FPGA implementations of a parallel associative processor with multi–comparand multi–search operations

Zbigniew Kokosiński and Bartłomiej Malus
Cracow University of Technology,
Faculty of Electrical and Computer Eng.
ul. Warszawska 24, 31-155 Kraków, Poland
zk@pk.edu.pl

Abstract

Multi–comparand associative processors are efficient in parallel processing of complex search problems that arise from many application areas including computational geometry, graph theory and list/matrix computations. In this paper we report new FPGA implementations of a multi–comparand multi–search associative processor. The architecture of the processor working in a combined bit–serial/bit–parallel word–parallel mode and its functions are described. Then, several implementations of associative processors in VHDL, using Xilinx Foundation ISE software and Digilent development boards with Xilinx FPGA devices are reported. Parameters of the implemented FPGA processors are presented and discussed.

Keywords: parallel associative processor, SIMD, FPGA processor, bit–parallel processing, multiple–search

1. Introduction

Searching is a key concept in computer science and engineering [6]. Various searching problems arise in different application areas. Increasing size of the searching tasks often becomes critical. Therefore, new techniques that can speed up search operations are required. An efficient solution for new demanding applications is massively parallel computing.

Associative parallel processors (content addressable processors) of the SIMD type are particularly well suited for performing fast parallel search operations in many search–intensive algorithms. They provide fast parallel search for the given content (pattern) in a data array. The results of the exact match/mismatch for all data records are stored in a tag vector and can be further interpretted.

Advances in associative processing include applications in computational geometry, relational databases, and artificial intelligence [5], selection of extreme search algorithms for fully parallel memories [14], multiassociative/hybrid search [4], an experimental implementation of the multi–comparand multi–search associative processor [8], the concept of an associative graph processor [13] and parallel algorithms for search problems from various complexity classes [7, 10, 11, 12].

In this paper a design of FPGA associative processor is reported, that implements the idea of multi–comparand search [2, 7, 8, 13]. All projects were performed in VHDL with Xilinx Foundation v. 8.1.i software. We used low cost Digilent development boards with standard Xilinx FPGA devices. The obtained results illustrate a steady progress on the way to take advantage of associative processors implemented in FPGA technology.

In the next section the processor architecture is described. In section 3 an application of the multi–comparand processor to range searching problems from computational geometry is presented and basic algorithms for their solving are given. Section 4 contains implementation data of the developed FPGA prototypes. Conclusions derived from the reported research are sketched in section 5.

2. The processor architecture

The multi–comparand associative processor may be described as follows [8].

The main components of the processor are data array and comparand array. The conventional single–comparand register is replaced by a comparand array CA, and the tag array TA is extended to the size of the Carthesian product of data and comparand sets. Basic data processing is organized in bit–serial word–parallel mode. Processing is performed neither in the data array DA, nor in the comparand array CA, but exclusively in a special tag array. The multiple-comparand associative memory with 2D tag memory TA is a key module of the processor. The memory implements up

978-0-7695-3472-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ISPDC.2008.42

to 16 different logic search functions. In order to extract global features of processed data, associative search capability is introduced into the 2D tag memory too. Thus, the associative search is organized hierarchically, on the two levels.

In each consecutive step of processing each tag memory cell is updated according to its previous state, the corresponding data/comparand values, and so called prescription function that defines type of the search (in general 24 logical and 10 arithmetical searches are valid [2]). Type of the search determines also the initial state of the tag array TA. The possibility to define a prescription function for each tag cell separately is a source of the memory's versatility but it requires an extra logic and $\lceil log_2 q \rceil$ control bits for each cell, where q is the number of different prescription functions. Obviously, presumptively homogenous cell functions could be handled in a less hardware consuming way, reducing the number of control bits.

Applying current technology, the processor's tag array TA can be implemented as a reconfigurable logic, with configuring data loaded from a PRAM, providing hardware savings and an essential improvement of time parameters. Comparison results collected in the tag array TA have to be further processed in order to extract global search results. Otherwise, partial immediate data collected in the tag array TA are difficult to interpret. This implies a need of providing the 2D tag array TA with basic associative processing capabilities (exact match/mismatch), an extra mask register, a comparand register and a tag register of the second level.

