption. Based on the reconfiguration time requirements, the following application fields are present:

• *User-requested reconfiguration*. In this scenario the user initiates a reconfiguration of the hardware, loading an IP core with different characteristics. The reconfiguration itself can be automatic, for example a result of a setting change on the graphical user interface. An example application is the change of the protocol-optimized monitoring IP core to another protocol. The reconfiguration time has limited importance here.
• *Per-flow on-demand reconfiguration.* The reconfiguration takes place in order to meet the requirements of the new flow. Example applications are changing the cryptographic core or video codec to meet the requirements imposed. Reconfiguration time is now important, but it should only affect the new flow.
• *Run-time reconfiguration of the packet processing logic.* Reconfiguration is triggered by a signal or by reaching a load threshold. In this case the reconfiguration time is critical; all packets handled by the module are affected.

In this paper we target on the last application area, where the delays introduced by reconfiguration can be compensated by buffering. However, the buffer sizes have constraints on both networking and hardware side: some networking applications have low delay and jitter requirements, while in hardware the memory required to buffer the packets is limited.

3 Networking address lookup algorithms

Address lookup can be realized by numerous algorithms. The goal of these algorithms is to find an entry in the so-called lookup table witch satisfies a given input criteria. The lookup table can change dynamically by adding, deleting or modifying entries; it is the duty of the lookup algorithms to realize these functions to maintain the table. The lookup table is usually stored in memory to allow fast access. The table itself realizes an overlay structure above the memory and the mapping of the table to the physical part of the memory is not the task of the lookup algorithms.

The simplest search method is the linear search, where the lookup is performed starting from the beginning of the table and progressing continuously comparing the entry in the pointed part of the table with the input entry. Other more complex algorithms use techniques like hashing and heuristics to increase lookup speed resulting in faster but more power consuming or increased storage complexity algorithms compared to the simple linear search.

In this work we have selected three different algorithms which can realize packet lookup: a Binary search, the HiCuts (Hierarchical intelligent Cuttings) [10] and the TCAM (Ternary Content Addressable Memory) based lookup. The Binary search algorithm maintains an ordered structure, where the search can be performed in binary decisions from the middle of the table.

The HiCuts algorithm works by cutting the state space along the number of classifiers (dimensions) until each leaf has no more entries than a predefined number. Based on this method entries can be stored in multiple leaves, therefore it has exponential space requirement (static consumption) depending on the number of classifiers in the entries.

The TCAM is a fully hardware accelerated algorithm which can be implemented in FPGA platform. It can find an entry in the table in one clock cycle independent of the size of the table [8]. In that time it consumes massive power which is proportional of the number of entries stored in the table. In fact the consumption is a linear function of the number of cells occupied by the implementation and of the frequency [7].

The average power consumption of these algorithms can be calculated as follows:

$$E_{sum} = E_{static} + E_{dynamic} + E_{mgmt}, \text{ where}$$

the static consumption (E_{static}) expresses the consumption of the algorithm for storing its table in the memory. The dynamic consumption ($E_{dynamic}$) relates to the operational consumption of the algorithm in case of lookup request. The management consumption (E_{mgmt}) is connected with maintaining the lookup table after entry deletion or addition. Since the modification of the lookup table by adding or deleting entries is a rare event in the provider core network the management consumption is negligible compared to the other two consumptions. Because of this we have omitted the management consumption from our simulations.

Table 1. Consumption complexity and response time of different lookup algorithms (proportional to memory accesses and sequential operations, respectively)

Algorithm	Consumption		Sum	Response time
	Static	Dynamic		
Binary	N	$\log_2 N$	$k*N + \log_2 N$	$C_1 * \log_2 N$
HiCuts(d)	N^{rd} $(r \leq 1)$	d	$k*N^r + d$	$C_2 * d$
TCAM	N	N	k*N + N	1

In **Table 1** we have collected the consumption complexity and response time of the examined algorithms [9]. All consumption values are proportional to memory operations needed in average and response time values are proportional to the number of sequential operations required for one request. Parameter **N** expresses the number of entries stored in the lookup table. Parameter **k** is the ratio between the static and dynamic consumption, while $\mathbf{C_x}$ are constants related to the operation of the algorithms in the memory. The exponential static consumption of the HiCuts algorithm represents the worst case scenario where all entries are present in all of the leaves. In practice the exponent is much less than **d** which is expressed by choosing parameter **r** properly.

