An Efficient Performance Improvement Method Utilizing Specialized Functional Units in Behavioral Synthesis

Tsuyoshi SADAKATA Yusuke MATSUNAGA

Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan
e-mail: {sadakata,matsunaga}@c.csce.kyushu-u.ac.jp

Abstract— This paper proposes a novel Behavioral Synthesis method that improves performance of synthesized circuits utilizing specialized functional units effectively. Specialized functional units (e.g. Multiply-Accumulator) are designed for specific operation patterns to achieve shorter delay and/or smaller area than cascaded basic functional units. Almost all conventional methods cannot use specialized functional units effectively under a total area constraint because of their less flexibility for resource sharing. The proposed method makes it possible to solve module selection, scheduling, and functional unit allocation problems utilizing specialized functional units in practical time with some heuristics, and to reduce the number of clock cycles under total area and clock cycle time constraints. Experimental results show that the proposed method has achieved up to 35% and on average 14% reduction of the number of cycles with specialized functional units in practical time.

I. Introduction

In Behavioral Synthesis, utilizing specialized functional units (e.g. Multiply-Accumulator) is effective in improving performance of synthesized circuits. However, module selection and functional unit allocation utilizing specialized functional units under a total area constraint are difficult problems. Specialized functional units (for short, SFUs) with shorter delay and/or smaller area can be designed for specific operation patterns than cascaded basic functional units (e.g. adder, multiplier). For example, carry-save adder (for short, CSA) based functional units for addition based operation patterns [1]. Therefore, the proper use of SFUs leads to a performance improvement under resource constraints. However, almost all SFUs are optimized for specific operation patterns, and they may not be able to execute different operation patterns effectively. Hence, when considering resource sharing, using SFUs does not always lead to a performance improvement, or may affect the performance.

Several related works for SFUs have been proposed. However no method can solve module selection and functional unit (for short, FU) allocation under a total area constraint in practical time. Landwehr and Mardwedel et al. proposed an Integer Linear Programming (for short, ILP) based method for module selection, scheduling, FU allocation, and binding with SFUs subject to minimize the number of cycles under resource constraints [2][3]. The method cannot solve large problems in practical time because of a computational complexity of an ILP problem. Corazao et al. proposed a heuristic method using template mapping techniques for scheduling and FU allocation subject to reduce the product of the number of cycles and the length of CCT [4]. Bringmann et al. proposed a heuristic method using hill climbing approaches for module selection, scheduling, and FU allocation [5]. Both heuristic methods do not consider resource constraints, and cannot control the increase of resources.

This paper proposes a novel heuristic Behavioral Synthesis method for module selection, scheduling, and FU allocation utilizing SFUs subject to reduce the number of clock cycles under CCT and total FU area constraints. The main contribution of the proposed method is that results with fewer cycles can be obtained utilizing SFUs in reasonable computational time with introducing a new heuristic metric for module selection.

II. Preliminaries

A synthesis target is assumed to be a disjoint Control/Data-Flow Graph [6] that includes a Control-Flow Graph (for short, CFG) and some Data-Flow Graphs (for short, DFGs) related to each CFG node. The number of CFG nodes is denoted as $L \in N$ (N denotes the set of numerical numbers), and the corresponding DFGs are denoted as $DFG_l(V_l, E_l), l = 1, 2, ..., L$. For each DFG_l, the set of immediate predecessor nodes of a node $v \in V_l$ is denoted as *PARENTS(v)*. The set of all predecessor nodes of a node $v \in V_l$ is denoted as *PRED(v)*.

A FU library is assumed to be given and includes the information of FUs (area, maximum delay, corresponding operation patterns, etc.). The set of FU types is denoted as F. The area of an FU is defined as the function $f_{area}(fu) : F \mapsto R^+$ (R^+ denotes the set of positive real numbers). The number of cycles to be required for an FU in executing operations is defined as the function $f_{cycle}(fu) : F \mapsto N$. For FUs implemented as combinational circuits, $f_{cycle}(fu) = 0$. The maximum delay of a FU is defined as $f_{maxdelay}(fu) : F \mapsto R^+$, the maximum delay from primary inputs to first level flip-flops is defined as $f_{maxdelay\_in}(fu) : F \mapsto R^+$, and the maximum delay from last level flip-flops to primary outputs is defined as $f_{maxdelay\_out}(fu) : F \mapsto R^+$.

