Practical Logic Synthesis for CPLDs and FPGAs with PLA-Style Logic Blocks

Kenneth Yan
ZettaCom
2833 Junction Ave., Suite 200, San Jose, CA 95134
kyan@zettacom.com

ABSTRACT

In some modern FPGAs and CPLDs, PLA(programmable logic array)-style logic blocks can be used as the storage elements for improved logic density and performance. PLA-style logic blocks were originally deployed in the early PLDs. Due to recent research developments in the FPGA community such as [6] and [4], PLA-style logic blocks are becoming an effective storage alternative in FPGAs. This paper presents an approach with clustering and functional decomposition to implement the circuit using the minimum number of PLA-style logic blocks. One important feature is that it simultaneously considers the routing resource reduction for better circuit performance after place-and-route. In order to effectively use PLA-style logic blocks in large clusters, functional decompositions are used to decompose large clusters so that the encoding functions and base functions can be mapped into PLA-blocks. Furthermore, implicit representation[5] of the crucial steps in the functional decomposition is used to consider 1) number of inputs. 2) number of product terms. 3) number of outputs required for the PLA-block synthesis. We have developed an algorithm called PLA_SynT that can be used in the logic synthesis flow for CPLDs and FPGAs with PLA-blocks. MCNC benchmarks are used to test PLA_SynT and the experimental results are compared with TEMPLA [1]. PLA_SynT shows 10.24% improvement over TEMPLA[1], in terms of the number of PLA-blocks needed to implement the circuit. PLA_SynT also shows 14.41% improvement over EMB_Syn[3] in circuit performances while maintaining comparable circuit areas.

1. INTRODUCTION

Most commercial FPGAs provide lookup-table structures as logic cells. However, LUTs are normally limited to a maximum of four or five inputs. For wide functions such as decoders, LUTs must be combined across logic cells with the accompanying delays from the interconnect routing. A PLA can directly implement a set of logic functions with wider inputs and outputs. It consists of a set of inputs and corresponding input complements, and two stages of logic. The first AND gates form a set of product terms that consist of any of the inputs or their complements. The second stage is an array of OR gates, each of which forms a logical sum of any number of the product terms. Therefore, PLA-block can be characterized by k inputs, m product terms, and n outputs. In order to utilize these PLA-style logic blocks wisely as logic functions, one can extract single-output and multiple-output logic blocks from the circuit for PLA-block mapping and consider the corresponding delays that the mapping imposes on the circuit.

Technique used in [1] is combinatorial in nature. To go beyond the combinatorial limit during logic synthesis for programmable logic devices with PLA-blocks, we decompose the large clusters so that they can be considered in the PLA-block implementation as well. EMB_Syn[3] performs logic resynthesis for area minimization. It targets for small number of EMB blocks and does not optimize for timing. Since interconnect delays tend to dominate the overall performance, we want to consider the routing resource dependency in our synthesis approach and converge on a design that is less routing-intensive.

In this paper, we present a general PLA-block synthesis method with clustering and functional decomposition for the PLDs and FPGAs with PLA-blocks. The novelty lies in (1) the integration of structural clustering and functional decomposition and (2) Implicit computation of the crucial steps in functional decomposition that enables us to consider efficiently inputs, product terms, or outputs of the candidate subcircuit to be mapped into the PLA-block. (3) Heuristic for routing resource dependency estimation to guide the synthesis steps for better circuit timing after place-and-route.

Preliminaries are presented first in Section 2. Section 3 discusses the PLA_SynT algorithm with MFFC/MFFS clustering and functional decomposition for FPGAs and CPLDs with PLA-blocks. Experimental results and conclusion are given in Section 4 and Section 5 respectively.

2. PRELIMINARIES

A boolean network can be represented as a directed acyclic graph (DAG) where each node represents a logic gate, and a directed edge (i, j) exists if the output of gate i is an input of gate j. A *cone* at v, denoted as C_v, is a subgraph consisting of v and its predecessors such that any path connecting a node in C_v and v lies entirely in C_v. The notation of $input(C_v)$ is also used to represent the set of distinct nodes outside C_v which supply inputs to the gates in C_v. A *maximum cone* at v, also the fanin network of v, denoted as N_v, is a cone consisting of v and all of its predecessors. A cone C_v is said to be *k-feasible* if and only if $| input(C_v) | \leq k$.