All consumption values represent orders of magnitude in average and response time values represent orders of magnitude of clock cycles (proportional to memory accesses required).

If we consider a *power unit* the energy required to make one memory operation, we can express the power consumption of different algorithms in power units. In spite of showing concrete power numbers for a concrete hardware we will come up with percentages of power savings using reconfiguration in general by simulation.

The basis of the optimisation is to use an algorithm which is fast enough to serve the demand and has the minimal consumption among them. When there is less demand we can reconfigure the device to a slower and also less power consuming algorithm.

4 Simulations

The simulations were implemented in SystemC/TLM using the ReSP (Reflective Simulation Platform) [5] designed by the Technical University of Milano. This platform combines the flexibility of the Python language with the robustness of the C language. The simulation handling is dynamic due to the reflective properties of Python. Reconfigurability was an amendment to this platform.

4.1 Simulation Scenarios

In the simulations we have modeled a 4 port router with 10G interfaces in an aggregation (core) network. We use an adaptive algorithm to perform reconfiguration based on the incoming traffic intensity. The reconfiguration algorithm targets to

exploit the daily variation of the traffic level which can be observed in the core network where data is aggregated from dozens of users. The daily traffic variation is a slow process, but the traffic in off-peak hours is only a percent of the peak traffic. This slow variation can be tracked by a few reconfigurations with insignificant impact on the Quality-of-Service (jitter) but still providing significant power gain.

In the simulations we have reconfigured the lookup algorithm on an FPGA, which is responsible for packet lookup. The three selected lookup algorithms (described in Section 3) are assigned to traffic speed intervals based on their lookup speed. We supposed that the cores are available in advance and reconfiguration is done by a hardware assisted mechanism.

The parameters have been chosen as follows for the simulations. The number of entries (N) to 1000, the number of classifiers (HiCuts dimension) to 2, the constants (C_1, C_2) to 3 for reading the memory, comparing content, and reading address of next entry. Parameter r has been chosen for 0.6 which is less than 1 but still makes the table size close to the values achieved in simulations in [10] which is a good compromise.

As introduced in Section 3 we measure consumption in power units by choosing the k parameter for 1/50 meaning that the operational consumption of the algorithms is 50 times higher than in idle state. This ratio can be observed by many used memory types and it is also true for FPGA block RAMs.

The bitstream required to insert the entries into the memory is calculated by multiplying the number of entries with the size of one entry. The dimension was chosen to be 2, which by having two 4 Byte IPv4 addresses gives 8 Byte for an entry. Multiplying this with the number of entries results in 8 KB as the size of the table. According to the speed of the ICAP controller (80 MB/s) explained in Section 2.1 the reconfiguration time is about 100 ns. In case of the HiCuts algorithm the table size calculated is approximately 253 KB and the reconfiguration time is about 3.2 ms. In **Table 2** the parameters of the algorithms calculated for the simulations are summarized. For example, the consumption of the binary search is calculated as follows:

$$E_{bin} = \frac{1}{50} * N + \log_2 N = \frac{1000}{50} + \log_2 1000 = 20 + 9{,}96 \cong 30$$

Table 2. Calculated simulation parameters for the selected algorithms (from **Table 1** with selected parameters)

Algorithm	Consumption (in Power Units)	Max. speed	Reconfiguration time [ns]
Binary search	30	4,7 Gb/s	100
HiCuts(d)	82	6,9 Gb/s	3200
TCAM	1020	9,6 Gb/s	100

A sinusoid traffic generator has been implemented to simulate the changing traffic in a daily period. The values on this curve represent the expected values of a Pareto probability distribution, while packet sizes are chosen according to simple IMIX model widely used to simulate network traffic. The incoming traffic speed is calculated and given to an IIR filter to smooth out bursts.