As a result of the above considerations simple multi-comparand associative processor architecture is obtained that is shown in Fig.1. and consists of the following components:

DATA ARRAY (DA) — n × p binary data matrix;

COMPARAND ARRAY (CA) — m × p binary comparand matrix;

DM — n × 1 data and tag (TA and T2) mask vector;

CM — m × 1 comparand mask vector;

SM1 — 1 × p mask vector for multi–comparand search;

(DM, CM and SM1 select proper submatrices of DA and CA for multi–comparand search)

BPC — bit position counter; (generates consequtive bit slices selected by SM1 for bit–serial processing; bit slice l is generated by bit BPC[l]=1; for all j $\neq$ l: BPC[j]=0)

TAG ARRAY (TA) — n × m binary tag matrix; (memory for processing and storing current comparison results); each cell TA[i, j] process and stores results of comparison of subsequent pairs of bits {DA[i, l], CA[j, l]}, where l, $1 \leq l \leq p$, is the coordinate of the processed bit slice, according to a prescription function loaded into TA from PRAM PFG; TA has built–in capability of fully parallel (bit–parallel word–parallel) associative processing its content with single comparand C2 and search mask SM2 — results of single comparand EQUALITY search in TA are stored in T2; TA can also perform SELECT FIRST function [3] in each column)

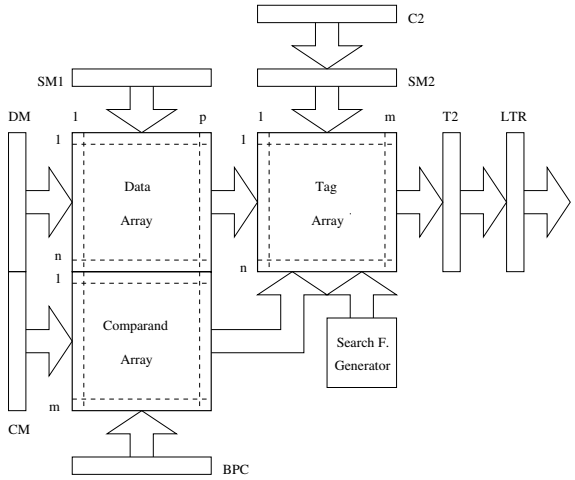

Figure 1. Architecture of the processor.

PRAM PFG — PRAM prescription function generator; (for our needs the number of prescription functions is 6, 12 or 16 but in general this set can be freely modelled for a given application)

C2 — 1 × m binary comparand vector;

SM2 — 1 × m TA mask vector;

T2 — n × 1 binary tag vector for TA;

LTR — logical tag resolver; (in particular, it computes binary variables w and u : w = 1 iff at least one unmasked bit of T2 is equal 1, while u = 1 iff all unmasked bits of T2 are equal 1; variable w, often denoted as SOME/NONE, is commonly used in tag resolvers [3, 5]; variable u, which can be denoted as ALL/NOT ALL is used in [5]).

The architecture is easy scalable since its basic components have very regular structure. The above processor architecture is described in a basic configuration. Depend-

ing on particular application it is necessary to add a number of the optional architecture components like counter of respondes, select first, (p,k)–combination generator [1], (p,k)–permutation generator, controllers for sequencing processor operations in TA and several additional registers [8].

The machine model is powerfull and flexible enough to satisfy numerous requirements in fast associative processing. The model is a parametrized generalization of many other simpler models proposed earlier and it can substitute them functionally, i.e. it can process all algorithms derived for those models. For instance, models of single–comparand processors are equivalent to our model with the parameter m=1. In many cases, where multi–comparand searching is crucial, the model provides a significant improvement of algorithms' performance.