Given a network N with a source s and a sink t, a cut (X,X') is a partition of the nodes in the network such that $s \in$ X, $t \in$ X' and no nodes in X' provide input to any node in X. Clearly X' may be considered as a cone at t inside network N. A *cut-set* of a cut is the set of all nodes v such that $v \in$ X and v drives a node in X'. If the size of the cut is no more than k, the cut is said to be *k-feasible*. Given a cone C_v rooted at v, the *maximum-volume k-feasible cut* is the

k-feasible cut (X,X') with the largest number of nodes in X'. A *fanout-free cone* (FFC) at v, denoted FFC_v is a *cone* of v such that for any node $u \neq v$ in FFC_v, $output(u) \subseteq FFC_v$.

Definition 1 (MFFC) *In a directed acyclic graph which represents a combinational circuit, the maximum fanout-free cone (MFFC) of v, denoted $MFFC_v$, is an FFC of v such that for any non-PI node w, if output(w) $\subseteq MFFC_v$, then $w \in MFFC_v$.*

For a given node set S, a *fanout-free subgraph* of S, denoted FFS(S), is a subgraph consisting of S and its predecessors such that any path connecting a node in the subgraph and a node v, $v \in S$, lies entirely in the subgraph. Also, any node $u \notin S$ in $FFS(S)$, $output(u) \subseteq FFS(S)$.

Definition 2 (MFFS) *The maximum fanout-free subgraph of S, denoted $MFFS(S)$, is an FFS of S such that for any non-PI node w, if output(w) $\in$ MFFS(S), then $w \in$ MFFS(S). S is called the root set of MFFS(S), and the nodes in S are the root nodes of MFFS(S). An MFFS(S) is said to be a k-input m-output MFFS if input(MFFS(S)) = k and $\mid S \mid = m$.*

Definition 3 Multiple-Output Decomposition Given a function vector $f = (f_1, ..., f_p)$ and a partition of its n input variables into the bound set $BS = x_1, ..., x_b$, and the free set $FS = y_1, ..., y_{n-b}$, multiple-output functional decomposition determines an *encoding function assignment* $A_f = d_1, ..., d_q$ and the *base functions* $g_1, ..., g_p$ such that for each j, $1 \leq j \leq p$,

$$f_j(x_1, ..., x_b, y_1, ..., y_{n-b}) = g_j(d_1^j(x_1, ..., x_b), ..., d_{c_j}^j(x_1, ..., x_b), y_1, ..., y_{n-b}) \quad (1)$$

where $\forall_i d_i^j \in A_f$.

Problem *Given a combinational network N and the FPGA or CPLD with PLA-blocks or hybrid architecture with PLA-blocks and LUTs, with architecture as described in the previous section, use decomposition and clustering so that the circuit area and performance can be optimized.*

	Input: seed node, I/O specifications, bound set size
	Output: MFFS and bound set variables
	procedure shared_input_root_set(v, k, m, l)
1	/* v=seed node, k=input size,
2	m=output size, l=bound set size */
3	compute max-volume k-feasible cut of v
4	E = set of extremal nodes of the cut
5	m_root_set = select m nodes from E
6	with the biggest MFFCs below the cut
7	and/or with the most level of logic
8	compute MFFS(m_root_set) below the cut
9	$retval.mffs\_nodes$ = MFFS(m_root_set)
10	$retval.seed\_nodes = v$
11	$retval.num\_input = k$
12	$retval.num\_output = m$
13	$retval.bound\_set$ = l inputs shared by max outputs
14	in m_root_set
15	**return** ($retval$)

Table 1: Template for shared_input_root_set

	Input: seed node, PLA-block input/output specifications
	Output: MFFS with the best gate reduction per PLA-block
	procedure root_set_for_large_MFFS(v, k, m)
1	/* v=a seed node, k=PLA-block input size,
2	m=PLA-block output size */
3	compute max-volume k-feasible cut of v
4	E = compute extremal nodes of the cut
5	m_root_set = select m nodes from E
6	with the biggest MFFCs below the cut
7	and/or with the most level of logic
8	compute MFFS(m_root_set) below the cut
9	$retval1.gate\_reduction$=size of MFFS(m_root_set)
10	$retval1.num\_PLA\_block\_needed$=1
11	$retval1.mffs\_nodes$= MFFS(m_root_set)
12	compute MFFS(E) below the cut
13	num_PLA_block=number of PLA_blocks to cover MFFS(E)
14	$retval2.gate\_reduction$=size of MFFS(E)/num_PLA_block
15	$retval2.num\_PLA\_block\_needed$=$num\_PLA\_block$
16	$retval2.mffs\_nodes$= MFFS(E)
17	if ($retval1.gate\_reduction > retval2.gate\_reduction$)
18	**return** ($retval1$)
19	else
20	**return** ($retval2$)

