Journal of Advances in Science and Technology
Vol. III, No. V, May-2012, ISSN 2230-9659

The Triptych FPGA Architecture

Bharti Gupta

Research Scholar, CMJ, University, Shillong, India

---♦-------------------------------------

BACKGROUND

Field-programmable gate arrays (FPGAs) have quickly become an important medium for the implementation of digital logic. These arrays exploit the increasing capacity of integrated circuits to provide designers with reconfigurable logic that can be programmed on an application-specific basis. This drastically increases flexibility in both the design process and the final artifact by permitting one board-level design to perform many functions, or to be upgraded in the field.

Almost all of the FPGAs currently available - and certainly all of the dominant ones - are based 011 a strict separation between logic and routing resources which pervades from the architecture itself to the tools employed in mapping designs. This closely parallels the development of integrated circuit gate arrays, arrays that utilize a few metal layers for customization. A similar distinction was made between logic cells and the routing resources that interconnect them. The strict separation was too confining, eventually leading to the sea-of-gates approach. Conceptually, the difference is that in a sea-of-gates the split between logic and routing area can be made 011 a per-mapping basis. This permits applications with regular logic structures to more efficiently utilize silicon area, while still permitting the use of many wires (at the expense of logic) for random logic circuits.

FPGAs are at a similar point today. As with mask-programmable gate arrays, an ever larger proportion of the silicon area is being devoted to routing resources to ensure that more and more designs are routable. Furthermore, the logic cells are becoming ever more complex, attempting to perform coarser-grain functions and lighten the load 011 the routing resources, but often end up being under-utilized. As was the case for gate arrays, it is now time to evaluate the logic/routing tradeoff in FPGA architectures.

We have developed the Triptych FPGA architecture, which can be viewed as an FPGA in the sea-of-gates style. Triptych addresses the two fundamental efficiency problems with current FPGAs: increasing routing area and decreasing cell utilization. The innovations include the flexible allocation of logic cells to either logic or routing functions, an array structure that more closely matches the wide shallow structure of most logic functions, and fine-grain cells that can be connected to form larger structures through short, fast local wires.

The rest of the section is divided into four major sections. Section provides the details of the Triptych architecture[1] and explains the rationale underlying the design decisions. Section describes some variations 011 the base architecture, including the first FPGA to fully support the implementation of asynchronous systems. Section completes the body of the section by presenting a methodology for comparing FPGA architectures, and demonstrates Triptych's advantages. Finally, section finishes with some conclusions.

1.1 The Triptych Architecture

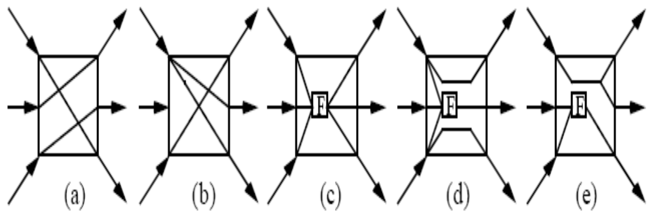
The overall goal motivating the development of the architectures described in this section was to reduce the significant cost paid for routing in standard FPGAs. The approach taken is twofold. First, instead of having a strong separation between logic and routing resources, with the percentage of each fixed in the architecture, the resources are combined in a way that allows the tradeoff of logic and routing resource on a per-mapping basis. This is done by replacing the logic blocks of standard architectures with Routing & Logic Blocks (RLBs), which perform both logic and routing tasks. The second modification is to match the structure of the logic array to that of the target circuits, rather than providing an array of logic cells embedded in a general routing structure. Most circuits are wide and shallow, with large fan-in/fanout trees. By matching the physical structure to this logical structure, we reduce the amount of “random” routing that is otherwise required.

The Triptych routing structure is shown in figure. Short, fast connections are provided between cells in a checkerboard pattern, with signals flowing from left to right. This basic structure is augmented with segmented routing channels between the columns that facilitate larger fanout structures than is possible in the basic array.

