Synthesis on Multiplexer-based F.P.G.A. using Binary Decision Diagrams

T. Besson, H. Bouzouzou, M. Crastes, I. Floricica, G. Saucier

Institut National Polytechnique de Grenoble / CSI
46 Avenue Felix Viallet 38031 GRENOBLE Cedex FRANCE

Abstract

This paper presents synthesis methods for multiplexer based programmable gate arrays based on Binary Decision Diagrams (BDDs) for speed optimization, and Reduced Ordered Binary Decision Diagrams (ROBDDs) for area optimization. A direct mapping is performed on the 2 types of Binary Decision Diagrams and the results show a high benefit compared to library based approaches.

1. Introduction.

The increasing use of programmable devices is the most relevant event in the ASIC world for the present decade. Due to their high flexibility and fast turn around time they constitute an ideal support for fast prototyping as well as for actual designs as long as the performances they provide are acceptable. This paper focuses on the well known family of ACTEL FPGAs, whose novel structure is multiplexer based [5]. It is obvious that multiplexer based synthesis may benefit from an initial specification of combinational blocks in terms of Binary Decision Diagrams. Therefore, we investigate here the BDD basic representations and analyze their applications to this synthesis problem.

First Binary Decision Diagrams with no constraint on the input order are considered. The heuristic used for choosing at each node the relevant splitting variable aims to reach the leaves as fast as possible. Therefore, this representation will be used as initial description for speed optimization.

Second, an input order will be imposed on all the paths of the BDD in order to favour subtree sharing. The subtree sharing will lead to concise Reduced Ordered Binary Decision Diagrams in which shared subtrees will be represented only once. This is extended like in [8] to reduced ordered binary decision diagrams with negative edges, in which a subfunction or its complement may be shared. All the efforts have been put on selecting a good order. This is a difficult research topic [8, 9, 10] and the order proposed here looks quite convincing.

In a second step the technology mapping phase is invoked. It consists in covering the Boolean functions, represented by Binary Decision Diagrams, by small trees or patterns representing the programmable multiplexer devices and all its variations. By variation we mean the basic pattern in which some inputs have been forced to fixed values (0 or 1). Some similar approaches may be found in [6] where ROBDDs are used as well as If Then Else DAGs [12].

In a last step, practical results are given on an exhaustive list of benchmarks, and all the comparisons available to the authors, dealing with academic research, are reported.

2. Binary Decision Diagrams based representations of Boolean functions.

2.1. Binary Decision Diagrams.

2.1.1. Definitions.

Given a Boolean function f(...,x,y,z,...) the positive and negative cofactors of f with respect to x are defined by f_x = f(...,1,y,z,...) and $f_{\wedge x}$ = f(...,0,y,z,...). A BDD of a Boolean function is a direct acyclic graph based on successive Shannon decompositions according to splitting variables. Each nonterminal node N corresponds to a splitting variable x and is the origin of two arcs : low(N) and high(N). The graph rooted at low(N) represents the negative cofactor $f_{\wedge x}$, and the graph rooted at high(N) represents the positive cofactor f_x.

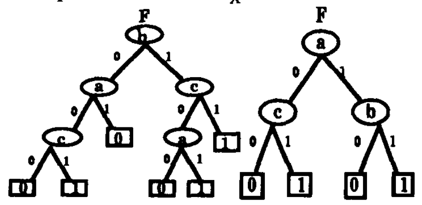


Fig. 1 BDDs for different splitting variable choices

The complexity in terms of node count and depth of such a representation strongly depends on the choice of the splitting variables. This is illustrated in figure 1 where F=ab+^ac is represented with two different splitting variable choices.

2.1.2. Optimized choice of the splitting variables for BDDs.

The following algorithm is used to select the splitting variable at each node:

For each Boolean function f expressed as a minimized sum of product terms, not equal to a constant or reduced to a single variable go through the following steps:

step 1: If the function is reduced to a single product term, then the size of the BDD will be the same for all orders but the size of the resulting ACTEL network may vary. We try to identify substrings ^x^yz as such a substring may be implemented by an ACT1 cell. Then the splitting variables will be selected according to this order. This step is similar to the step found in [6].

step 2: If a variable appears in all product terms under the same form, then select it; if several choices exist select a variable in complemented form.

step 3: If a product term is restricted to a single literal then select this literal; if several choices exist select a variable in complemented form.

step 4: If all the variables appear only once in a function then consider the smallest product term, select the variables of this product term as splitting variables according to the order defined in step 1. This will create identical subtrees for the variables fixing to 0 this product term. In order to avoid the duplication of this subtree, it becomes a subfunction.

step 5:(general rule): Consider the set of variables of maximum occurrence; within this set select one variable (x) minimizing C=|[nb of occurrences of x] - [nb of occurrences of ^x]| (f_x and $f_{\wedge x}$ will have nearly the same number of product terms).