4.2 Results

In the first simulation we have considered the daily traffic fluctuation approximated by a sinusoidal traffic varying between 3-8Gbps. In reality this should have a daily periodicity however we have reduced the period to 1 second only for simulation scalability reasons. Algorithms are configured according to their average throughput, with a hysteresis of 5% around the configuration threshold of each algorithm. Results are presented on Figure 1. The upper graph shows the traffic load. The middle graphs show the queue utilization and consumed power. The reconfiguration cost considered in our simulations is 2000 power units. When the traffic load level crosses an algorithm configuration threshold, reconfiguration occurs. During reconfiguration the arriving packets are buffered. The queue utilization is around zero most of the time, with short spikes around the configuration changes. After reconfiguration the queue is emptied with a faster algorithm. When a new threshold is crossed (either upper or lower) the most appropriate algorithm is configured again. The lower graph shows the selected algorithm in a particular moment.

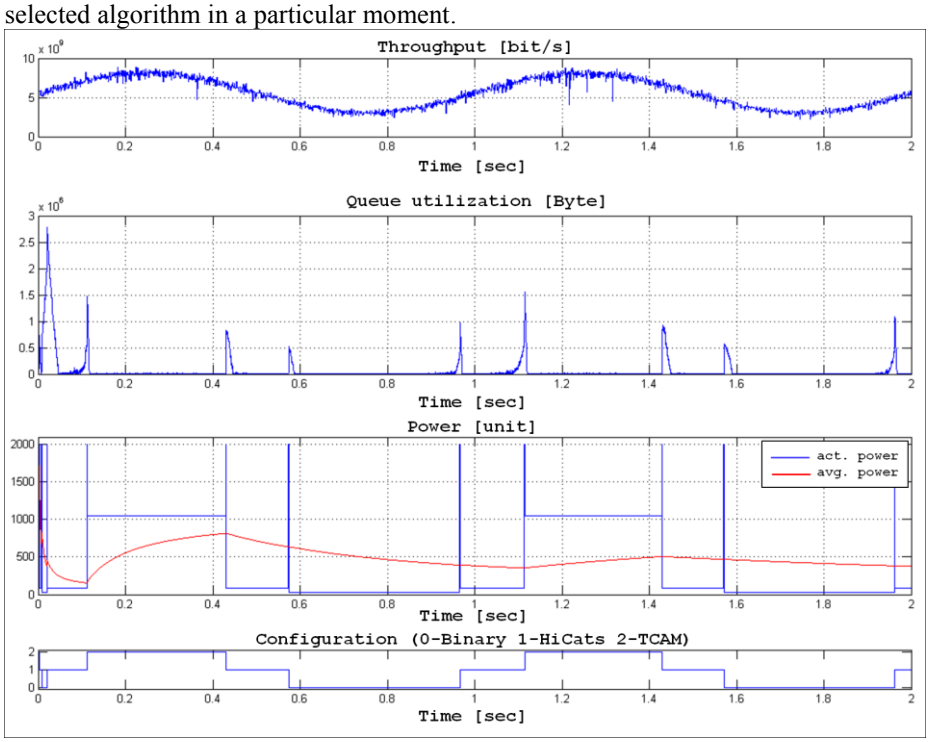

Figure 1 Throughput based reconfiguration

Beside actual power average power is also shown on the 3rd graph. If we do not use reconfiguration, the fastest (and most power hungry) algorithm must be used all the time. This is the TCAM which has a power consumption of 1020 power units. With reconfiguration the average power consumption is less than half (below 500 in long-term) indicated by the red line on the graph.

In the second simulation we show that higher power gain can be achieved if we use buffer level based reconfiguration instead of traffic intensity based. In this case

reconfiguration occurs whenever the buffer utilization crosses a threshold (selected to 50kbytes, 200kbytes and 500kbytes respectively). The new algorithm with its higher throughput will empty the buffer, and when the buffer utilization gets below the lower threshold of 50kbytes, the slower algorithm is configured again. Before reconfiguration the possible energy gain is also evaluated and a reconfiguration is carried out only if the gain is higher than the cost of reconfiguration. Figure 2 shows the results in case of buffer based reconfiguration. The offered traffic has a 7Gbps load with Pareto distribution. This configuration uses 2 algorithms only, a faster TCAM based and the slower HiCuts. The HiCuts algorithm is unable to handle this much traffic, which results in increasing buffer utilization. When the TCAM based algorithm is then configured the buffer starts to decrease quickly as the TCAM lookup speed is faster than the load. During the reconfiguration the buffer is increasing fast (almost vertical increase between 3Mbytes and 4Mbytes).