Finally, two copies of the array, flowing in opposite directions, are overlaid. Connections between the planes exist at the crossover points of the short diagonal wires. It is clear that this array does not allow arbitrary point-to-point routing like that associated with the Xilinx FPGA. However, we claim that this array matches the form of a large class of circuits, and that a mapping strategy that takes this structure into account can produce routable implementations1. Such an approach will use the fast diagonal connections for critical paths, while less critical signals can be routed longer distances via segments for vertical movement, and unused RLB inputs for horizontal movement.

A Triptych RLB is capable of performing both function calculation and routing tasks simultaneously, which leads to several different uses of the RLB. The three most obvious are: (a) a routing block with each input connected to one of the outputs; (b) a splitter with one of the inputs going to two or three of the outputs; and (c) as a function calculator with the three inputs going to the function block and the function going out the outputs.

However, there are two important classes of hybrids that help produce more compact designs. The first comes from the observation that in blocks used to calculate a three-input function, the function block value will most likely not go out all three outputs, and one or two of the input signals could be sent out the unused output connection(s), as in (d). Secondly, a function of two inputs can be implemented by making the function insensitive to the third input, thus allowing the unused input to be used to route an arbitrary signal, as in (e). An important observation is that the RLBs will never need to be used for one-input functions (i.e., an inverter), since any output signal will only be used either as an input to another arbitrary function block, where the inverter could be merged into the function computed, or to an output pin, where an optional inversion can be applied.



As was shown earlier, the Triptych FPGA has no global routing for moving signals horizontally. Instead, there is a heavy reliance on unused RLBs and unused portions of RLBs to perform these routing tasks.

The structure detailed above differs little from Triptych's initial conception. With the experience we have gained with the original Triptych architecture, with placement and routing tools, and with asynchronous and interfaced synchronous circuits, we have extended this architecture. These extensions can be grouped into two classes, 4-input 4-output RLBs, and asynchronous support, with the architecture for asynchronous circuits named "Montage".

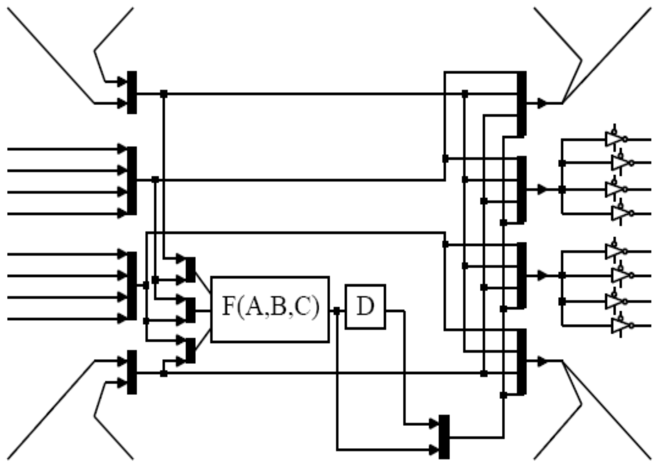


Figure. A 4-input, 4-output RLB. The segmented channel input is split into two separate inputs, to allow for greater routeability. Muxes are also added to the function block inputs to choose three of the four RLB inputs for function calculation.

1.2. The Interdependence of Architecture and Tools

It is important when developing a new FPGA architecture to ensure that the mapping tools will be able to take advantage of it. There is a strong analogy between processor architecture and compilers. Architectural features that tools cannot handle are not useful. Thus, it is impossible to evaluate an architecture or a set of tools in isolation. They are sufficiently interdependent that they must be developed and evaluated together. An unfortunate result is that some architectural features that may be valuable in their own right may be discarded because current tools cannot support them sufficiently. With increasingly sophisticated tools, previously discounted architectural ideas may become viable.