2.2 Reduced Ordered BDDs.

2.2.1. Definition.

For an Ordered BDD (OBDD) a total ordering "<" of the set of variables is imposed. On any path from a root to a terminal node the splitting variables occur in the given order. The graph may have redundant vertices and duplicated subgraphs. They are eliminated by repeatedly applying transformations rules [1]. The maximally reduced graph is referred as Reduced Ordered BDD (ROBDD). This graph is unique for a given variable ordering and therefore constitutes a canonical representation of the function. This property has important consequences for functional equivalence testing [11]. Fig. 2 shows the ROBDD for the function F=ab+cd with the order a<b<c<d.

Let us mention again that the input ordering is a very strong optimization criterion for reducing the complexity of ROBDD representations. Figure 3 shows the influence of such an order.

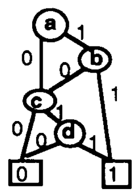


Fig. 2 Reduced Ordered BDD

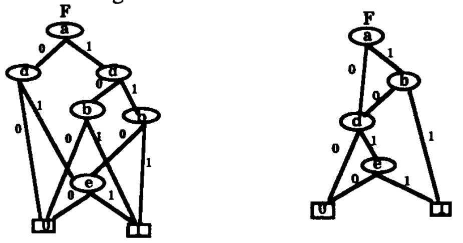


Fig. 3 Representation of f=ab+de by ROBDDs for the orders a<d<b<e and a<b<d<e

The area minimization approach proposed here makes use of a more reduced representation of ROBDD, including the negative edge capabilities [8]. In this representation, not only common subtrees corresponding to the classical notion of subfunctions (SF) are identified, but the complemented subtrees (^SF) are also detected. The subtree and its complement will be represented only once and a negative edge can be considered as an inverter. The node count or ROBDD size takes into account only once the subfunction nodes, ignoring the negative edge symbol. Figure 4 shows a ROBDD representation with negative edges, illustrating the sharing of subfunctions with the inverting capabilities.

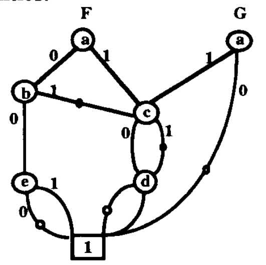


Fig. 4 ROBDD with negative edges for the functions F=^a^be+^abcd+^ab^c^d+a^cd+ac^d and G=a^cd+ac^d

2.2.2. Good input order for ROBDDs with negative edges.

As said above, finding a good input order is fundamental for handling efficiently ROBDDs. Original methods starting from Sum Of Product terms expressions and independent of any previous factoring or netlist representation are proposed here. Three methods were investigated and compared.

The first one is based on kernel extraction and kernel analysis. The kernels are considered in a greedy manner according to a cost function. For each kernel, a precedence relation states that the variables of the cokernel precede the variables of the kernel. A new kernel is chosen as long as there is no contradiction with the relations induced by previously chosen kernels. At the end, no more kernel can be accepted and an input order respecting the precedence relation is chosen.

For the second method, kernel extraction is restricted to kernels shared by different functions, thus favouring later subtree sharings.

A last method is a simplified one and looks for variable occurrences analysis. The variables are ordered according to their occurrences in the set of functions.

In a last step, ordering improvement is performed by variable exchanges, either within a window or within equivalence classes (set of variables not ordered in the previous step).

In table 1, the best solution is shown in terms of node count for some combinational benchmarks processed from their SOP expressions. In average, the best results in terms of number of nodes are those obtained by considering the global kernels. This is coherent with the initial idea that ROBDDs are convenient representations for subtree sharings and should lead to a good solution.

In this table, K means kernel analysis order, GK means global kernel order and O means occurence order. CPU is the CPU time in second on a SPARC station 4/65 with 24 Mo of memory.

