21st Iranian Conference on Electrical Engineering (ICEE2013)

Exploring the Design Space for Area-Efficient Embedded VLIW Packet Processing Engine

M. Hassan Najafi and Mostafa E. Salehi
School of Electrical and Computer Engineering, Faculty of Engineering, University of Tehran, Iran
{mhassannajafi, mersali}@ut.ac.ir

Abstract

Today area-efficiency is an important factor in designing embedded systems. This paper presents a design space exploration based on an embedded VLIW processor for finding out the optimum architecture for ever increasing demands of embedded packet-processing applications. In our exploration, we use the VEX toolchain for exploring the effects of memory hierarchy, different architectural configurations, and compiler optimizations on both performance and area. Exploration results will find out the best architecture and compiler optimizations for VLIW embedded packet- processing engines to have area-efficiency as well as time-efficiency.

Aims of research

We present a design space exploration to find the area-efficient VLIW processor architectures to be used in network domain. Our goal is selecting the best VLIW configurations that save area and execution time.

Materials and Methods

In this paper, we select:

- **The VEX toolchain which models a scalable platform for exploring embedded VLIW processors design space.**
- **Mibench and Packetbench network representative benchmarks: Dijkstra, CRC32, IPv4-radix, IPv4-trie, Flow Classification, IPSec Encryption**

We change some of the VEX Default architecture parameters and memory architecture as well.

To reduce the large design space we limit our exploration to most important VEX parameters such as issue width, number of clusters, ALU per clusters, multiply per cluster, integer registers, branch registers, and instruction and data cache size.

We start from default vex value for these parameters and try to study the influence of increasing or decreasing the value of these parameters. So in this way we succeed to prune our configurations set from 108 entries to 14.

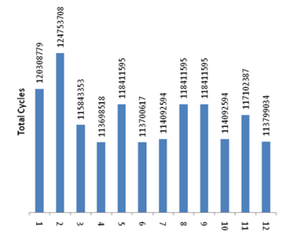
Exploiting a logic synthesis tool and a 90nm standard cell library, we have synthesized the default VEX architecture and calculated the area of different parts of the VEX processor. In addition, to compute the area of caches with different sizes we used CACTI tool and MCPAT.

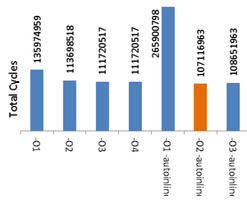
To define a good design space for each benchmark, we increase instruction and data cache size until the cache miss rate becomes less than 2%.

Results

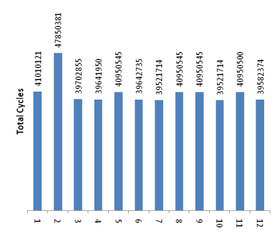
Selected machine configurations.

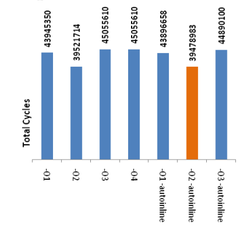
# config	BR*	GR*	MUL*	ALU*	Issue Width*	Cluster
1	8	16	1	2	2	1
2	8	16	1	2	1	1
3	8	16	1	4	4	1
4	2	16	1	4	2	1
5	8	16	1	2	2	2
6	8	16	1	4	4	2
7	8	16	1	4	2	2
8	8	16	2	2	2	2
9	8	32	1	2	2	2
10	2	16	1	4	2	2
11	8	16	1	2	2	4
12	8	16	1	4	2	4
13	8	64	1	4	2	4
14	2	16	1	4	4	4





Dijkstra Execution time and the optimum based on config 4





CRC32 Execution time and the optimum based on config 4

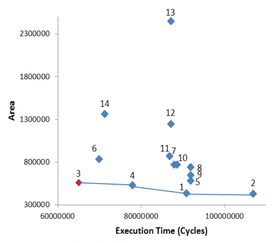


Figure 1-IPSEC performance per area.

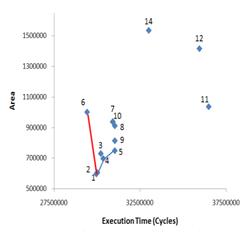


Figure 2-IPV4.radix performance per area.