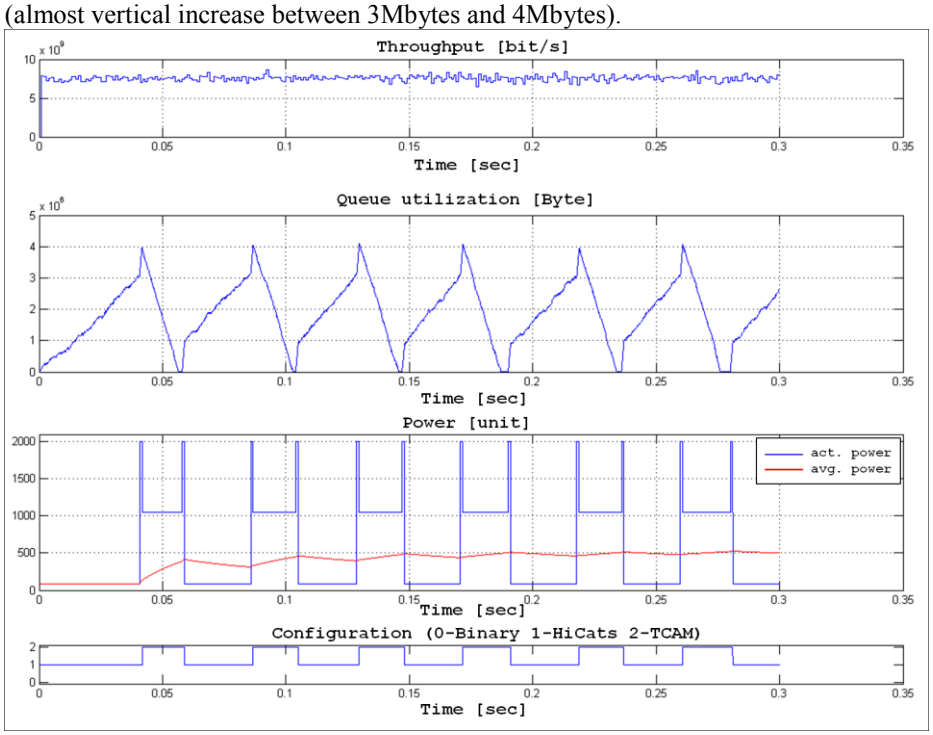

Figure 2 Buffer threshold based reconfiguration

The introduced delays are proportional to the queue utilization, the jitter in this case is determined by minimal and maximal buffer utilization, and it is in range of 0-4.5ms. This jitter is considerable, and may not be tolerable by some applications.

To reduce the jitter introduced by buffering we can limit the lower threshold on the buffer utilization for each algorithm for the desired jitter limit. This of course results in lower gain in energy, but also in lower jitter.

In the simulations we have used a jitter limit of 2ms. Power consumption for different loads and different reconfiguration schemes is shown on Figure 3. As we can see, without reconfiguration (blue line) the power consumption is a bit more than 1000 power units. In case of jitter-limited (green line) scheme the gain is determined

by the algorithm properties only, and only a few reconfigurations are allowed. Maximal gain (red line) can be achieved if algorithms are changed dynamically, and in this case further gain can be achieved at the cost of an accepted jitter (which can be limited, as an input to the scheme).