We structured the Triptych tools to support architecture development. Both the placement and routing programs were optimized for flexibility and not performance in terms of CPU time. It was more important to be able to retarget the tools quickly to evaluate variants 011 the Triptych architecture than to have the fastest turn-around time for individual tool runs. We recognized a tension between generality, which allows flexibility, and specificity, which allows the tools to take advantage of specific architectural features. Thus the first requirement of the placement and routing tools was that they be specific enough to take advantage of the primary features of Triptych. Second, we required enough generality to allow changes in the design of the RLB, the local interconnect and the vertical bus structure.

Flexibility was incorporated into the placement program by isolating the architecture-specific features to the cost function. While we could have introduced an architectural analysis phase that precompiled a cost function based 011 an architectural description, we instead opted for parameterizing certain aspects of the cost function while requiring other components such as local routability to be rewritten for the new architecture.

The routing resources of Triptych were described using the schematic capture system WireC [McMurchie94]. The description includes all specifics about the construction of the RLBs. the segmentation of vertical buses and diagonal connections. The output of the WireC system is a directed graph over all routing resources that includes delay information. Retargeting the router to a new architecture is a straightforward matter of modifying an existing template or creating a new one; no code modifications to the router are required. Indeed, we have retargeted the router to the Xilinx 3000 architecture and achieved very encouraging results.

1.3 USING TRIPTYCH

In this section, we present several circuits that we have mapped by hand to Triptych. The purpose of these examples is to demonstrate the constraints on routing and how multilevel logic circuits do indeed map to the physical structure provided by Triptych. In these examples, each RLB is shown as a cell with three input entries and three output entries. Each entry indicates an incoming or outgoing signal. Note that each block may create a new signal by computing a logic function over the inputs. Diagonals and reverse diagonals that are used in the implementation are highlighted, as are connections to the channel wires. For clarity, only those vertical wires carrying signals are shown.

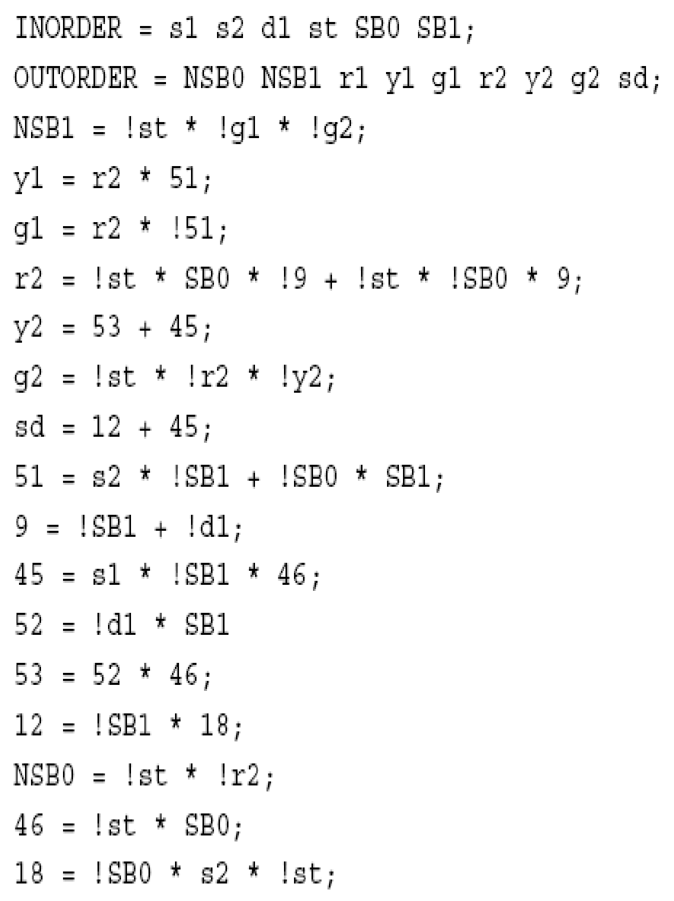
The power of using columns of RLBs for routing only is shown in this example which rotates a set of 8 bits 4 positions. Each level can be used to send one signal from each RLB to a neighbor of the final position. Since each RLB has two outputs, one intermediate RLB column and two vertical channels are required to route the signals to their final destination. This generalizes to the case where three signals are routed per RLB, which requires two intermediate RLB columns and three channels. Generalizing this use of the vertical channels suggests a naive place and route algorithm that alternates columns of RLBs used for routing with columns used to compute logic functions.