The set of nodes in each DFG with data-dependencies that can be executed on a FU is defined as an **operation block** (for short, OB). An OB concept is introduced for handling SFUs. OBs are assumed as the targets of scheduling and FU allocation instead of DFG nodes.

Definition 1 *An operation block $b = V_l'$ is a subset of the node set V_l of $DFG_l(V_l, E_l)(l = 1, 2, ..., L)$ satisfying the following all conditions.*

1. $|V_l'| \geq 1$
2. $|V_l'| > 1 \Rightarrow \forall v_1' \in V_l', \exists v_2' \in V_l', (v_1', v_2') \in E_l \vee (v_2', v_1') \in E_l$
3. $\forall v' \in V_l', \forall v'' \in$ *PARENTS*$(v') -$ (*PARENTS*$(v') \cap V_l'$), $|$*PRED*$(v'') \cap V_l'| = 0$
4. $\exists v' \in V_l', \forall v'' \in V_l'(v'' \neq v')$, *there is a path from* v'' *to* v'

The second condition describes that the induced subgraph of V_l' is a connected graph. The third condition is required to satisfy a convexity. The forth condition restricts the number of output nodes to be one. The all OBs in DFG_l are denoted as B_l. The data-dependencies between OBs can be decided based on data-dependencies of DFG nodes.

Module selection problem is to decide a **module set vector** (for short, MSV). A MSV is defined as $msv \in N^{|F|}$. The notation $msv[fu](fu \in F)$ represents the number of *fu* to be selected. The total area of a MSV is $\sum_{fu \in F} f_{area}(fu) \cdot msv[fu]$. A MSV is called **feasible** if all operations in all DFGs can be executed on the modules included in the MSV, and given total area and CCT constraints are met. It is obvious that results of module selection should be feasible MSVs (for short, FMSVs). Scheduling problem is to decide cycles where the executions of operations in each DFG are started. The result of DFG_l can be defined as $schedule_l(b) : B_l \mapsto N$. FU allocation problem is to decide FU types that operations in each DFG are executed on. The result of DFG_l can be defined as $allocation_l(b) : B_l \mapsto F$.

The results of module selection, scheduling, and FU allocation have to satisfy data-dependencies between OBs, a total FU area constraint $\alpha \in R^+$, and a CCT constraint $\beta \in R^+$.

The objective is to minimize the sum of the products of the number of cycles of each scheduled DFG and a weight that is given as a constraint for each DFG: $\sum_{l=1}^{L} weight(l) \cdot max_{b \in B_l}(schedule_l(b) + f_{cycle}(allocation_l(b)))$. The weight is defined as $weight(l) : N \mapsto N$. It is used to represent the execution count of DFGs that correspond to the bodies of loop structures. The costs of memories or interconnections are ignored to analyze the ideal performance with only FUs.

III. Proposed Method

The proposed method consists of three parts: 1) enumeration of maximal FMSVs, 2) operation block selection, 3) resource constrained scheduling and FU allocation. For each part, a heuristic algorithm is introduced. The overall process is as follows. At first, the set of FMSVs are enumerated based on a heuristic metric. Then, OB selection and scheduling/FU allocation are performed iteratively for each MSV. Finally, the best result is selected.

A. Enumeration of Maximal Feasible Module Set Vectors

The proposed method focuses on **maximal** FMSVs. A MSV is called **maximal** if any additional module causes the area constraint violation. It is obvious that if scheduling and FU allocation can be solved exactly for a MSV, the number of cycles with non-maximal FMSVs will be always fewer than or equal to the result with maximal FMSVs that cover the non-maximal FMSVs. Therefore, only maximal FMSVs should be considered under an area constraint.

The total number of all maximal FMSVs can be exponential to both $|F|$ and an area constraint. Therefore, enumerating all maximal FMSVs is not practical. To avoid the increase of computational time, proposed method restricts the number of enumerated maximal FMSVs based on **unit** FMSVs. A MSV is called **unit** if $\forall fu \in F, msv[fu] \leq 1$. For a maximal FMSV $msv_{maximal}$, a corresponding unit FMSV msv_{unit} satisfying the following condition can be considered.

$$\forall fu \in F, msv_{unit}[fu] = \begin{cases} 0 & (\text{msv}_{\text{maximal}}[\text{fu}] = 0) \\ 1 & (\text{msv}_{\text{maximal}}[\text{fu}] \geq 1) \end{cases}$$

It is obvious that there is a unique unit FMSV corresponding to a maximal FMSV. Therefore, the proposed method enumerates all unit FMSVs instead of maximal FMSVs, evaluates unit FMSVs based on a heuristic metric, and processes only maximal FMSVs that can be obtained from several unit FMSVs with better evaluated values.