The proposed architecture satisfies the following design assumptions:

1. single bit–serial search operation lasts a unit time;
2. all bit–serial word-parallel search operations last a time proportional to the number of bit slices, i.e. $O(p)$;
3. TA prescription functions can be loaded from PRAM PFG in $O(1)$ time;
4. single mask programming is performed in $O(1)$ time;
5. single comparand permutation is performed in $O(1)$ time;
6. LTR, "select first" circuits and counter of the number of respondes consume a time that depends on the construction of these components and available technology. In general that time is a logarithmic function of n.

3. Basic range searching algorithms

The multi–comparand multi–search associative machine is a powerful tool suitable for procesing complex search problems in many application areas. In this section we are going to demonstrate the idea of parallel associative search on an exemplary problem from computational geometry, i.e. Geometric Range Search [9].

There are many variants of geometric search problems. We will consider two of them:

Problem A
Given a system of ranges of the d–dimentional Euclidean space R^d and a given n–point set P in R^d find for the given range R whether all points of P belong to R.

Solution
For this decision problem the conventional sequential algorithm checks for each point $p \in P$ if its coordinates satisfy the corresponding set of relations. Having n points and $O(d)$ relations the required complexity of the search is clearly $O(dn)$. In a simple associative processor with fully parallel word processing it is possible to check simultaneously all relations for one point in a single step. Thus, the total processing time is reduced to $O(n)$ steps. In our multi–comparand multi–search processor it is possible to process in parallel all n points in the bit–serial word–parallel mode, therefore the resulting processing time for the *Problem A* is only $O(p)$, where p is the number of bits used for the given number representation.

Algorithm A
1. DA $\leftarrow$ P
2. CA $\leftarrow$ R
3. DM, CM, SM $\leftarrow$ 1
4. compute TM for the given system of relations
5. C2 $\leftarrow$ 1
6. SM2 $\leftarrow$ 1
7. compute T2
8. if u=1 then return YES else return NO

Problem B
Given a system of ranges of the d–dimentional Euclidean space R^d and a given n–point set P in R^d find for the given range R the number of points of the set P that belong to R.

Solution
Similarly like in the previous example it is possible to process in parallel all n points with the processing time $O(p)$. Then the numer of respondes have to be computed by a specialized circuitry in $O(logn)$ time.

Algorithm B
1. NUMBER=0
2. DA $\leftarrow$ P
3. CA $\leftarrow$ R
4. DM, CM, SM $\leftarrow$ 1
5. compute TM for the given system of relations
6. C2 $\leftarrow$ 1
7. SM2 $\leftarrow$ 1
8. compute T2 and NUMBER OF RESPONDES
9. if w=1 then NUMBER=NUMBER OF RESPONDES
10. return NUMBER

4. FPGA designs

In this section the implementations of the associative multi–comparand processor are described with popular Xilinx Spartan 3 and Virtex 4 FPGAs.

4.1. Designs for Spartan 3 FPGAs

For basic prototyping Xilinx Foundation ISE v.8.1.i software and Digilent Spartan 3 board with Xilinx XC3S200 FPGA device were used. The Spartan 3 device with density 200k gates contains 1920 slices (3840 slice Flip–Flops and 4–input LUTs) and 173 bounded IOBs. In all processor implementtions six basic logic searches were built-in, i.e. $\{=, \neq, <, >, \leq$ and $\geq\}$.

For reference we used the first FPGA prototype of such processor [8], developed for Xess XS40 board with Xilinx XC4005XL FPGA device. The maximum processor size was n=m=p=5, with maximum clock frequency 18.7 MHz, and 80 % logic utilization (157 out of 196 CLBs). Therefore, we started our designs from the processor size 6×6×6.

Prototyping work was twofold. The first option was to build the processor using area optimization strategy. The maximum size of the obtained prototype was 16×16×16 and the corresponding maximum clock frequency was 75.8 MHz. In that case 256 simultaneous comparisons were performed with the clock rate 13.2 ns per bit. The implementation details from utilization reports are given in Table 1.

