Tree Restructuring Approach to Mapping Problem in Cellular-Architecture FPGAs

N. Ramineni
Synopsys, Inc
Logic Modeling Group
19500 NW Gibbs Drive
Beaverton OR 97006, tel. (503) 531 2398

M. Chrzanowska-Jeske and N. Buddi
Department of Electrical Engineering
Portland State University
P.O. Box 751,
Portland OR 97207, tel. (503) 725-5415

ABSTRACT

A new technique for mapping combinational circuits to Fine-Grain Cellular-Architecture FPGAs is presented. The proposed tree restructuring algorithm preserves local connectivity and allows direct mapping of the tree to the cellular array, thus eliminating the traditional routing phase. The developed bus assignment algorithm efficiently utilizes medium and long distance routing resources (buses). The method is general and can be used for any Fine-Grain Cellular-Architecture type FPGA. To demonstrate our techniques, the ATMEL 6000 series FPGA was used as a target architecture. The results are very encouraging.

1. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) have become a popular design technology for designers seeking fast and cost effective implementation of their circuits. In recent years a lot of effort was spent on the development of the technology mapping and layout synthesis methods for two categories of these devices, namely Look-Up-Table based (LUT-based) and row-based FPGAs. A number of the architecture-specific technology mapping approaches were developed [6,7], but most of the placement and routing techniques were adopted from the semi-custom design styles like standard cells and gate arrays, with some modifications. The other types of the programmable architectures, Cellular-Architecture (CA) type FPGAs and the Complex Programmable Logic Devices (CPLD) have not drawn much attention of the CAD research community. In this paper, we focus on layout synthesis for one of these programmable architectures, namely CA-type FPGAs [2]. The main features of these devices, which distinguish them from the other types of FPGAs is the local connectivity between logic blocks placed as a symmetrical array. Logic blocks are usually of small granularity and of the standard-cell type with a limited number of inputs and outputs. Local or global buses are used for distance connections.

The "macro block" approach which is currently used in the industry to solve the layout problem for these devices is based on macro-generators. A technology independent representation of a circuit design is covered with a minimum number of relatively small standard subfunctions (macros) which usually have non-uniform shapes. Placement of macros is usually performed using a simulated annealing algorithm, which places the macros far from each other to assure that in the routing phase all connections between macros are completed. This technique leaves a lot of unused cells around the placed macros for possible use as routing blocks. Consequently, the number of cells which needs to be used for routing between macros is very large. In ATMEL 6000 series, on average, about 50% of the area occupied by a design is used for wiring connections or left unused [2]. This problem is mainly caused by not creating locally connected netlist during the synthesis steps. As the routing resources are very limited, efficient usage of these resources can significantly reduce the area occupied by the design, and thereby increase the capacity of the chip and improve circuit performance.

Recently, several logic synthesis approaches applicable to CA-type FPGAs have been presented [1,10]. In order to generate the circuit layout some of these methods require an additional layout synthesis step and some others do not. For example, if logic functions are represented as a two-dimensional array, they can be directly mapped onto a given CA-type architecture. The spectral methods based on orthogonal expansions [10] , and restricted factorization method [8] belong to the latter type. The approaches [1,5,10] based on trees and decision diagrams [9] which preserve local connectivity but still require the placement and routing steps, have also been reported. In most cases, however, when the tree is finally mapped to a rectangular area, the triangular structure of the tree may waste a large amount of area. Therefore, new comprehensive solutions to the optimized mapping of such trees to the regular, locally-connected arrays are of interest.

In this paper, we propose a new approach to the mapping of the binary trees onto the FPGAs with localized connections, like the CA-type FPGAs. It is based on the restructuring of the binary tree before final mapping is performed. A Squashed Binary Tree (SBT) [3] and a new Modified Squashed Binary Tree (MSBT) approaches are used to restructure a binary tree, before the mapping is performed. The mapping of the restructured tree is a simple process and routing step is eliminated. The method developed here is applicable to any binary tree. Our general approach is presented using ATMEL 6000 Series FPGAs as the target architecture, and it can be adopted easily to other CA-type FPGAs, such as Motorola, Algotronix or Pilkington.