Table 2: Template for root_set_for_large_MFFS

3. PLA_SYNT ALGORITHM

3.1 Introduction to PLA_SynT

Given a combinational network, PLA_SynT extracts large single-output and multiple-output fanout-free logic blocks and covers them entirely or partially by PLA-blocks. It consists of 4 main phases: (1) It computes a set of MFFCs and MFFSs in the network for possible PLA-block implementations (Section 3.2) For area minimization, we search for clusters that lead to reduction of large number of gates. For timing optimization, we search for clusters that have multiple-level logic and hence reduces the need for interconnect resources when they are mapped into PLA-blocks. (2) The *infeasible* MFFCs are functionally decomposed to cater the PLA-block configurations (Section 3.3). (3) It selects the clusters from the PLA-block candidates in (1) and (2) for mapping. (4) if the selected PLA-block candidate is a *feasible* MFFS, compute a larger *infeasible* MFFS from it and perform functional decomposition to see if better area reduction can be achieved.

We want to decompose the *infeasible* MFFCs and *infeasible* MFFSs so that more opportunities to reduce the circuit area can be explored. After functional decomposition, if the numbers of inputs and outputs of the encoding functions or the base functions are compatible with the PLA-block specifications, they become PLA-block candidates. If no such functional decomposition is possible, the functional decomposition routine returns failure and the *infeasible* MFFC/MFFS cannot be considered in the selection process for PLA-block mapping. After we have a set of candidates for PLA-block implementation, we want to select MFFCs or MFFSs for mapping such that the circuit area can be minimized. Currently, a greedy approach is used. We iteratively select the MFFC or MFFS with the largest reduction without overlapping with the other selected MFFCs or MFFSs. For a *feasible* MFFS, we incrementally increase its input size(cut size) and compute a larger infeasible MFFS below the cut using the heuristic in Table 1. Next, multiple-output functional decomposition is used to decompose the MFFS and check if better gate reduction can be achieved. To speed up the process of recomputing the MFFCs and MFFSs after mapping a cluster into a PLA-block, we only compute the MFFCs and MFFSs for those root nodes or root sets that are affected by by the deletion of a cluster in the previous iteration. Therefore, the MFFC and MFFS recomputations are incremental and does not impose a heavy penalty in run time.

3.2 Cluster Computation

Computation of MFFC and MFFS can be found in [2]. The construction of MFFS depends heavily on how we choose the root set. The root sets are important because we want (1) a set of large *feasible* MFFSs. (2) a set of large MFFSs that can be covered by multiple PLA-blocks. (3) a set of large infeasible MFFSs that are decomposable and produce good area reduction during PLA-block mapping. (4) a set of MFFSs with the most level of logic to reduce interconnect dependency. The effectiveness of multiple-output functional decomposition is determined by (1) The number of encoding functions that are shared by outputs. (2) The complexity of the encoding functions. (3) The complexity of the base functions. (1) leads directly to good reduction of the number of inputs to all base functions g_j in Equation 1. This gives higher feasibility with different PLA-block configurations. (2) and (3) are important in PLA-block mapping because simple encoding and base functions can be mapped by a smaller number of gates/less level of logic and result in smaller/faster circuit. In order to exploit the multiple-output functional decomposition, a good root set should produce maximum sharing of the encoding functions when multiple-output functional decomposition is performed. Bound set variables for decomposition can be pre-determined to help achieving this goal. Root sets of size m are m-root sets. Since it is computationally expensive to enumerate all $O(n^m)$ m-root sets for large circuits of n nodes, a heuristic is used to find the root sets with shared inputs efficiently. They are based on the following observation.

Observation: Input variables that are shared among the output functions are encouraged to be used as bound set variables because they are likely to produce encoding functions that are inherently shared by output functions.

Heuristic : Shared Input Root Set from Shadow Nodes (Table 1) First, compute a *maximum-volume k-feasible cut(v)* for a node v. Node v is called the seed node. Find the *shadow nodes* and *extremal nodes* of the *k-feasible cut(v)* as follows:

Definition 4 Shadow Node Given a node v and a *k-feasible cut(v)*. The nodes that form such cut are the *cut set*. A node s is called the *shadow node* if all paths from the primary inputs to node s go through the *cutset* of the *k-feasible cut(v)*.

Definition 5 Extremal Node The set of *extremal nodes* E is a subset of shadow nodes S such that for each node v in E, $u \in output(v)$ implies that $u \notin S$.

