AREA OPTIMIZED SOLUTION FOR STRUCTURED ASIC DYNAMIC RECONFIGURABLE PLA

Traian TULBURE

Dept. of Electronic and Computers, "Transilvania" University of Brasov, Romania
E-mail: tulbure@vega.unitbv.ro

Abstract – **This paper proposes a new area optimized architecture for dynamic reconfigurable logic array with implementation on structured ASIC.**
Reconfigurable architectures allow the dynamic reuse of the logic blocks by having more than one on-chip SRAM bit controlling them. Thus, rather than the time needed to reprogram the function of the device from external memory which is the order of milliseconds logic block functions can be changed by reading a different SRAM bit which only takes time of order of nanoseconds.
Programmable Logic Array (PLA) structures can be implemented on Structured ASIC technology to eliminate the constraint given by fixed wire routing. Adding dynamic reprogramming to the structure implies adding a big distributed memory that affects both array utilization and device performance.
We are proposing in this paper a solution that uses the available block RAM memory to reduce area utilization and improve performance for dynamic reconfigurable PLA. The implemented PLA array presents dynamic reconfiguration with 16 configurations contexts for classical PAL 22v10 structure. The contexts are stored in inactive block RAM memory allowing fast context change.
The implementation results validate the proposed structure showing high-speed frequency operation and area reduction of about 30% compared with same PLA structure implemented with distributed memory.

Keywords**:** *reconfigurable computing, PLA, structured ASIC*

1. INTRODUCTION

Logic design capacity is conventionally measured in number of gates required to solve a particular problem. This metric of gate utilization is purely a spatial metric that does not consider the temporal aspect of gate usage. There can be cases when a gate is used only a small fraction of time it is employed. As described in [1] taking the temporal usage of a gate into account, we recognize that each gate has a capacity defined by its bandwidth. Reconfigurable structures try to make better usage of a gate bandwidth using dynamic reconfiguration during circuit operation and timing multiplexing techniques.
Structured ASIC is a technology between standard ASIC and FPGA that combines the benefits of both technologies. It has standard cell ASIC-like unit cost, power consumption performance and density, low up-front development cost, simple, FPGA-like design flow and device turnaround in only few weeks. Selected Structured ASIC (eASIC) has look-up table based logic cells while routing is fixed using single via metallization layer. Nowadays, the concept of reprogramming for structured ASIC means only changing the functions implemented by the LUTs. There is no way to modify the structure because the logic cell configuration and connections cannot be modified in time so very small changes can be made after the design has been manufactured. Programmable logic structures (PAL, PLA) can be generated with dedicated tools for structured ASIC that allows configuration change via bitstream load at power on but there no support for dynamic reconfiguration.
Previous work in this area was mainly focused in dynamic reconfiguration of FPGAs (DRFPGAs). Several different architectures have been proposed such as the Dynamically Programmable Gate Array (DPGA) [1], the Time Switched FPGA (TS-FPGA) [2] and Time Multiplexed FPGA(TM-FPGA) [3]. One solution for structured ASIC is the reconfigurable coprocessor described in [4].
These architectures allow the dynamic reuse of the logic blocks and wire segments by having more than one on-chip SRAM bit controlling them. Thus, rather than the time needed to reprogram the function of the FPGA from external memory which is the order of milliseconds logic blocks and interconnect can be changed by reading a different SRAM bit which only takes time of order of nanoseconds. Using the terminology in [5] each on chip configuration is called a context and a device with more than one context is called a multicontext device.
The main usage of the proposed architectures would be in the area of high performance reconfigurable computers (HPRC) [6] based on conventional processors and reprogrammable arrays but it could also fit wherever there is a need to reconfigure a part of logic for debugging or other purposes. The development of HPRCs has made substantial progress in the past years and near all important computing vendors have now HPRC product lines.
HPRC are parallel computing systems that contain multiple microprocessors and multiple