Name		#Node	CPU	Name		#Node	CPU
5xp1	GK	42	3,3	frg1	K	105	390,5
9sym	O,K,GK	25	13	keyb	O	178	23,3
alu2	GK	159	18,6	kirkman	K	110	30,5
alu4	K	582	142,8	misex1	O,GK	36	1,6
apex1	K	1379	2238	misex2	GK	88	47,4
apex2	GK	581	1664,4	planet	GK	363	53,3
apex6	K	692	923,8	rd53	O,K,GK	17	1,2
apex7	K,GK	301	1586	rd73	O,K,GK	31	8
b9	K	70	15,3	rd84	O,K,GK	42	24
becount	O	27	1,6	sand	GK	319	91,8
bw	K,GK	95	5,7	sao2	O	81	13,8
clip	GK	75	22	sct	K	506	587
count	K	82	71,4	seq	O	1309	1945
dk512	GK	39	1,5	styr	O	290	50,2
duke2	O	377	189,9	vg2	K,GK	94	160
f51m	GK	39	4,2	z4ml	O	17	3,4

Table1: Number of nodes.

3.Decomposition on multiplexer-based cells.

3.1. Representing the multiplexer based cells by ROBDDs

The FPGAs considered are the ACT1 and ACT2 from ACTEL. The basic cell is a connection of 3 2-to-1 multiplexers. Figure 5 (resp. figure 6) shows the structure of the ACT1 (resp. ACT2) cell and the associated ROBDD (or BDD). In ACT2, the 2 control lines SOA and SOB are ANDed together.

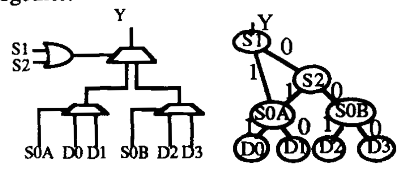


Fig. 5: ACT1 cell

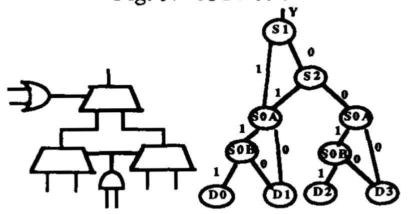
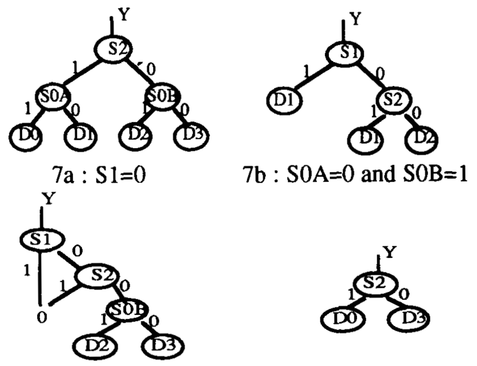


Fig. 6: ACT2 cell

Variations or degenerated patterns can be obtained by fixing some input variables in the basic cell. Some degenerated patterns for ACT1 are shown in figure 7, all these diagrams are called ACT1 patterns. ACT2 patterns are obtained in the same way.



7a : S1=0 7b : S0A=0 and S0B=1

7c: D0=0 and D1= 7d: S1=0, S0A=1 and S0B=0

Fig. 7: Degenerated ACT1 patterns

3.2. Decomposition methods for ACT1.

Both BDDs and ROBDDs will be used. As shown in the previous section, BDDs have been constructed separately for each function without any sharing. The splitting variable is chosen at each node to reach the leaves as fast as possible. Therefore, this representation will be used for speed optimization. On the contrary, the ROBDDs are constructed while looking for a maximum amount of shared subtrees. This is obtained by choosing an adequate input ordering (based on global kernels) as well as using negative edges. Therefore, these ROBDDs will be used for area reduction.

3.2.1. Speed oriented mapping on BDDs.

The mapping is performed in a greedy manner starting from the leaves, and based on two properties:

Property 1: Any 2-level BDD can be mapped on a single ACT1 cell.

Proof: All nodes of the 2-level degenerated BDD given in figure 7a are labeled by distinct variables. Therefore, any 2-level BDD can be mapped on an ACT1 pattern.

Property 2: A 3-level BDD shown in figure 8 can be mapped on a single ACT1 cell if N1 and N3 are roots of isomorphic subtrees.