An evaluation metric for each unit FMSV is the estimated minimal number of cycles that can be achieved by maximal FMSVs from a unit FMSV. The number is estimated with the linear interpolation between the results of resource constrained scheduling with a unit FMSV and an unlimited FMSV. An unlimited FMSV is defined as follows.

$$\forall fu \in F, msv_{unlimited}[fu] = \begin{cases} 0 & (\text{msv}_{\text{unit}}[\text{fu}] = 0) \\ \infty & (\text{msv}_{\text{unit}}[\text{fu}] = 1) \end{cases}$$

Fig. 1 shows the linear interpolation scheme. At first, with a unit FMSV as a resource constraint, OB selection and scheduling/FU allocation are performed. The result is considered to be the maximum number of cycles that can be obtained with maximal FMSVs from the unit FMSVs. Next, with an unlimited FMSV, OB selection and scheduling/FU allocation are performed, and the maximum total area for each cycle is calculated. The result is considered to be the minimum number of cycles that can be obtained with maximal FMSVs from the unit FMSVs when ignoring a total area constraint. Then, with interpolating lineally by a straight line between two results, the numbers of cycles under total area constraints between the area of a unit FMSV and an unlimited FMSV are estimated.

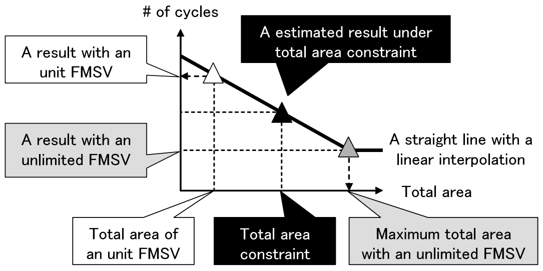

Fig. 1. Estimation scheme for the best result of maximal FMSVs from a unit FMSV

After the estimation for all unit FMSVs, from the unit FMSV with the smallest estimated result, ϵ maximal FMSVs are enumerated and evaluated ($\epsilon \in N$ is a given constant). OB selection and scheduling/FU allocation are preformed for all enumerated maximal FMSVs.

The maximum size of all unit FMSVs is still exponential to $|F|$. If no unit FMSV violate an area constraint, all unit FMSVs will be enumerated. Therefore, the limit of $|F|$ is about 16 practically. The number of maximal FMSVs enumerated from unit FMSVs can be restricted by giving an appropriate ϵ value.

B. Operation Block Selection

OB selection is performed for each DFG in a hill climbing manner. Initially, the block set with only simple OBs is prepared. An OB is called **simple** if the size of a block is 1. An OB is called **complex** if the size of a block is more than 1. Then, each complex block is added to a current block set, and scheduling/FU allocation is performed with each new block set. If an added complex OB shares the same node with simple OBs included in a current block set, the simple OBs will be removed. If an added complex OB shares the same node with complex OBs included in a current block set, the addition will be ignored. The block set with the best objective value is selected as a new current block set. While the objective value is improved, the processes are iterated. The maximum execution number of scheduling/FU allocation is $1 + |B_{complex}|(|B_{complex}| - 1)/2$.

C. Scheduling and Functional Unit Allocation

The algorithm of scheduling and FU allocation is based on static list-scheduling algorithm [6]. The priority of an OB ready list is the shortest time from a block to the end block of DFG when ignoring resource constraints. It can be calculated by ASAP scheduling algorithm in $O(|B'|)$ if blocks are sorted in topological order beforehand (where B' is a given block set from OB selection). The blocks in a ready list are sorted based on a priority in $O(|B'| \cdot log_2(|B'|))$. Then, from the top of a ready list, blocks are scheduled and allocated the fastest FU available at the time. The scheduling and allocation are performed in $O(C_l \cdot |B'| \cdot |F|)$. Therefore, the whole computational time is in $O(|B'| \cdot (log_2(|B'|) + C_l \cdot |F|))$. Consequently, operation block selection, scheduling, and allocation are performed in $O(|B_{complex}|^2 \cdot |B'| \cdot (log_2(|B'|) + C_l \cdot |F|))$ for each MSV.