Subject to a sufficient number of routing tracks, this leads to a viable routing of arbitrary logic functions. However, as the next example shows, this scheme is much less area-efficient than is generally achievable.

State machine example - Figure shows the factored logic equations and corresponding Triptych implementation for the ubiquitous traffic light example. This example shows that circuit mappings can be very

compact if the individual logic blocks are correctly placed. The inputs and outputs of this circuit are all connected at the left and right of the array, except for three signals that use the pin input track of the vertical channels (shown dangling off the bottom). In this example RLBs are used to compute logic functions, 2 RLBs are used only for routing, and 6 RLBs are left unused (these 6 RLBs must be counted in order to achieve a rectangular mapping; they might be used in neighboring circuits). Also, this circuit is assumed to be placed along the left edge of the chip, so the vertical tracks at that edge are used to connect RLBs in the same column.

Note that this example would have been easier to map if the vertical wires could be used to route within a column anywhere in the chip, not just at the borders, and in fact such an extension is under consideration. This is about as compact a Triptych layout as can be expected for a random logic function.



```
INORDER = s1 s2 d1 st SB0 SB1;
OUTORDER = NSB0 NSB1 r1 y1 g1 r2 y2 g2 sd;
NSB1 = !st * !g1 * !g2;
y1 = r2 * 51;
g1 = r2 * !51;
r2 = !st * SB0 * !9 + !st * !SB0 * 9;
y2 = 53 + 45;
g2 = !st * !r2 * !y2;
sd = 12 + 45;
51 = s2 * !SB1 + !SB0 * SB1;
9 = !SB1 + !d1;
45 = s1 * !SB1 * 46;
52 = !d1 * SB1
53 = 52 * 46;
12 = !SB1 * 18;
NSB0 = !st * !r2;
46 = !st * SB0;
18 = !SB0 * s2 * !st;
```

Lyon bit-serial multiplier - Although our experience shows that Triptych can be used to implement a wide range of circuits, its locally connected structure makes it especially good for repetitive arrays like bitserial arithmetic circuits. The Triptych structure has some of the same features (e.g., nearest neighbor connections) as the Labyrinth FPGA which was targeted to bit-serial and pipelined/systolic circuits. We have chosen the Lyon bit-serial multiplier cell (Lyon 1981) as a representative circuit from this class. A full n-bit multiplier comprises n copies of this cell, and signal processing circuits typically make use of several of these multipliers, containing many individual cells.

This multiplier cell presents the same classic layout problem as that faced by VLSI cell designers. The cells need to tile horizontally so that inputs match outputs and vertically so that little space is wasted between adjacent multiplier cells. In this case, however, there is an extra dimension since a string of multiplier cells will wrap around the chip on the opposite direction of RLBs. Since there is one RLB that is used from the opposite direction, the layout must provide a “hole” into which this RLB can fit.

Note that these two logical RLBs can share a single physical RLB since they use independent paths through the RLB. The cost of this multiplier cell design is 12.5 RLBs which is not much more than the smallest conceivable design, which costs 11 RLBs. The 0.5 RLB results from the sharing of one RLB between two vertically adjacent multiplier cells.