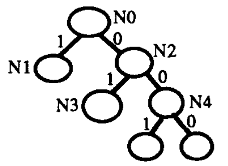


Fig. 8: Property 2 illustration

The decomposition algorithm is the following:

For all BDDs

(1) If the BDD depth is 2 (property 1)

or If the BDD depth is 3 and if property 2 is satisfied then an ACT1 cell is identified; else go to (2).

(2) Select a node of maximum depth in the BDD, identify the largest sub-BDD which is isomorphic to one of the ACT1 patterns. This identification is made using the two above properties. Create a subfunction; go to (1).

3.2.2. Area oriented mapping on ROBDDs.

The starting point is now the ROBDD of the functions in which intensive sharing has been performed.

As multiple fanouts cannot be covered by ACTEL cells, the ROBDDs have to be decomposed into small pieces by cutting them at each multiple-fanout nodes. The covering phase may be quite more complex.

To avoid a too complex covering phase, a preliminary expertise has been performed and showed that in 90% of the cases, the covering quality is not affected by using only the 2 patterns shown in figure 9. Therefore only these 2 patterns will be used.

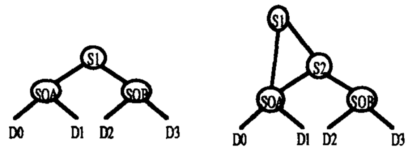


Fig. 9: ROBDD patterns M1 and M2

The mapping is performed in two steps. In a first step, an initial virtual decomposition is performed by declaring potential or virtual cutnodes. In a second phase definite cutnodes define the final ACT1 cells.

step 1: A node N becomes a virtual cutnode if either N has more than one predecessors or N is the root of an ACT1 pattern (M1 or M2).

step 2: The following 2 rules are applied to define the definitive cutnodes:

Rule 1: If all the predecessors and the 2 successors of a virtual cutnode are also virtual cutnodes, then this node is no more a virtual cutnode.

Rule 2: If the 2 successors of a virtual cutnode N are virtual cutnodes and if all its predecessors except one are also virtual cutnodes, then the virtual cutnode N is no more a virtual cutnode and its predecessor which was not a virtual cutnode becomes one.

The definition of definitive cutnodes is done as follows:

(1) Reduce the number of virtual cutnodes applying rule 1.

(2) Change the location of the virtual cutnodes applying rule 2.

(3) Apply again rule 1 on the virtual cutnodes.

Due to lack of space, the justification of the rules will not be discussed.

4. Results and present work.

4.1. Results on ACT1.

Table 2 shows the results of ASYL-FPGA on some classical benchmarks. Three techniques have been runned. The first one (resp. second) (B, resp. R) uses the BDD (resp. ROBDD) based method explained in the previous sections. The third technique is a classical library-based solution (L) [12]. Amap, XAmap, mis-pga and proserpine correspond respectively to [2], [3], [6] and [4]. These last results concern only the area oriented solution.

Name	ASYL C.P.	ASYL Area	A-map	XA-map	mis-pga	pro-ser
5xp1	B 36	R 37	53	58	35	50
9sym	B 41,4	R 17	-	-	17	-
9symmlB	40,6	R 17	102	110	17	-
alu2	B 39,8	R 99	196	210	175	-
bw	B 25,8	R 59	94	102	54	63
clip	B 42,2	R 44	53	57	43	68
count	L 47,6	R 43	62	55	39	-
duke2	L 53	L 165	169	188	158	175
f51m	B 34,4	R 35	58	64	39	59
misex1	B 23,4	R 21	-	-	16	24
misex2	B 32	R 43	-	-	38	42
rd53	B 25,8	R 10	-	-	-	-
rd73	B 36	R 20	-	-	25	-
rd84	B 36,8	R 30	93	101	36	65
sao2	B 39,8	R 45	-	-	49	-
vg2	L 47,6	R 37	37	39	30	45
z4ml	B 34,4	R 9	-	-	14	-

Table 2: Published and ASYL results

The important result is that the expected interest of ROBDD for area reduction and BDD for speed oriented mapping is demonstrated. The ROBDD based approach gives clearly the best area, whilst the BDD approach yields a good critical path. These results also confirm that the library-based solution (L) usually never lead to the best result. Compared to other published results, ASYL requires respectively 31,5%, 34%, 2,6% and 20,3% less cells than Amap, XAmap, mis-pga and proserpine (for the existing benchmarks).

4.2. Extension to ACT2.

In the ACT2 cell, the two control lines S0A and S0B are ANDed together. The same splitting variable appears on the two arcs successor of the root. Therefore, a special preprocessing must be applied for the BDD and ROBDD representations of the functions.