The paper is organized as follows. In Section 2 we present a generic model of the CA-type FPGA and discuss a method of representing a Boolean function as a Permuted Reed-Muller (PRM) tree. The formal description of the problem is given in Section 3. In section 4, the various phases of the Tree Restructuring

Method (TRM) are discussed. Results and conclusions are given in sections 5 and 6, respectively.

2. CA-Type FPGA and PRMT

In general, the CA-Type FPGA is a regular array of locally connected programmable logic blocks. Each logic block is directly connected to a limited number of neighbors, usually four or eight, and to a small number of local and global (express) buses, usually four or eight, which are used for medium and long distance connections to link the logic array with the I/O blocks. Local and global buses run horizontally and vertically. Logic blocks, in addition, to logic and storage functions can also be used as wires. A generic CA-type FPGA is shown in Fig. 1.

The set of logic functions which can be implemented in one logic block is defined by the block architecture. All primitive logic functions like OR, NOR, NAND, AND, EXOR, 2-input mux, and a few combinations of the above gates can be realized by a logic block. All gates are two-input gates and the maximum number of outputs from a logic block is two.

A tree structure is very useful for mapping circuits to the CA-Type FPGAs as the connections between logic blocks, represented as the vertices of the tree, are local and each node in the tree has only connections to its parent and child nodes. In a binary tree, the maximum node degree is three, and therefore this configuration can be realized by using only adjacent logic blocks and local connections in the cellular array. The exponential growth of the number of nodes as a function of the tree level can, however, result in a very inefficient mapping. For most of the real functions, the shape of the tree is not getting as much wider at the bottom as could be expected. The tree is expanding from the root for a few levels, then the width of the tree tends to stay constant, and decreases towards the leaves of the tree. Therefore, by developing a good restructuring method that shape can be easily mapped to the cellular architecture without wasting many logic blocks for routing.

In general, any binary decision diagram or binary tree can be used as an input to our rrestructuring algorithm. However, to get better results a tree which can provide the best matching between a structure of the tree and functionality of logic blocks in a given FPGA should be selected. Any combination of Shannon, Davio I and Davio II expansions can be used to produce decision diagrams [1,9] which are binary trees with a decomposition variable associated with each node. By using only Davio I expansion, a completely specified Boolean function can be decomposed to produce a binary AND/EXOR tree structure called Reed-Muller Tree [1] with only positive decomposition variable associated with each node of the binary tree. This tree is called Reed-Muller tree because once it is flattened, it represents a Reed-Muller canonical form. If the order of variables used for decomposition is the same in all branches of the tree then it is called Non-Permuted Reed-Muller-Tree (RMT). However, if the order of the variables is not the same in all the branches the tree is usually smaller. Such, tree called Permuted-Reed-Muller-Tree (PRMT), is generated by the program REMIT [1], and is used as the input to our algorithm. PRM Tree structure is very well suited for ATMEL FPGAs as it matches the AND-EXOR configuration of the ATMEL logic block. Fig. 2 shows the PRMT of the function given below.

$$f = \bar{a}\,bcdef\,\bar{g} + \bar{a}\,efg + aef$$

The root of the tree represents the output of the function. The logic gates are represented with generic logic gate symbols and the primary (expansion) variables (a, b, c ...), which are associated with each node, are also shown.

3. MAPPING PROBLEM FORMULATION

PRMT which represents a given Boolean function is modeled as a binary tree $T = (V, E)$ which consists of the ordered set of nodes V, and the set of directed edges E defined as follows:
* V = {$v_j | v_j$ represents a PRM Tree node which can be realized in one logic block of a given architecture}
* E = {$e_j | e_j$ is an edge from v_j to v_{j+1}, and represents a functional relation between these two nodes}. Direction of the edge represents the direction of signal flow in the actual design. The nodes of the tree T are labeled with the primary signals (expansion variables) entering the logic blocks represented by the nodes. The fan-out of v_j is equal to 1 and the fan-in of each v_j is not greater than 2.