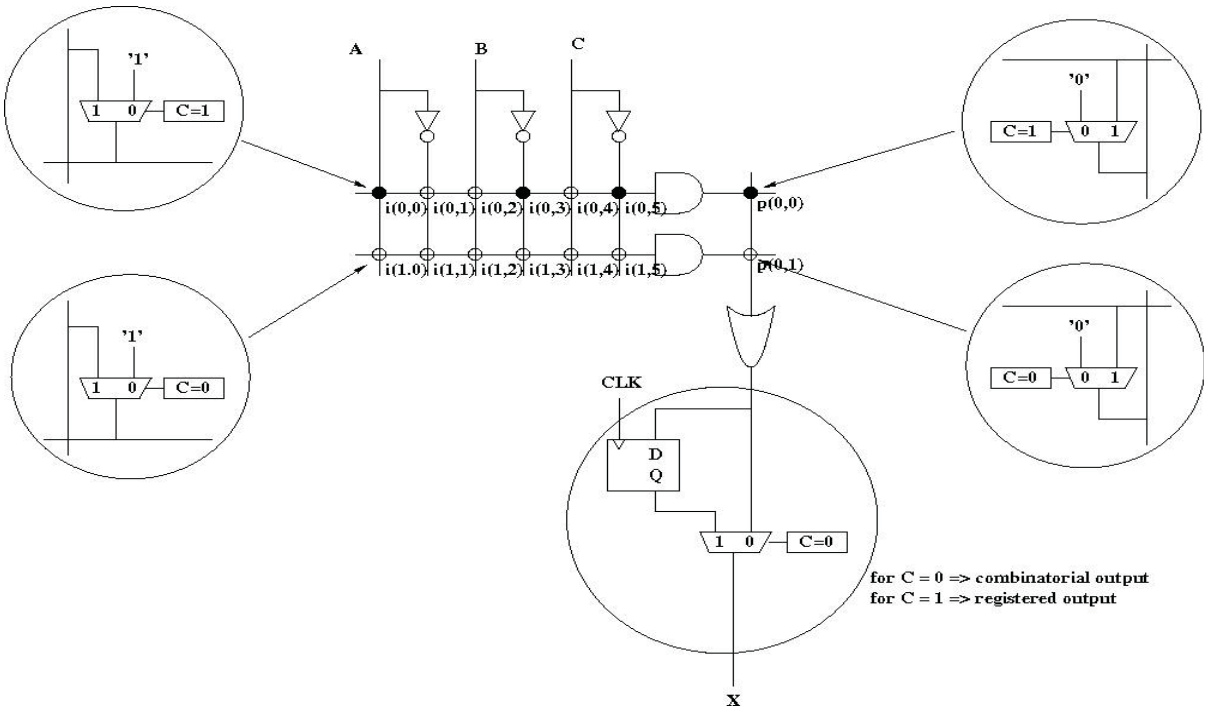

Figure 1. ePLA Internal Structure

reprogrammable arrays typically based on FPGA technology. In current settings the design uses FPGA as coprocessors that are deployed to execute the small portion of the application that takes most of the time. In theory any hardware reconfigurable device that can change their configuration under the control of a program can replace FPGA to satisfy the same key concepts behind this class of architectures. FPGA are the mostly used as reconfigurable structures but recently structured ASIC technology appeared as an alternative for designs that want to reduce costs and improve performance in terms of power and speed compared with the FPGA.

In this paper we are presenting a new area optimized solution for dynamic configuration of Programmable Logic Array (PLA) implemented on structured ASIC technology which combines advantages of multicontext devices with advantages of structured ASIC.

In Section 2 we present the proposed circuit architecture and Section 3 will depict the mapping to Structured ASIC. Finally results are presented in Section 4 and conclusions and further study are presented in Section 5.

2. THE ARCHITECTURE

This section describes two structures, ePLA and eTMPLA, that supplies a new feature to structured ASIC technology - dynamic reconfiguration without any additional hardware or layout change. ePLA is a dynamic reconfigurable structure which use the neutral element for AND/OR logic functions. Overlapping several ePLA will produce a Time Multiplexed ePLA (eTMPLA) which shares the AND/OR levels between several contexts.

Figure 1 show a simple ePLA structure with programming nodes controlled by memory elements for both AND/OR array and output type selection. The dynamic reconfigurable node is implemented with a simple multiplexer and a memory element that store the active configuration bit.

The ePLA described in the previous section does not share the AND/OR logic levels. Assuming that we want to implement n functions at different time moments, when using the ePLA structure we must implement n identical structures, for which only the configuration memory contents are different. By overlapping n ePLA structures and sharing the AND/OR logic level, a Time Multiplexed ePLA-eTMPLA will result. Figure 2 preset the structure of eTMPLA with multiple contexts.

Dynamic configuration of eTMPLA requires that each node is controlled by a small size memory. All memory instances need to be placed close to the corresponding node and they need to be able to switch simultaneously to change the active eTMPLA context. Using distributed memory can create problems because of increased fabric utilization and routing congestion for global signals that need to be driven to all memories.