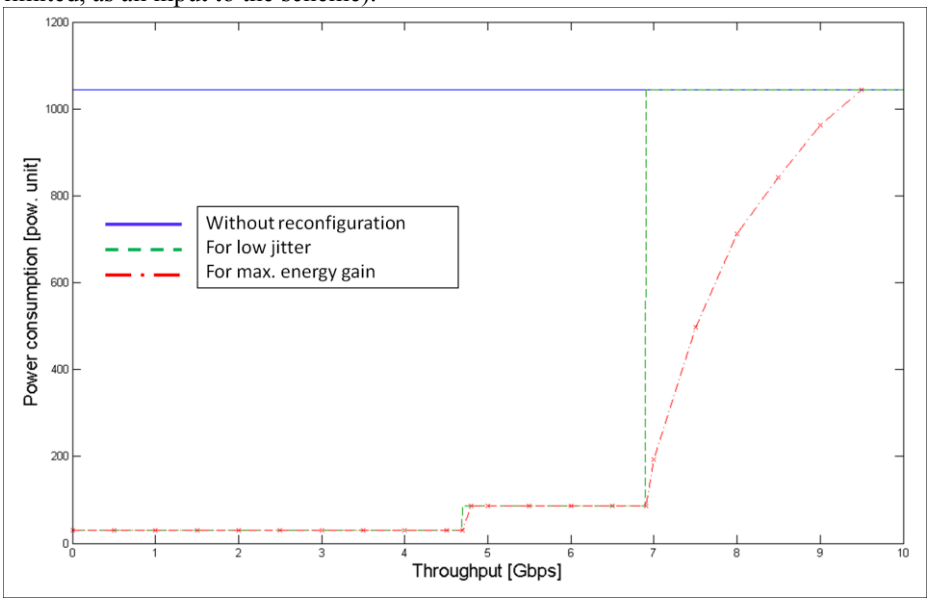

Figure 3 Power consumption for different reconfiguration schemes

5 Conclusions

Partial reconfiguration is a very interesting technique which has many benefits. It can be used to gain space on FPGA, to reduce power consumption and it also can be used in specific network-related applications like lookup tables or filtering rule sets.

We have presented the partial reconfiguration possibilities on the C-board, and also we have identified the most important components of the reconfiguration.

The possible use cases for partial reconfiguration have been discussed and we have investigated by simulation the applicability of dynamic reconfiguration in a high speed networking environment. Results show that dynamic reconfiguration can be used to reduce power consumption by adapting to the load levels, with an increased jitter penalty. The daily traffic fluctuation can be exploited by minimal impact, but considerable higher gain can be achieved by allowing buffering and rapid reconfigurations at a higher but controlled jitter penalty.

The reconfiguration process inevitably introduces delays, and further delays are introduced while maximizing the energy gain. The impact of the delays on network traffic can be simulated in network simulators, which take into consideration both the transport protocol and application specific behaviour.

6 Acknowledgement

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n° 100029 and from the Hungarian National Office for Research and Technology (NKTH).

This work is also connected to the scientific program of the "Development of quality-oriented and cooperative R+D+I strategy and functional model at BME" project. This project is supported by the New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002).

7 References

1. John W. Lockwood et al., “An Extensible, System-On-Programmable-Chip,Content-Aware Internet Firewall”, Field Programmable Logic and Applications (FPL), Lisbon, Portugal, Paper 14B, Sep 1-3, 2003
2. John W. Lockwood et al., “Internet Worm and Virus Protection in Dynamically Reconfigurable Hardware”, (MAPLD), Washington DC, 2003, Paper E10, Sep 9-11, 2003
3. Baodong Yu Xuecheng Zou, “PKI based dynamic reconfiguration home network media centre”,, Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, 28-30 May 2005.
4. Christoforos Kachris, Stamatis Vassiliadis, “A Reconfigurable Platform for Multi-Service Edge Routers”, SBCCI’07, September 3–6, 2007, Rio de Janeiro, RJ, Brazil.
5. ReSP simulator, “http://www.resp-sim.org/”
6. M. Liu, W. Kuehn, Z. Lu, A. Jantsch, “Run-Time Partial Reconfiguration Speed Investigation and Architectural Design Space Exploration” Field Programmable Logic and Applications, 2009. FPL 2009, Prague, Czech Republic
7. Y. Chen, O. Oguntoyinbo, “Power efficient packet classification using cascaded bloom filter and off-the-shelf ternary CAM for WDM networks”, Computer Communications Volume 32, Issue 2, 12 February 2009, Pages 349-356
8. H. Le, V. K. Prasanna, “Scalable High Throughput and Power Efficient IP-Lookup on FPGA”, 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines, Napa, California
9. P. Gupta, N. McKeown, “Algorithms for Packet Classification”, Network, IEEE, Volume: 15, Issue 2, Pages 24-32
10. P. Gupta, “Hierarchical Intelligent Cuttings: A Dynamic Multi-dimensional Packet Classification Algorithm”, PhD thesis, Chapter 5, Online: “http://klamath.stanford.edu/~pankaj/phd.html”