An alternative option was to use area/speed optimization strategy. The maximum size of the obtained prototype was 18×18×18 and the corresponding maximum clock frequency was 90.0 MHz. In that case 324 simultaneous comparisons were performed with the clock rate 11.1 ns per bit. The implementation data are collected in Table 2.

In both cases area constraint ratio of 100 (+ 5) is improved in order to meet constraint for the designed processor block (the final ratio should be less then 100).

Asymptotic circuit complexity is $O(n^2)$ (it is dominated by TA size). It is observed that the clock frequency decreases slowly taking into account the processor size n.

The observed utilization of LUTs and IOBs never exceeds the utilization rate of slices.

If XC3S1500L Spartan 3 FPGA was used (which has the highest density in XC3S series) the estimated processor size could be 48×48×48, with 12533 out of 13312 slices used (utilization about 94%), 18508 out 26624 of LUTs (utilization about 69%), 70 out of 221 bounded IOBs (31 %) and maximum clock frequency 74 MHz.

Table 1. Associative processor parameters (XC3S200 FPGA, area optimization).

associative processor size	max. clock frequency	resource utilization			
	MHz	slices	%	LUTs	%
6×6×6	63.0	314	16	411	10
8×8×8	63.2	502	26	639	16
10×10×10	61.8	771	40	982	25
12×12×12	50.6	1064	55	1337	34
14×14×14	50.6	1396	72	1773	45
16×16×16	**75.8**	**1703**	**88**	**2069**	**53**
17×17×17	75.6	1942	101	2328	60
available resources		1920	100	3840	100

Table 2. Associative processor parameters (XC3S200 FPGA, speed optimization).

associative processor size	max. clock frequency	resource utilization			
	MHz	slices	%	LUTs	%
6×6×6	104	280	14	433	11
8×8×8	95.1	451	23	698	18
10×10×10	81.9	648	33	983	25
12×12×12	82.0	948	49	1469	38
14×14×14	77.9	1239	56	1900	49
16×16×16	96.8	1464	64	2164	56
18×18×18	**90.0**	**1810**	**94**	**2491**	**64**
19×19×19	89.6	2002	104	2808	73
available resources		1920	100	3840	100

Spartan 3 designs were carefully tested. Small instances of graph problems like CLIQUE, INDEPENDENT SET and SET INCLUSION problems were processed in our multi–comparand multi–search processor with associative algorithms described in [7]. The analysis of waveforms confirmed correctness of the designed devices.

4.2. Designs for Virtex 4 FPGAs

Considering application of fast Xilinx Virtex 4 FPGAs with speed optimization we designed our processor also for XC4VLX15 destination device with density 1500k gates, that contains 6144 slices (12288 slice Flip–Flops and 4–input LUTs) and 240 bounded IOBs.

We covered the whole range of processor sizes from 6×6×6 to 18×18×18 corresponding to the processor designed for XC3S200 device.

For the processor size 18×18×18 the corresponding maximum clock frequency was 198.0 MHz. In that case 324 simultaneous comparisons were performed with the clock rate 5.05 ns per bit.

According to device utilization summaries the designs for that range of processor sizes did not exceed 30 % of the available resources.

The observed utilization of LUTs and IOBs never exceeds the utilization rate of slices.

In addition the range of processor sizes from 20×20×20 to 35×35×35 was covered.

For the processor size 34×34×34 the corresponding maximum clock frequency was 145.6 MHz. In that case 1156 simultaneous comparisons were performed with the clock rate 6.87 ns per bit. The device utilization: 5837 out

Table 3. Associative processor parameters (XC4VLS15 FPGA, speed optimization).

associative processor size	max. clock frequency	resource utilization			
	MHz	slices	%	LUTs	%
6×6×6	230.1	286	4	433	3
8×8×8	214.9	461	7	698	5
10×10×10	187.7	663	10	983	7
12×12×12	189.0	971	15	1469	11
14×14×14	171.1	1269	20	1900	15
16×16×16	202.3	1501	24	2164	17
18×18×18	198.0	1878	30	2164	21
20×20×20	194.2	2264	36	2164	26
24×24×24	187.0	3295	53	4763	38
28×28×28	183.5	4318	70	6214	50
32×32×32	178.0	5423	88	7765	63
34×34×34	**145.6**	**5837**	**95**	**8217**	**66**
35×35×35	166.0	6290	102	8933	72
available resources		6144	100	12288	100

of 6114 slices used (utilization about 95%), 8217 out 12288 of LUTs (utilization about 66%), 76 out of 240 bounded IOBs (31 %).