Measurement and comparison - Although our experience with mapping circuits to Triptych is thus far very limited since automated placement and routing are still being developed, we have some preliminary measurements of the cost of Triptych implementations relative to Labyrinth and Xilinx. Since the area cost is measured for each FPGA type in terms of the number of logic blocks used for that technology, we must first normalize the cost of the different FPGA logic blocks to be able to compare the different FPGAs. Although such relative figures are difficult to come by, we have combined a relative size estimate based on die size and number of logic blocks, along with the relative number of program bits to arrive at the following relative cost figures. Using the cost of the Labyrinth logic block as the normalized unit cost, we estimate that the cost of a Triptych RLB is about 4-5 (4.5) units and that of a Xilinx CLB (configurable logic block) is about 20-25 (22) units. This places the Triptych logic cell squarely in the middle between the very cheap Labyrinth cell and the relatively expensive Xilinx cell.

Table gives the approximate cost of implementing a number of circuits using all three FPGAs, both in terms of each technology’s logic blocks and in normalized cost as defined above. We believe these figures indicate that Triptych is a promising architecture for a range of different circuits. These results are of course very preliminary and many more experiments must be done with other circuits and using automatic place and route tools.

Circuit	Labyrinth # blocks	normalized cost	Xilinx # blocks	normalized cost	Triptych # blocks	normalized cost
Multiplier	54	54	5	110	12.5	56
Traffic	280	280	6	132	24	108
s208	N/A	N/A	26	572	61	275

Table. Results of mapping three examples: the Lyon bit-serial multiplier; a traffic light controller; and ISCAS benchmark s208 the Labyrinth, Xilinx and Triptych.

Issues in mapping to Triptych - We have successfully mapped a number of regular structures and small control circuits to the Triptych architecture, and we are currently working on CAD tools that will automatically perform the mapping for arbitrary circuits. As with other FPGAs, the process of mapping a circuit onto Triptych can be considered to consist of three steps:

• covering: forming a circuit graph containing function nodes with at most three inputs,

• placement: assigning these function nodes to cell locations on Triptych, and

• routing: making the connections in the graph through the available routing on Triptych.

If the circuit to be mapped has a regular structure, as is encountered in domain-specific applications such as digital signal processing, an initial pattern for the repeating portion may be derived by hand. Circuits without regular structure, or “random logic”, must rely on heuristicbased automatic placement and routing methods similar to those used by other FPGAs.

However, because Triptych’s routing resources are highly constrained, placement and routing must be more closely integrated than they are in other FPGAs.

For the covering portion of mapping to Triptych, we assume that a tool such as chortle or mis-pga is available to express the original circuit as a graph of elementary gates and then cover the graph’s fanout-free trees with collections of three-input RLBs (Francis 1991, Murgai 1990). It should be noted, however, that a covering which minimizes the total number of RLBs may not be optimal when placement and routing are taken into consideration. For example, if after placement two of the inputs to a three-input RLB naturally both occur at a single location distant from that RLB, it is usually advantageous to split the RLB into two twoinput functions. If this is possible, we can route one less signal across the large distance.

Clearly, such situations are not unique to Triptych. However, we particularly wish to avoid routing extra signals horizontally whenever it can be avoided. Otherwise, RLBs become congested with signals they do not use. Such optimizations are difficult to predict at cover time and thus need to be attempted during routing.

Because Triptych’s routing resources are limited and fairly tightly constrained, we believe it is necessary to keep placement and routing well integrated. Evaluating possible placements with simple measures of routing length can lead to placements whose congestion make routing nearly impossible. Currently, we are exploring iterative improvement methods for placement which will assign an RLB only into locations which are adjacent to enough free tracks to route the RLB’s inputs and outputs. Thus, we avoid congestion at a local level whenever we place an RLB.

A complicating factor is that Triptych’s distance metric is non-symmetric. All pairs of RLBs that face in the same direction, except those in the same column, have a distance from the first’s output to the second’s input different than that of the second’s output to the first’s input.

Also, vertically adjacent blocks have the same routing distance as diagonally adjacent blocks. For these reasons, routing distance is not well represented by the x-y coordinates given to the RLBs. Work is ongoing to develop an integrated force-directed placement procedure, a Triptych-specific distance measure, and the congestion avoiding method mentioned above.