The physical resources of the CA-type FPGA are represented as the undirected graph $G_p(V_p,E_p)$ with the ordered set of vertices, V_p, and a set of edges E_p defined as follows:
*V_p ={v_p | v_p represents a logic cell of the CA-type FPGA}
*E_p ={e_P | e_p represents the programmable connections between the adjacent cells}
The vertices are numbered according to their positions in the column-row matrix of the CA-type chip.

Mapping Problem Formulation: Given the undirected physical graph G_p (V_p,E_p) and the network tree $T(V, E)$ representing a design, find a mapping of the tree T to the physical graph G_p that satisfies the routing constraints of the given architecture and that minimizes the size of the rectangular area covered by the design and the number of logic blocks used for routing.

4. Tree Restructuring Method (TRM)

The input to the logic optimization phase is in *pla* format. The output of the logic optimization phase is a binary tree. In this paper the logic optimization program REMIT is used to produces the PRMT representation of the given function. Next, an optional technology mapping step is introduced to perform technology specific optimization. The tree is then restructured using MSBT technique, and then mapped to the target architure. Finally, the primary inputs are assigned to the local buses in the bus assignment phase.

4.1. Technology Mapping

This phase is specific to the architeture of the target FPGA. For example in ATMEL 6000 [2], the combination of AND/EXOR gates can be realized in one logic block [4]. The size of PRMT can be reduced by grouping nodes corresponding to such gates. The grouping algorithm described in [4] is used and the grouped PRMT corresponding to the function in Fig. 2 is shown in Fig 3.

4.2. Squashed and Modified-Squashed Binary Tree

The Squashed Binary Tree [3] approach was chosen as it gives us a possibility to shape the tree into a rectangular form which closely resembles the CA-type architecture. The rectangular shape can be directly mapped to the array satisfying the design restrictions. Mapping a SBT to the CA-type architecture is just a straight-forward process as we place each node of the SBT in one column of the target array and then make

the neccessary connections. The details are explained in Sec 4.3.

The squashed binary tree is formed by projecting the binary tree onto its leaves. Starting from the root, nodes are projected onto their left-most descendants. Next, the tree is traversed in the bottom-up direction [5], and if a node has two children, the process is repeated starting with the child node which was not projected earlier. Figures 4(b) and 4(c) represent the Squashed Binary Tree (SBT) and the Modified Squashed Binary Tree (MSBT) of the binary tree shown in Fig. 4(a). Fig. 5 shows the squashed binary tree representation of the PRM tree from Fig. 3.

To obtain a more compact shape of the mapped design the original SBT algorithm was modified and the Modified SBT (MSBT) algorithm was implemented in our package. The modified squashed binary tree is formed by projecting nodes of the binary tree onto its leaves in the depth-first manner. The node can be projected into its left or right descendent depending on the situtation (i.e left most descendant constraint is relaxed). As a result, a tree with a smaller number of nodes is generated.

The modified squashed binary Tree $T_b(V_b,E_b)$ consists of the set of vertices V_b and the set of directed edges E_b.
*V_b = {v_b | v_b represents the set of nodes of the Tree T(V,E), projected into the same vertex of $T_b(V_b,E_b)$}
*E_B = {e_b| e_b represents the directed edge from v_{bi} to v_{bj} if any of the nodes of the Tree $T(V, E)$ which were collapsed to nodes v_{bi} was connected to any of the nodes of the Tree $T(V, E)$ which were collapsed to vertex v_{bj}}.