We set k equal to the number of inputs in the PLA-block configuration. After finding a set of extremal nodes, we compute the MFFC below the cut for each extremal node. We select the m extremal nodes with the largest MFFCs below the cut as m-root for MFFS. To take decomposition into consideration, the shared input variables are used as bound set which is used in functional decomposition later. If timing is more critical, we select the m extremal nodes that have the most level of logic below the cut. To help us make the trade-off decision, we devise the objective function: $\alpha \times AR + (1-\alpha) \times LL$. AR is the area reduction by mapping the MFFCs. LL is the total number of logic level for the selected extremal nodes from the cut. α is adjustable based on the optimization goal during bound set selection. The high density logic usually overloads the interconnect structures in today's FPGAs and CPLDs. So we try to reduce the routing resource dependency by putting the maximum amount of multiple-level logic into the PLA-blocks. After the root sets are computed, we compute the MFFS that is below the *k-feasible* cut. The MFFS can be covered by the PLA-block regardless of whether $|\ input(MFFS(m\text{-}root))| > k$. In other words, the MFFS(m-root) below the cut is guaranteed to be *k-feasible* because only k inputs in the cut set are needed to generate the MFFS outputs. We devise a heuristic in Table 2 to compute a set of large feasible MFFSs and large MFFSs that can be covered by multiple PLA-blocks.

3.3 Decomposition for Infeasible Clusters

Functional decomposition methods based on *Binary Decision Diagrams (BDD)* were proposed in papers like [7]. In IMODEC[7] multiple-output functional decomposition, due to the *implicit* computation in the crucial steps of their algorithm and manipulating only the *preferable* functions, their decomposition algorithm is very efficient and it produces good decomposition results. In addition, every unique path from the root to leaf nodes in the implicit representation in BDD corresponds to a product term. Therefore the number of product terms can be evaluated quickly using standard BDD routines. This allows us to eliminate a lot of functional decomposition that leads to infeasible MFFSs due to the product term constraint violation. The multiple-output functional decomposition technique in [7] is implemented in this paper. Let $num\_global\_class =$ the number of global compatible classes for multiple-output functions. In IMODEC, $log(num\_global\_class)$ is the minimum number of encoding functions required to decompose the multiple-output functions. Therefore we only continue the decomposition if $log(num\_global\_class)$ does not exceed the number of PLA-block outputs. If necessary, we try functional decomposition for the different PLA-block configurations. The size of the bound set is the same as the number of PLA-block inputs. We fill the bound set with shared input variables of the MFFS. If the number of PLA-block input exceeds the number of shared variables, we fill the remaining bounded variables with the non-shared input variables.

The encoding function mapping or base function mapping in PLA_SynT is done by using SIS routines for area minimization. The number of gates used for the encoding functions or base functions is subtracted from the size of the MFFC or MFFS to obtain the gate reduction for PLA-block implementation.

4. EXPERIMENTAL RESULTS

PLA_SynT has been implemented with C language on SUN SPARC workstation and is integrated into the Berkeley SIS system. Our experimental setup is focused on using PLA_SynT in postmapping process. The rugged.script in SIS is used to generate the initial mapped circuits with 2-input gates. Table 3 shows the CPU run time for running PLA_SynT on a Sun Ultra 60 360MHz with 1 gigabyte of memory. PLA-blocks used in the experiment have 10 inputs, 12 product terms, and 4 outputs. PLA_SynT's result and result in [1] are presented in Table 4. Same set of MCNC benchmarks reported in TEMPLA[1] are used in the comparison. We set α to 0.75 in our optimization objective function. 10.24% improvement in area reduction is obtained for PLA_SynT over TEMPLA[1]. Table 5 shows the distribution of the clusters that are mapped into PLA-blocks. The percentages of infeasible MFFCs and infeasible MFFSs are 19.4% and 13.6%.

The decomposition of large infeasible cluster does contribute to the final mapping of the selected candidates into PLA-blocks. We also compare experimental results with EMB-Syn[3]. Altera MAX+PLUSII is used to place and route the synthesized circuits and the circuit delays are reported in Table 6. The initial number of LUTs used in the circuit before running EMB_Syn and PLA_SynT are also presented. Same number of EMBs are used to map the circuits. α in our optimization objective function is set to 0.05. Overall, PLA_SynT shows a 14.41% improvement in delays in final circuit implementation while using the similar number of LUTs.

	run time		run time		run time
alu4	596	ex5p	334	fir	893
apex4	679	misex3	389	fsm8_8_13	130
clma	1303	pdc	2401	pmac	1034
cps	448	s38417	3482	psdes	459
dalu	284	seq	1045	sort	632

Table 3: Sample CPU run times in seconds running PLA_SynT on a Sun Ultra 60 360MHz with 1 gigabyte of memory.