In this paper we propose to use block RAM to implement context memory with the advantage of less area and less congestion and disadvantage that

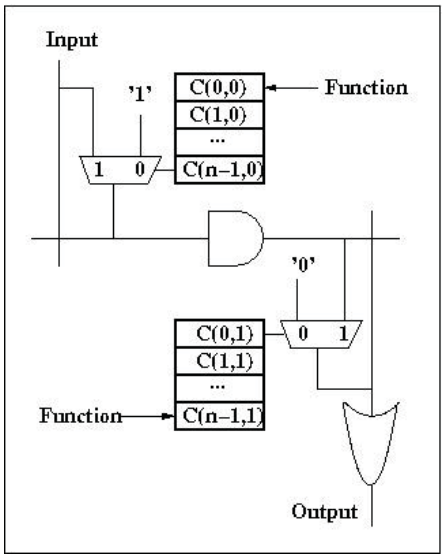

Figure 2. eTMPLA structure

context change cannot be realized in single cycle. A small logic context controller was designed to perform sequential context change and provide read/write arbitration for block RAM memory.

Because the context memory organization is basically changed from a wide memory with size of <number_of_contexts>x<number of eNodes> to a fixed <depth>x<width> size of the block RAM the dynamic reprogrammable node described in figure 1 need to be modified to include a memory element (flip-flop). Figure 3 presents the modified structure for eNode with possibility to store one configuration bit that can be loaded from external memory.

The configuring change for the modified structure cannot be done in single cycle as in the structure that uses small distributed memory blocks but at least <number_of_eNodes>/<block RAM port width> cycles are needed to transfer a new configuration in all eNodes. Thus the number of clocks needed to reconfigure the array with a different context is smaller when wide memory models are created using the standard available block RAM.

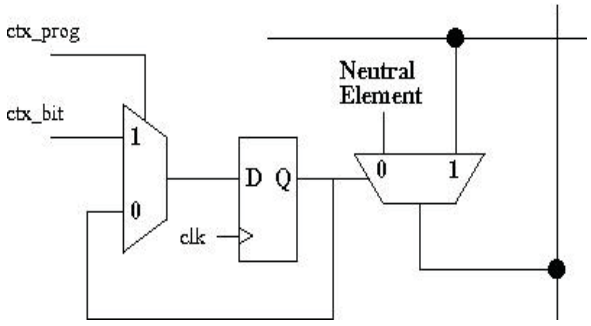

Figure 3. Modified eNode for area optimized eTMPAL

3. IMPLEMENTATION

3.1 Structured ASIC

Structured ASIC is a technology between standard cell and FPGA. Cell based layout has been dominating the high performance ASIC for a long time. Despite great progress, field programmable gate arrays (FPGA) cannot compete with cell based ASIC in terms of performance, area and power consumption. With rapid technology advances, manufacturing and design process of cell based designs have become extremely complicated and the cost of cell based designs has increased significantly in recent years. The big performance and cost gap between standard cell ASICs and FPGAs have inspired designers to search for new design alternatives. Structured ASIC can potentially fill this gap; they have most of the parts prefabricated so designers need to customize only few masks to complete de design. Design produced in structured ASIC have shorter development and manufacturing cycles as well as lower cost than cell based ASIC and they also have higher performance, gate density and lower power consumption then FPGA. In general cell based ASIC are used for high volume production, FPGA used for prototyping while structured ASIC is used for mid volume production. We have selected a structured ASIC technology based on lookup-table logic and single via used for programming and interconnect.

For implementation and validation of the area optimized eTMPLA structure we selected the Nextreme 90 nm product from eASIC structured ASIC vendor [7]. Their solution is to use predefined, pre-characterized logic structures containing LUTs and NAND logic gates with predefined routes that can be programmed through upper metal via structures, in this way improving the overall speed and power, but loosing the re-programmability advantage of the FPGA. The advantage of ”via programmed” circuits is that the circuit can be manufactured up to the last metallization layer and based on the design features customize this via layer to implement the desired operations. The base layer fabric offers intrinsic test resources (parallel scan chains, AC/DC scan circuitry, etc) that are guaranteed to work at a specified maximum frequency.

The logic cell in Nextreme called eCell perform functions based on two Look-up tables with 3 inputs, two NAND gates, one multiplexer, one flip-flop and three buffers as presented in figure 4. The logic cell can be configured to implement a 16 bit memory; multiple cells can be grouped to generate distributed memory, called eRAM, with size up to 256x16. However the possibility to convert logic cells to eRAM is limited by the number of address decoders