The vertices of the MSBT are labelled as $v_{b1},v_{b2}.....,v_{bn}$, where n is the number of leaf nodes in the original PRMT tree. Each vertex v_{bi} is a set of nodes ($v_1,v_2,.....v_m$) of the PRMT tree, which were collapsed to that vertex. An edge exists between v_{bi},v_{bj} if there exists an edge e_{kl} between v_k and v_l, where v_k is a node which was collapsed to vertex v_{bi}, and v_l is a node which was collapsed to vertex v_{bj}.

The MSBT of the PRMT from Fig. 4 is shown in Fig. 6.

4.3. MSBT Mapping

Each vertex of the MSBT is be implemented in one column of the CA-type array. The number of vertices of MSBT is equal to the number of columns of the CA-type array used for design implementation, and the maximum number of nodes of the grouped PRM tree which are projected into one vertex of the MSBT determines the number of rows required.

MAPPING PROCEDURE

For all vertices v_{bi} of the MSBT, place all nodes belonging to v_{bi} in the same column of the target cellular array. Vertex v_{bi} will occupy as many rows as there are nodes belonging to that vertex. First, place all nodes v_i belonging to v_{b1} in column *one*. Then place nodes belonging to v_{b2} in column *two* staring with Row(j) defined by the edge between v_{b1} and v_{b2}. If node, $v_k \in v_{b1}$ is conected to node $v_l \in v_{b2}$, then node v_l is placed in the same row(j) as node v_k. This mapping is continued until all vertices are placed. Add additional routing cells if required.

The $v_{b1},v_{b2}..v_{b6}$ are the vertices of the MSBT in Fig. 6. We map v_{b1} to column 1 of the array; v_{b2} to column 2 and so on. Since there is a directed edge $e_{4,9}$ from v_{b1} to v_{b2}, therefore we connect cell 9 placed in column 2 with cell 4 in column 1. Similarly, there is a directed edge $e_{0,2}$ between v_{b1} and v_{b3}, and we connect them by using the "routing cells" in Row 1 between cell '0' and cell '2'. The "routing cells" are indicated by 'R'. Fig. 7 shows the mapping of SBT from Fig. 5, and Fig. 8 shows the mapping of the MSBT from Fig. 6 onto the CA-type array.

Comparison of Fig. 7 and Fig. 8 reveals that the MSBT approach results in smaller number of columns and routing cells needed for the layout implementation than SBT. A significant advantage of MSBT approach, which can be observed from Fig. 6, is that we can easily predict the area of the rectangle enclosing the design. It can be assumed that in FPGAs all cells have approximately the same delay. Therefore, in MSBT approach the signal delays can be calculated before the actual mapping is performed. The predictability of the signal delays is a very important advantage of this approach. The area and delays can be optimized by properly choosing the order of the projected nodes.

4.4. Bus Assignment

Once MSBT is created it can be directly mapped to the CA-type array, and only the necessary logic blocks required for routing need to be added. The exact number and the exact locations of these additional routing blocks is defined by the MSBT and hence is known a priori. Then, to complete all the connections, the bus assignment has to be performed. The primary (decomposition) variables from the original PRMT have to be assigned to the local buses. We have developed efficient heuristic which assigns the variables to the local buses such that the number of local buses used by the same variable and the number of cells needed to distribute the same signal to different buses are minimized.

The steps of the bus assignment algorithms are as follows:

Step a : For each decomposition variable d_i; calculate M_i, the total number of nodes in PRMT to which the variable d_i is assigned. Form a list 'L' of variables for which bus assignment has to be done. Initially this list contains all the primary variables.

Step b : For all d_i calculate R_{ij}, the total number of nodes of the PRMT named with variable d_i and placed in row r_j of the cellular array. If $M_i = R_{ij}$; then assign variable d_i to the upper local bus UR_j of the row R_j. If another variable is already assigned to UR_j then assign d_i to the bottom local bus BR_j of the row R_j. Remove d_i from the list 'L'. If BR_j is occupied then pick the next variable in the list. If the list is empty, EXIT. Repeat the same procedure for columns.