	TEMPLA	PLA_SynT	% reduction
alu4	155	138	10.96
apex4	193	181	6.21
clma	957	903	5.64
cps	120	107	10.83
dalu	64	64	0
ex5p	132	101	23.48
misex3	154	127	21.26
pdc	618	527	14.72
s38417	603	502	16.74
seq	229	221	3.49
fir	249	232	6.82
fsm8_8_13	49	48	2.04
pmac	237	228	3.79
psdes	151	136	9.93
sort	138	119	13.76
total	4049	3634	10.24

Table 4: PLA-block number comparison between TEMPLA and PLA_SynT.

	total	feas. MFFS	feas. MFFS	infeas. MFFC	infeas. MFFS
alu4	138	40	37	43	18
apex4	181	57	52	33	39
clma	903	257	356	161	129
cps	107	47	39	11	10
dalu	64	28	19	8	9
ex5p	101	38	34	19	10
misex3	127	40	45	23	19
pdc	527	232	167	74	54
s38417	502	189	194	82	37
seq	221	69	38	64	50
fir	232	37	129	47	19
fsm8_8_13	48	12	14	22	0
pmac	228	74	49	35	70
psdes	136	42	38	37	19
sort	119	26	34	47	12
total	3634	1188	1245	706	495
Dist.		32.6%	34.2%	19.4%	13.6%

Table 5: Distribution of the clusters that are mapped into PLA-blocks.

5. CONCLUSION

	Original	Post-PLA_SynT			Post-EMB_Syn		
	# LUT	# LUT	# EMB	Delay (ns)	# LUT	# EMB	Delay (ns)
C5315	673	601	5	50.2	599	5	59.7
C6288	555	514	5	91.9	514	5	106.1
C7552	850	778	7	52.4	778	7	69.2
C880	142	97	5	53.1	97	5	65.1
alu2	192	60	5	52.1	61	5	64.2
alu4	326	205	5	52.9	203	5	67.9
apex6	300	253	5	41.2	253	5	46.1
apex7	88	59	5	31.9	59	5	35.1
des	1367	1252	8	70.2	1257	8	78.1
i10	1166	1016	8	151.2	1002	8	163.1
pair	641	534	5	41.1	534	5	49.5
total	6300	5369	63	688.2 -14.41%	5357	63	804.1 1

Table 6: Delay comparison between PLA_SynT and EMB_Syn

In this paper we have presented a practical PLA-block logic synthesis approach for the FPGAs and CPLDs with PLA-blocks. It combines functional decomposition and structural clustering to map subcircuits into PLA-blocks. It provides the tradeoff factor to optimize for timing or area. The structural clustering is based on MFFC/MFFS and the root set selection takes the routing resource reduction, area reduction, and the feasibility for PLA-block mapping into consideration. Experimental results are given to show the effectiveness of the proposed algorithm.

6. REFERENCES

[1] J.H. Anderson and S.D. Brown, "Technology Mapping for Large Complex PLDs", Proc. of International conference on Computer-Aided Design, November 1998, pp. 698-703.

[2] J. Cong, P. Li, S.-K. Lim, T. Shibuya, and D. Xu, "Large Scale Circuit Partitioning with Loose/Stable Net Removal and Signal Flow Based Clustering", Proc. of International Conference on Computer-Aided Design, November 1997, pp. 441-446.

[3] J. Cong and K. Yan, "Synthesis for FPGAs with Embedded Memory Blocks", Proc. of International Symposium of FPGA 2000, pp. 75-82.

[4] F. Heile and A. Leaver , "Hybrid Product Term and LUT Based Architecture Using Embedded Memory Blocks", Proc. of FPGA 1999, pp. 13-16.

[5] T. Kam, T. Villa, R. Brayton, and A.L. Sangiovanni-Vincentelli, "Implicit Computation of Compatible Sets for State Minimization of ISFMS's", IEEE Transactions on Computer-Aided Design of Integrated and Systems, Vol. 16, No. 7, July 1997, pp.657-676.

[6] A. Kaviani and S. Brown, "The Hybrid Field Programmable Architecture", IEEE Design and Test, April-June 1999.

[7] B. Wurth, K. Eckl, and K. Antreich, "Functional Multiple-Output Decomposition: Theory and an Implicit Algorithm", Proc. of Design Automation Conference 1995, pp. 54-59.