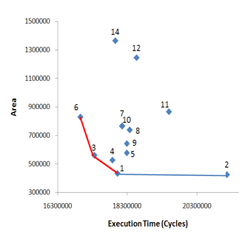


Figure 3-IPV4-Lctrie performance per area.

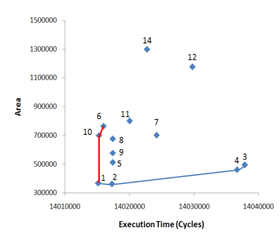


Figure 4-Flow-Class performance per area.

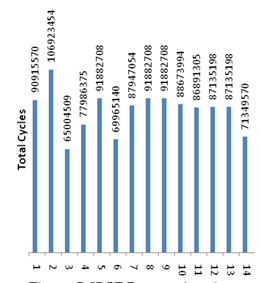


Figure 5-IPSEC execution time.

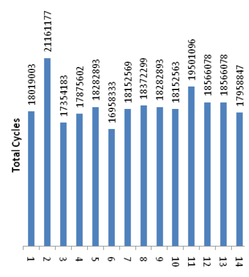


Figure 6-IPV4.radix execution time.

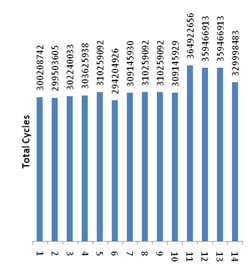


Figure 7-IPV4-Lctrie execution time.

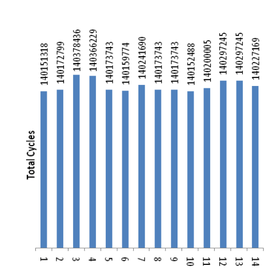


Figure 8-Flow-Class execution time.

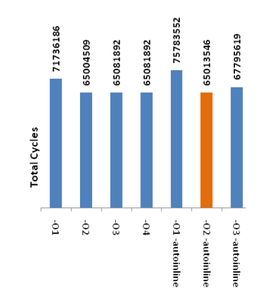


Figure 9- Optimized IPSec based on Config 3.

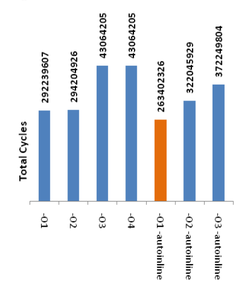


Figure 10- Optimized IPv4-Radix with config 1.

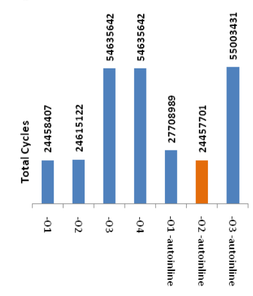


Figure 11- Optimized IPv4-Lctrie with config 1.

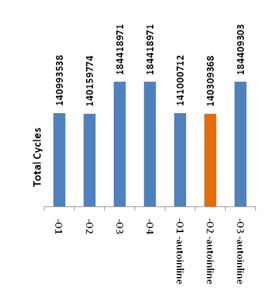


Figure 12- Optimized Flow-Class with config 1

Red and blue lines are the area and performance pareto-optimal lines. The blue line introduces us the range of configus that save area and the red line presents the best configs to produce minimum execution time.

Conclusion

We purpose to choose 4KB D and I cache. Configuration 3 and 6 yields the best execution time and configuration 1 leads the minimum area. We conclude that if the major design concern is execution time, we must increase the number of ALUs and Issue widths up to 4, without exceeding the number of clusters over 2. If we are limited by area, we would use one cluster with two ALUs and two issue widths without changing in other default parameters. In addition, for optimizing the source code we found that the composition of "-O2" and "-autoinline" flag can give us the best performance.

References

[1] P. F. J. Fisher, and C. Young, Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools: Morgan Kaufmann, 2004.

[2] Hewlett-Packard Laboratories. VEX Toolchain. [Online]. Available: http://www.hpl.hp.com/downloads/vex/

[3] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli, "Design Space Exploration of Network Processor Architectures," in In Network Processor Design: Issues and Practices vol. 1, ed, 2002, pp. 30--41.

[4] R. Ramaswamy and T. Wolf, "PacketBench: a tool for workload characterization of network processing," in Workload Characterization, 2003. WWC-6. 2003 IEEE International Workshop on, 2003, pp. 42-50.