1.4 Results

We have performed experiments using PathFinder (specifically algorithm NCD) on two different FPGA architectures. We chose Triptych because the limited routing resources would expose the limitations of the algorithm, and Xilinx because this allowed comparison to an FPGA router currently in wide use.

For both architectures the routing resources were described using the schematic capture system WireC. The output of the WireC system is a directed graph over all the routing resources. All architectural information required by

PathFinder including delay information is contained in this directed graph. Retargeting PathFinder to a new architecture is a straightforward matter of modifying an existing template or creating a new one; no code modifications to the router are required. This approach provides a convenient mechanism for changing configurations of routing resources and examining the impact of these changes on the routability of circuits.

Experiments 011 Triptych - The Triptych architecture is an array of 3-input blocks ([Hauck92]). These blocks, known as RLBs (Routing and Logic Blocks) contain 3-input LUTs, as well as routing resources that can route inputs through blocks to neighboring blocks or onto buses. This approach is markedly different from other FPGA architectures, notably Xilinx, which place CLBs in a sea of routing resources. By comparison, Triptych has considerably fewer routing resources, many of which connect only nearest neighbors. The placement problem is obviously coupled closely to the routing problem. A placement program was constructed using] a simulated annealing approach, where the cost function is composed of both a routing distance metric and a metric that attempts to estimate routing congestion. Even with these measures of routability included in the placement cost function. PathFinder has the difficult problem of allocating the relatively limited routing resources to signals to achieve feasible source-sink routes. Factoring in the delay of critical paths obviously complicates the problem.

Bench	Reps	Logic Levels	Optimal Delay	Routed Delay	% over optimal
1	10	2	23.3	23.3	0.0%
2	4	6	57.3	57.3	0.0%
3	7	7	65.1	66.6	2.3%
4	2	14	123.3	125.3	1.6%
5	4	16	150.2	155.9	3.8%
6	8	16	134.2	138.0	2.8%
7	5	7	92.8	97.2	4.7%
9	5	10	61.6	62.6	1.6%

Table. Critical path delays for the PREP benchmarks mapped to Triptych.

The results of our experiments are shown in Tables 1 and 2. Covering with three-input functions was performed with the SIS mapper. All circuits were mapped to an 8x64 array of RLBs (512 total RLBs). Table shows the results of mapping the PREP benchmarks. The number of repetitions (Reps) of any particular benchmark is determined by th maximum that will fit in the Sx64 array when routed usin algorithm NC of Section. This insures a dense circu and is therefore a good test to determine how well the rout< can optimize for delay when algorithm NC'D of Section. is used.

The Logic Levels column in the table is the maximur number of 3-input functions between registers. Optima Delay is a lower bound to the delay that can be obtaine given a placement. This number is obtained during the firs iteration of the router when only the delay term in the co« function is present. Note that this number may not b achievable when congestion is resolved due to competitio between critical routes for the same routing resources. Th Routed Delay column is the delay of the critical path afte convergence. % over optimal is the % degradation of th Routed Delay from the Optimal Delay, which average 2.1%, and is at worst 4.7%.

Benchmark	Logic Levels	Optimal Delay	Routed Delay	% over optimal
ex1	8	76.1	79.7	4.7%
keyb	10	95.4	102.9	7.8%
C880	14	153.8	159.4	3.6%
clip	11	126.0	131.0	3.9%
C1908	15	161.9	171.3	5.8%
mm9b	14	128.2	128.3	0.0%
bw	7	89.6	98.1	9.4%
s832	11	117.0	118.1	1.0%
s820	10	112.8	114.7	1.7%
x1	7	84.1	94.7	12.6%
s953	10	101.7	103.8	2.1%
s1423	30	265.2	271.0	2.1%

Table. Critical path delays for selected circuits from ISCAS93 mapped to Triptych.