4.2.1. Speed oriented mapping on BDDs.

The different steps of the ACT1 synthesis process are still valid. Only the selection of the splitting variables is modified. Indeed, the splitting variable has to be the same for the two nodes successor of the same node. Therefore a pair of successive splitting variables is chosen instead of a single splitting variable. This is done as follows:

- Sort the variables according to the decreasing number of occurences.

- Within a window, consider all pairs of variables:

  * evaluate the complexity of the 4 cofactors (in terms of number of variables)

  * select a pair minimizing the Max {complexity of the cofactors}

  * refine this selection by minimizing the sum of the complexity of the 4 cofactors

- Create the 4 subfunctions corresponding to the 4 cofactors and iterate recursively the process on these 4 subfunctions.

4.2.2. Area oriented mapping on ROBDDs.

The startup point is the optimized ROBDD based solution found for ACT1, thus ignoring the ACT2 constraint on the splitting variable. According to the order respected while constructing the ROBDDs, in most of the cases the splitting variable on the arcs successor of the same node will be identical. If this variable does not appear on one of these arcs, then it is introduced in a fictious way by inserting dummy nodes. This is illustrated in figure 10.

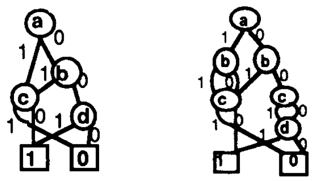


fig. 10: Insertion of dummy nodes.

The mapping algorithm is afterwards similar to the one described for ACT1.

5. Conclusion.

Most of the FPGA structures commercially available today are not well suited for a library cell based technology mapping. Moreover, it is inefficient to separate the factorization step from the technology mapping step. The whole synthesis approach has to be coherent and targeted to the final device. For the multiplexer based cells, it is quite obvious that the Binary Decision Diagrams are the ideal initial representations. But an important distinction has to be made between the non shared Binary Decision Diagrams (BDD) and Reduced Ordered Binary Decision Diagrams where the sharing is one of the main goals. The former leads to excellent critical path oriented mapping whilst the latter yields minimum area solutions. The results shown in this paper clearly demonstrate this point.

References

[1] Bryant R.E. - "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Trans. on Computers, Vol C-35, no.8, August 1986, pp. 677-691.

[2] Karplus K. - "Amap : a Technology Mapper for Selector-based Field Programmable Gate Arrays", 28th ACM/IEEE DAC, San Francisco, June 1991, pp. 244-247.

[3] Karplus K. - "Using if-then-else DAGs to do Technology Mapping for Field-Programmable Gate Arrays", Computer Research Labotatory, Technical Report, Sept 1990.

[4] Ercolani S., De Micheli G.- "Technology Mapping for Electrically Programmable Gate Array", 28th ACM/IEEE DAC, San Francisco, CA, June 1991, pp. 234-239.

[5] ACT family Field programmable gate array data book - April 1990.

[6] Murgai R., Brayton R.K., Sangiovanni-Vincentelli A. - "An Improved Synthesis Algorithm for Multiplexor-based PGAs", 29th ACM/IEEE DAC, Anaheim, CA, June 1992, pp. 380-386

[7] Besson T., Bouzouzou H., Crastes M., Saucier G. - "Synthesis on Multiplexer-based Programmable Devices using (Ordered) Binary Decision Diagrams", proc. EUROASIC, Paris, June 1992, pp 8-13.

[8] Fujita M., Matsunaga Y., Kakuda T. - "On Variable Ordering of Binary Decision Diagrams for the Application of Multi-level Logic Synthesis", EDAC 91, pp. 50-54.

[9] Saucier G., Poirot F. - "A Good Input Ordering for Circuit Verification Based on Binary Decision Diagrams", Proc. EUROASIC, Paris 1991, pp 385-387.

[10] Ishiura N., Sawada H., Yajima S. - "Minimization of Binary Decision Diagrams Based on Exchanges of Variables", Proc. ICCAD 1991, pp. 472-475.

[11] Malik S. et al. - "Logic Verification Using Binary Decision Diagrams in a logic Synthesis Environment", Proc. ICCAD, Nov. 1988, pp. 6-9.

[12] : R.K. Brayton et al., "MIS, a multilevel logic optimization system", IEEE trans on CAD, pp. 1062-1081, November 1987.