In that case area constraint ratio of 100 (+ 5) is improved in order to meet constraint for the designed processor block (the final ratio should be less then 100).

The implementation data from device utilization reports are collected in Table 3.

5. Conclusions

In this paper, FPGA associative processor designs have been reported which enable highly parallel search operations. The processor can emulate many simpler associative machines used so far. The processor is versatile and flexible enough to meet a wide range of requirements and solve combinatorial problems from various complexity classes.

The feasible processor sizes implemented in Spartan 3 and Virtex 4 FPGAs are significantly bigger and faster then the first prototype built in 2002 [8]. It is now possible to process non–trivial application problems that involve intensive search operations.

6. Acknowledgements

The authors are gratefull to Sylwia Cieplińska, Krzysztof Dąbrowski and Grzegorz Forysiński of CUT, who developed the simulator of the associative processor as their student project.

References

[1] G. Bubniak, M. Góralczyk, M. Karp, and Z. Kokosiński. A hardware implementation of a generator of (n,k)–combinations. In *Proc. IFAC Workshop on Programmable Devices and Systems*, volume 228-231. IFAC, 2004.

[2] D. Digby. A search memory for many–to–many comparisons. *IEEE Transactions on Computers*, C-22:768–772, 1973.

[3] C. Foster. *Content addresable parallel processors*. Van Nostrand Reinhold, 1976.

[4] M. Herbordt and C. Weems. Associative, multiassociative and hybrid processing. In A. Krikelis and C. Weems, editors, *Associative processing and processors*, pages 26–49. IEEE Computer Society Press, 1997.

[5] A. Kapralski. *Sequential and parallel processing in depth search machines*. World Scientific, 1994.

[6] D. Knuth. *The art of computer programming. Sorting and searching*. Addison-Wesley, 1973.

[7] Z. Kokosiński. An associative processor for multi-comparand parallel searching and its selected applications. In *Proc. Int. Conf. Parallel and Distributed Processing Techniques and Applications PDPTA'97*, pages 1434–1442, Las Vegas, USA, 1997.

[8] Z. Kokosiński and W. Sikora. An FPGA implementation of multi-comparand multi-search associative processor. In *Proc. 12th Int. Conf. FPL 2002, Lect. Notes in Comp. Sci.*, volume 2438, pages 826–835. Springer–Verlag, 2002.

[9] J. Matoušek. Geometric range searching. *Computing Surveys*, 26:421–461, 1994.

[10] A. Nepomniaschaya. Associative parallel algorithm for dynamic reconstruction of a minimum spanning tree after deletion of a vertex. In *Proc. Int. Conf. PACT 2005, Lect. Notes in Comp. Sci.*, volume 3606, pages 159–173. Springer–Verlag, 2005.

[11] A. Nepomniaschaya. Associative parallel algorithm for dynamic update of a minimum spanning tree after addition of a new node to a graph. *Cybernetics and Systems Analysis*, 42:18–27, 2006.

[12] A. Nepomniaschaya. An efficient associative algorithm for multi-comparand searching and its applications. *Bulletin of Novosibirsk Computer Center: Computer Science*, 25:38–49, 2006.

[13] A. Nepomniaschaya and Z. Kokosiński. An associative graph processor and its applications. In *Proc. 6th Int. Conf. PARELEC 2004*, pages 297–302. IEEE Computer Society, 2004.

[14] B. Parhami. Extreme-value search and general selection algorithms for fully parallel associative memories. *The Computer Journal*, 39:241–250, 1996.