Step c : For all variables d_i in the list, calculate N_i, where $N_i = M_i/2$ (if M_i even); $N_i = (M_i/2) + 1$ (if M_i odd). Substitute $M_i = N_i$.

Step d : If $M_i \geq R_{ij}$ for d_i, then assign d_i to UR_j. If UR_j is already assigned then assign the variable to BR_j. Decrement M_i by R_{ij}. If $M_i = 0$; Remove d_i from the list. If BR_j is occupied then pick the next variable in the list. If the list is empty then, EXIT. Repeat the same procedure for columns.

Step e: If the list is not empty, repeat *Step c* and *Step d* until all assignments are completed.

The completed mapping of the MSBT of our leading example to the ATMEL 6000 series FPGA is

shown in Fig. 9.

5. RESULTS

The, TRM was tested on a set of MCNC benchmarks. Since at present TRM can handle only single output functions, we modified MCNC benchmarks by extracting single output functions. We compared the area and the number of local buses used by Squashed Binary Tree (SBT) method versus Modified Squashed Binary Tree (MSBT) approach, and MSBT approach versus commercially available ATMEL (IDS) tools. We assumed ATMEL 6000 as the target architecture. The results are presented in Table I and Table II.

MCNC	PRMT	GROUPING			SBT			MSBT		
		L	C	GT	R	**T**	RT	R	**T**	RT
5x10	17	15	0	15	5	20	26	0	15	27
misex54	24	18	4	22	6	28	65	1	23	35
misex62	776	394	131	525	173	728	2240	173	728	2208
misex63	845	446	148	594	183	777	2736	183	777	2704
f18	423	272	14	286	313	599	1930	37	328	1140
f19	634	443	241	684	541	1225	4424	209	893	3072
sao21	551	337	105	442	186	628	2016	151	593	1792

Table I

Table I shows the results of the SBT and MSBT approaches for the modified set of MCNC benchmarks. The second column, "PRMT", shows the number of gates required to implement the function in PRMT form (tree generated by REMIT). In the GROUPING section of the Table I, column "L" shows the number of logic blocks used to implement the logic and column "C" shows the number of connecting cells added due to A-B restrictions of the ATMEL architecture [2]. Column "GT" shows the total number of cells. SBT and MSBT column sections present the results of SBT and MSBT approaches, respectively. R is the number of routing logic blocks added when constructing SBT or MSBT. T is the total number of logic blocks required to realize the function. RT represents the size (in terms a of number of cells) of the smallest rectangle enclosing the mapped circuit. The results clearly show that MSBT approach has significantly reduced the total number of logic blocks required to implement a given function and the size of the enclosing rectangle is also smaller.

The layouts of the leading example are shown to illustrate the differences between our final layout and the layout generated by ATMEL tools The ATMEL generated layout is shown in Fig. 10, and the layout generated by our TRM package in Fig. 11. It can be easily noticed that our layout is more compact and gives the better utilization of the chip resources, and therefore better perfromance. In Table II the comparison between our method Tree Restructuring Mapping (TRM) and the ATMEL commercial tools package (IDS) is presented using the MCNC benchmarks. B is the number of buses and L is the number of logic blocks used for implementing the logic, C is the number of cells used for routing, and A represents the rectangular area occupied by the core of the design (without I/O pins). We compared only core area of the mapped design because the ATMEL tools perform bus assignment in an inefficient way. The resulting area is very large and contains a lot of unused logic cells. As it can be seen from the Table II our methods give much more compact layouts than the ATMEL tools. The number of local buses and logic blocks used for routing is much smaller for all run examples. The L/R ratio, logic blocks to routing blocks, is high for our method, which it gives more "logic power" to the implementation of the designs and improves performance.

MCNC	ATMEL(IDS)				Our Program(TRM)			
	B	L	C	A	B	L	C	A
5X10	46	17	23	90	19	15	6	27
f5	28	11	12	45	10	8	6	12
misex54	58	24	27	180	19	18	13	35

Table II

To compare the perfromance improvement we have done timing analysis using ATMEL (IDS) package. The design, generated by our TRM package is entered using interactive editor of the ATMEL (IDS) tools, and then we run the timing analysis from the ATMEL (IDS) package on both implementaions. The longest path delays obtained are shown in Table III. The average improvement achieved with our approach is around 50%.