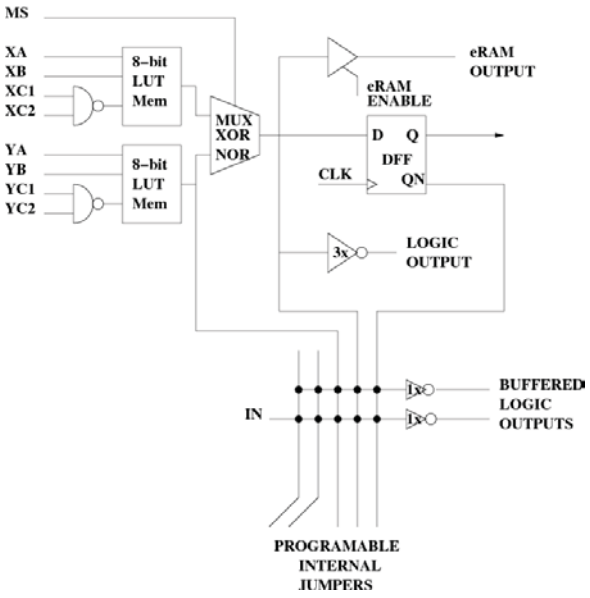

Figure 4. eCell internal structure

with only one address decoder per block of 16x16 eCells.

The structures ASIC arrays also include 32 kbit single port block RAM memory, called bRAM, which can be via configured as: 32kx1, 16kx2, 8kx4, 4kx8, 2kx16 and 1kx32. The block RAM configuration is selected using via and cannot be changed later on.

3.1 PLA architecture

A programmable logic array is a programmable device used to implement combinational logic circuits. The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes, which can then be conditionally complemented to produce an output. This layout allows for a large number of logic functions to be synthesized in the sum of products (and sometimes product of sums) canonical forms.

In this paper the PLA implemented structure is the classical PAL22V10 presented in [8]. We enrich the structure by adding programmability based on SRAM memory cells with timing multiplexing option obtaining a dynamic reprogrammable structure. PAL22V10 has a variable number of product terms per output, from 10 outputs 2 have 8 product terms (pins 14 and 23), 2 have 10 product terms (pins 15 and 22), 2 have 12 product terms (pins 16 and 21), 2 have 14 product terms (pins 16 and 20) and 2 have 16 product terms (pins 18 and 19). The number of inputs is constant for all outputs and equal with 44 because each AND44 needs to receive both inverted and non inverted input.

By using memory elements to store 16 active configurations we can view the implemented structure as 16 distinct PLAs that are time multiplexed into a single device.

3.3 Estimated area improvement

Next we are comparing the area for the eTMPLA memory distributed version (v1) with area for eTMPLA with context stored in block RAM (v2).

Original dynamic reconfigurable node (eNode) is a simple 2:1 multiplexer that can be implemented in single LUT. The version for block RAM based configuration memory includes two multiplexer and flip-flop so it requires full eCell:

$$\begin{aligned} A_{eNode\_v1} &= 0.5[eCell] \\ A_{eNode\_v2} &= 1[eCell] \end{aligned} \tag{1}$$

For each eNode we have a configuration memory of 16x1 to implement 16 possible contexts which can be implemented in one eCell for distributed configuration memory and 16 locations of one of 32 kbit block RAMs available on structured ASIC platform. In this case we will need to consider also the limitation of having a single distributed memory block decoder for an array of 16x16 eCells called eUnit.

$$\begin{aligned} A_{RAM16x1\_v1} &= 1[eCell] \\ A_{RAM16x1\_v2} &= 1/2048[bRAM] \end{aligned} \tag{2}$$

An 44 input AND function, AND44, corresponding to one product term can be implemented using two AND8 levels. One AND8 can be implemented with 2 LUTs and one multiplexer so it requires a single eCell:

$$A_{AND44} = A_{L1} + A_{L2} = \frac{\lceil N_{inputs} \rceil}{8} + \frac{\lceil N_{inputs} \rceil}{64} = 7[eCell] \tag{3}$$

For PAL22V10 the biggest number of product terms is 16, so in worst case we will need to implement 16 input OR function, OR16. Again we will use two levels of OR8 where OR8 can be implemented into single eCell:

$$A_{OR16} = A_{L3} + A_{L4} = \frac{\lceil N_{pterms} \rceil}{8} + \frac{\lceil N_{pterms} \rceil}{64} = 3[eCell] \tag{4}$$

The output macro cell (OLMC) need to implement a flip-flop and two multiplexers (4:1 and 2:1) so implementation requires 2 eCells:

$$A_{OLMC} = 2[eCell] \tag{5}$$

Based on equations (1)-(5) we can compute the area for both solutions in (6):

$$
\begin{aligned}
&A_{eTMPLA\_v1} = N_{eNode} * (A_{eNode} + A_{eRAM16x1}) + \\
&N_{pterm} * A_{AND44} + N_{output} * (A_{OR16} + A_{OLMC}) = \\
&5808 * 1.5 + 132 * 7 + 10 * 5 = 9686[\text{eCell}] \\
&A_{eTMPLA\_v1} = N_{eNode} * A_{eNode} + A_{bRAM} + \\
&N_{pterm} * A_{AND44} + N_{output} * (A_{OR16} + A_{OLMC}) = \\
&6782[\text{eCell}] + 3[\text{bRAM}]
\end{aligned} \tag{6}
$$