Table shows the results of running PathFinder on benchmarks obtained from ISCAS93. All circuits in the benchmark suite were included that utilized between 25% and 50% of the 8x64 array for logic (i.e. between 128 and 256 RLBs). In this case the delay degradation from optimal is an average of 4.6%$_?$ and is at worst 12.6%. The only other work quoting delay degradation from optimal is that of [Frankle92], in which an average degradation of 16% is found on the Xilinx 4000 architecture.

REFERENCES

[1] Ahrens, M., Gamal, A., “An FPGA Family Optimized for High Densities and Reduced Routing Delay,”Actel Corporation, IEEE Custom Integrated Circuits Conference, 1990.

[2] Atmel "Configurable Logic Design and Application Handbook" , 1995.

[3] Black, P, Meng. T., "A 140 mb/s 32 State, Radix 4 Viterbi Decoder," IEEE Journal of Solid State Circuits Vol. 27, No. 12, December 1992, pp. 1877-1885.

[4] Bowhill, W., et al, "A 433 MHz 64b Quad Issue RISC Microprocessor," IEEE Journal of Solid State Circuits, Vol 31, No. 11, November 1996, pp ##.

[5] Brebner, G. , "Configurable Array Logic Circuits for Computing Network Error Detection Codes," Journal of VLSI Signal Processing, 6, 1993, pp. 101-117.

[6] Chandrakasan, A. , "Low Power Digital CMOS Design," PhD. Thesis, U.C. Berkeley, August 1994.

[7] CLAy Family Introduction Datasheet, National Semiconductor, June 1994.

[8] DeHon, A. , "Reconfigurable Architectures for General Purpose Computing," M.I.T. PhD Thesis, A.I. Technical Report 1586, October 1996.

[9] Dobbelaere, I. , Horowitz, M. , Gamal, A. , "Regenerative Feedback Repeaters for Programmable Interconnections", ISSSC Digest of Technical Sections 1995, p.116- 117.

[10] Farrhi, A. , Sarrafzadeh, M. , "FPGA Technology Mapping for Power Minimization," International Workshop on Field-Programmable Logic and Applications, FPL '94. Proceedings, Springer-Verlag, 1994. p. 66-77.

[11] Gamal, A. , et al., "An Architecture for Electrically Configurable Gate Arrays," IEEE Journal of Solid-State Circuits, Vol. 24, No. 2, April 1989, pp 394-398.

[12] George, V. , The Effect of Logic Block Granularity on Interconnect power in a Reconfigurable Logic Array", CS 294 report, May 1997.

[13] Goto, G., et al "A 4.1ns Compact 54x54 Multiplier Utilizing Sign-Select Booth Encoders," IEEE Journal of Solid State Circuits, Vol 32, No. 11, November 1997, pp 1676-1683.

[14] Hauck, S., Borriello, G., Ebeling, C. , "Triptych: An FPGA Architecture with Integrated Logic and Routing", in Advanced Research in VLSI and Parallel Systems: Proceedings of the 1992 Brown/MIT Conference, pp. 26-43, March 1992.

[15] Infopad Project, U.C. Berkeley, http://infopad.EECS.Berkeley.EDU/infopad

[16] Izumikawa, M. , et al., "A 0.25um CMOS 0.9v 100-MHz DSP Core," IEEE Journal of Solid-State Circuits, Vol. 32, No. 1, January 1997, p. 52-60.

[17] Jou, S., et al, "A Pipelined Multiply-Accumulator using a High-Speed Low-Power Static and Dynamic Full Adder Design," IEEE Journal of Solid State Circuits, Vol 32, No. 11, November 1997, pp ##.

[18] Kaushik, R., Prasad, S., "FPGA Technology Mapping for Power Minimization," International Workshop on Field-Programmable Logic and Applications, FPL '94. Proceedings, Springer-Verlag, 1994. p. 57-65.