MCNC	ATMEL ns	TRM ns
5X10	71.14	27.90
f5	57.30	26.70
misex54	107.40	36.40

Table III

6. CONCLUSIONS

We proposed a new tree restructuring method for mapping combinatorial circuits onto CA-Type FPGAs. By preserving the local connectivity among the logic blocks the routing phase was completely eliminated. Mapping process is straight forward, and therefore enables predictability of the signal delays, which is very important advantage of this method.

Our TRM program is independent of the logic optimization steps as long as the function is represented as a binary tree. Our method is a general method and can be applied for a general class of CA-type FPGAs. The results on some MCNC benchmarks shows that our method is better both in area and delay when compared to commerically available tools. Currently, we are working towards extending the TRM for multi-output functions.

7. REFERENCES

[1] L. Wu, and M. A. Perkowski, "Minimization of Permuted Reed-Muller Trees for Cellular Logic Programmable Gate Arrays."*In H. Gruenbacher and R. Hartenstein (eds.), LNCS*, No. 705, Springer Verlag, pp.78-87, 1993.

[2] ATMEL Corporation CMOS Integrated Circuit Data Book, 1993, 1994. 2125 O'Neil Drive, San Jose, CA, 95131.

[3] F.T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays , Trees, Hypercubes, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1992.

[4] Z, Songhua, "Grouping of The Permuted Reed-Muller Tree", *Report, Portland State University, OR 97207.*, 1993.

[5] N. Ramineni, and M. Chrzanowska-Jeske, "A Routing-Driven Mapping For Cellular-Architecture FPGA's", *36th Midwest Symposium On Circuits & Systems*, pp.308-311 , Aug 1993.

[6] Jason Cong and Yuzheng Ding, "An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *Proc. IEEE ICCAD,* pp. 48- 53,

November 1992.

[7] R. J. Francis, J. Rose, and Z. Vranesic, "Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs," *Proc. 28th DAC, pp. 227-233,* June 1991.

[8] A. Sarabi, N. Song, M. Chrzanowska-Jeske, M. Perkowski, "A Comprehensive Approach to Logic Synthesis and Physical Design for Two-Dimensional Logic Arrays", *Proc. of 31st DAC*, pp.321-326, June 1994.

[9] U. Kebschull, E. Schubert, W. Rosenstiel, "Multilevel Logic Synthesis Based on Functional Decision Diagrams," *Proc. IEEE European Design Automation Conference,* pp. 43-47, 1992.

[10] I.Schaefer, M.A.Perkowski,H.Wu, "Multilevel Logic Synthesis for Cellular FPGAs Based on Orthogonal Expansions," *Proc. IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design,* Sept. 1993, Hamburg, Germany, pp. 42-51.