From equation (6) we estimate that eCell area for the block-RAM solution will be reduced with at least 30% but it will require usage of block RAM. Number of block RAMs needed can be easily computed dividing configuration memory size to bRAM size. bRAM usage is not a problem in eASIC as they are single port and nowadays most of design use more dual port memories than single port.

3.4 Results

The implementation was performed on structured ASIC using eASIC Nextreme NX750 device with BG480 package. The chosen synthesis strategy was the top-bottom one using the Magma Design Automation environment for structured ASIC 90 nm Fujitsu process. Table 1 presents area results for the implementation of the two versions of eTMPLA:

Table 1. Comparative area results

Design name	eCells [no.]	eCell [%]	bRAM [no.]	bRAM [%]
eTMPLA_v1	15027	27.17	0	0
eTMPLA_v2	10320	18.66	3	11.11

During placement we focused timing optimization only on the AND-OR structure while the context configuration timing paths were ignored. For eTMPLA_v1 we ignored eRAM placement constraints to be able to fit structure in the selected device.
The final placed and routed netlist was analyzed with a static time analysis tool - Synopsys PrimeTime. The final analysis included parasitic information about the design, silicon die and package. Table 2 present timing results showing improvement of the critical paths through PAL structure with increased time for configuration change.

Table 2. Comparative timing results

Design name	PLA timing [ns]	Context timing [ns]	Context change [ns]
eTMPLA_v1	3.010	6.533	6.533
eTMPLA_v2	2.184	4.448	809.536

The time for context configuration change is dependent on the selected block RAM width. In our case we used 32 bit width and number of cycles required to update a context is 5808/32, so 182 clock cycles are needed. This can be further reduced by using a wider memory 1kx128 that will reduce the reconfiguration latency to 46 clock cycles.
Testability of the implemented eTMPLA structure was demonstrated by running ATPG with Synopsys Tetramax, obtaining more than 99% test coverage for both implementations.

4. CONCLUSIONS

This paper describes a new area optimized architecture that allows dynamic reconfiguration for structured ASIC with high frequency performance, better than existing PAL/PLA commercial structures. The proposed structure can be embedded in designs implemented on structured ASIC allowing easy, in system debugging and reprogramming.
Dedicated on-chip area used to hold multiple configurations allows logic resource to implement different functionality in time. Consequently, the proposed architecture can exploit both the temporal and the spatial aspects of capacity to provide increased functional capacity. Area occupied by configuration memory is further optimized by using block memory for big reprogrammable arrays with a 30% area reduction.
The proposed solution can be used for high speed reconfigurable computing or in any design that require fast in-circuit debugging and dynamic reconfiguration.
Fully exploiting the time-space capacity of the multicontext devices introduces new tradeoffs and raises new challenges for design and CAD that can be covered in further studies.

References

[1] A. DeHon, *DPGA Utilization and Application*, Proc ACM Int'l Symp. FPGAs, pp. 115-121, Monterey, 1996.

[2] A DeHon, *Reconfigurable architectures for General-Purpose Computing,* PhD thesis, Massachusetts Inst. Of Technology, 1996

[3] S. Trimberger, D. Carberry, A. Johnson, J. Wong, *A Time Multiplexed FPGA*, Proc IEEE Symp. FPGAs for Custom Computing Machines, pp. 34–40, 1997.

[4] N. Denes, *eRECOP - Coprocesor Reconfigurabil In Tehnologia eASIC,* unpublished PhD paper, 2005

[5] D. Chang, M, Marek-Sadowska, *Partitioning Sequestial Circuits on Dynamically*

Reconfigurable FPGAs, IEEE Trans Computers, vol. 48, no. 6 pp. 565-579. 1999

[6] D.Buell, T. El-Ghazawi, K. Gaj, V. Kindratenko, *High Performance Reconfigurable Computing*, IEEE Computer Society, vol. 40, no. 3, pp. 23-27, March 2007

[7] eASIC, *Nextreme Zero Mask-Change ASIC Device Handbook,* www.easic.com, version 1.0, 2009

[8] Lattice & Vantis, *GAL22V10 – High Performance E2CMOS PLD Generic Array Logic,* http://www.latticesemi.com, 1998