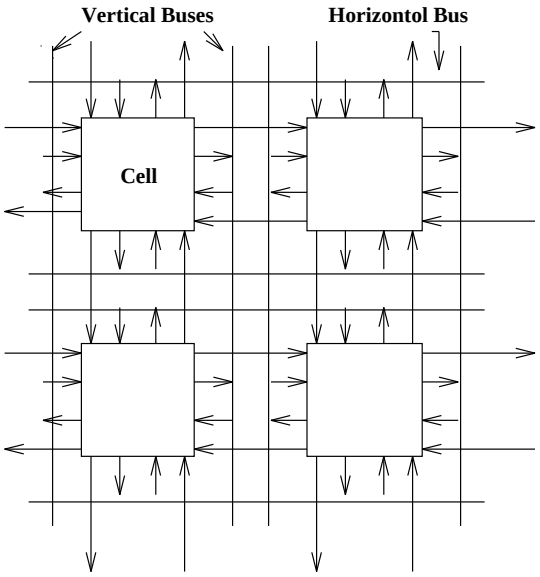

Fig. 1 A generic CA-type FPGA

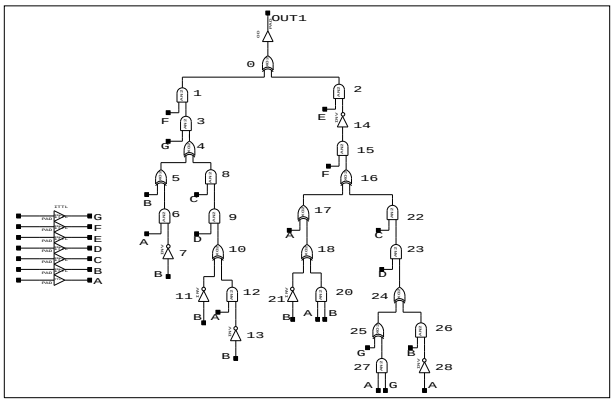

Fig. 2 Permuted Reed-Muller Tree

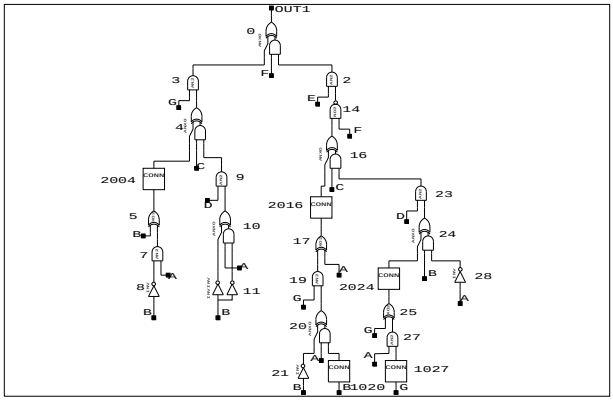

Fig. 3 Grouped PRMT

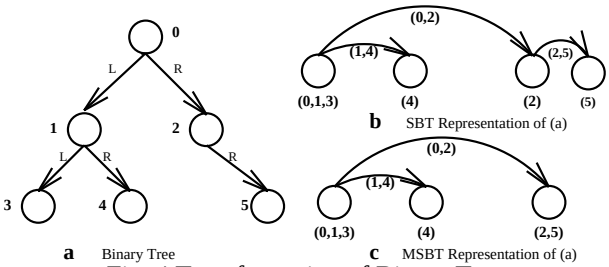

Fig. 4 Transformation of Binary Tree

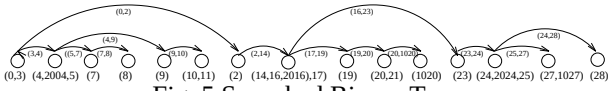

Fig. 5 Squashed Binary Tree

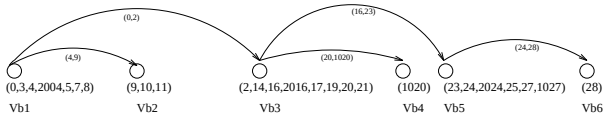

Fig. 6 Modified Squashed Binary Tree

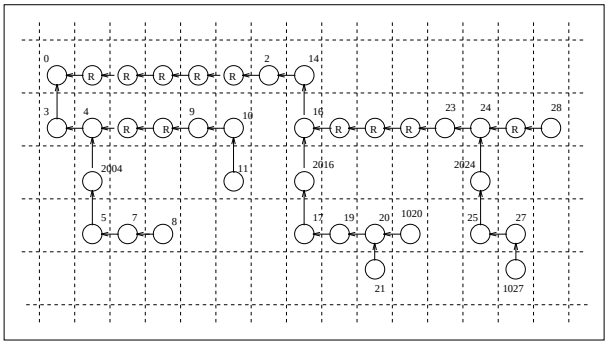

Fig. 7 Mapping SBT to CA-type Array.

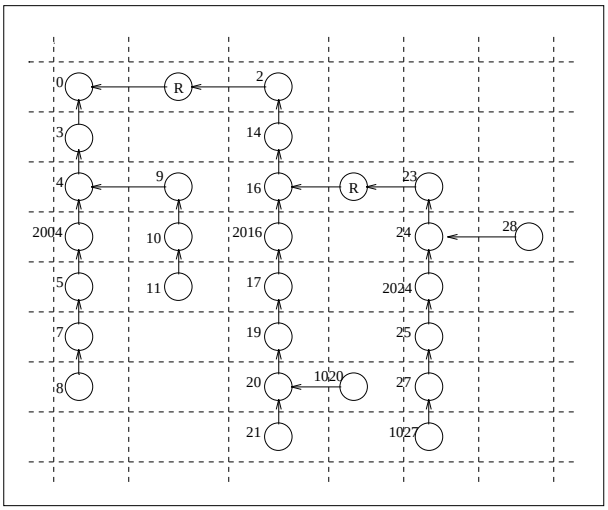

Fig. 8 Mapping MSBT to CA-type Array.

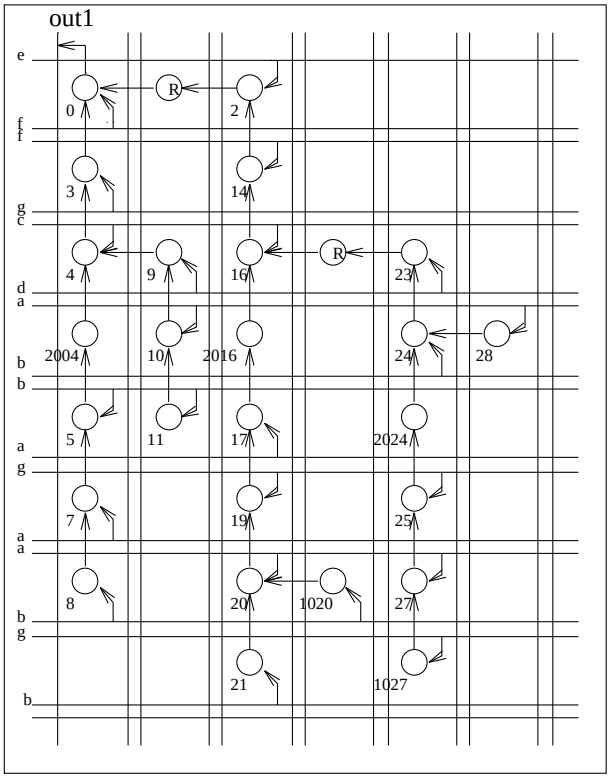

Fig. 9 Mapping of MSBT with Bus assignment

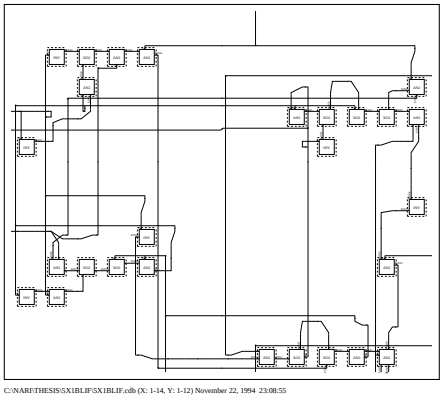

Fig. 10 Layout of the Eaxample with ATMEL Tools

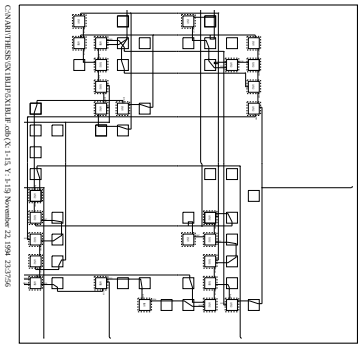

Fig. 11 Layout of the Example with our Program TRM