IV. Experimental Results

The proposed method has been implemented as a C++ program and evaluated with three benchmarks: differential equation solver (diffeq) from HLSynth'92 benchmark [7], a squared error/bidirectional prediction calculation function (bdist2) for motion estimation in MPEG-2 encoder program, and a DCT function (jpeg_fdct_islow, for short, fdct) in the JPEG program from MediaBench [8]. The numbers of operation nodes are 11 for diffeq, 43 for bdist2, and 138 for fdct. All complex OBs that satisfy the condition for a CSA-based FU construction algorithm [1] were enumerated beforehand. Then, all combinational SFUs corresponding to enumerated OBs were synthesized by Synopsys Module Compiler under timing constraints 3 or 6 ns with the cell library for HITACHI 0.18 μm CMOS process technology from VDEC. TABLE I shows the results. The functions without "8 bit" show the results for 32-bit FUs. The results are used in following experiments.

At first, diffeq was synthesized by an ILP approach based on [2] and the proposed method without a heuristic module selection algorithm on an AMD Opteron 275 + 8GB RAM + CentOS 4.5 machine, and the results are compared to evaluate OB selection and scheduling/FU allocation algorithms. In module selection of the proposed method, all maximal FMSVs have been enumerated. The given CCT constraint is 6 ns, and the total area constraint is assumed to be between 110,000 and 190,000 μm^2. The results with an ILP based approach were obtained by formulating and solving an ILP problem with a commercial ILP solver, ILOG CPLEX. Although the details of the results are omitted for the lack of spaces, the proposed method has achieved the same number of cycles as those with an ILP based method in practical time. Under the area constraints, the numbers of cycles without SFUs range from 9 to 7, and with SFUs from 8 to 6. The reductions of the number of cycles are from 1 to 3. An ILP based method have required about 2,000,000 seconds at the worst case. On the other hand, the proposed method have required at most 2 seconds. The results show that utilizing SFUs are effective in reducing number of cycles and the proposed method has achieved the same numbers of cycles as those with an ILP approach in reasonable time.

Next, dist2 and fdct were synthesized by the proposed method without/with a heuristic module selection algorithm on Intel Xeon 5140 + 8GB RAM + CentOS 4.5 machine, and the results are compared to evaluate the heuristic module selection algorithm. The proposed method without a heuristic module selection algorithm enumerates all maximal FMSVs in module selection. The result with an ILP based method cannot be obtained in practical time. TABLE II and III show the number of enumerated unit and maximal FMSVs. TABLE IV and VI show the results of the number of cycles. TABLE V and VII show computational times under CCT = 6 ns. The computational times under CCT = 9 ns are omitted for the lack of spaces. However, these are almost all the same as those under CCT = 6 ns. ϵ for a heuristic module selection algorithm is given as 1,000. When comparing the number of cycles between the results with/without SFUs ("w" and "w/o"), the reductions of cycles have been achieved at almost all conditions. Especially, under α = 120,000 μm^2 and CCT = 6 ns for bdist2, "w/o" has achieved 35% reduction. In addition, under α = 110,000 μm^2 for bdist2 and α = 120,000 μm^2 for fdct, the feasible results have been obtained only with SFUs. When comparing the number of cycles between the results with/without a heuristic module selection algorithm ("ALL" and "OUR"), the differences are at most one cycle and "OUR" have achieved the same results at almost all conditions. On the other hand, computational times of "OUR" are shorter than those of "ALL" under $\alpha \geq 130,000\mu m^2$ for bdist2 and under $\alpha \geq 160,000\mu m^2$ for fdct. Because the proposed method has to obtain the results with unit and unlimited FMSVs for each unit FMSV, the computational times of the proposed method where the number of unit FMSVs are more than the half of the number maximal FMSVs have been longer. However, when the number of maximal FMSVs is larger, the proposed method has been able to obtain the results in less computational time, maximally 1/50 for bdist2 and 1/10 for fdct. These results show that the proposed method has been able to obtain almost all the same results as those with all maximal FMSVs in practical time.

V. Conclusion

This paper has proposed a novel performance improvement method utilizing SFUs effectively. The proposed method restricts the maximal FMSVs to be enumerated with a new heuristic metric, and makes it possible to obtain results in practical time. Experimental results show that proposed method has achieved almost all the same results with SFUs as those without SFUs in practical time. Our future work includes improving module selection algorithm for the more reduction of computational times, and evaluating with more precise delay and area costs considering memories and interconnections.

Acknowledgements

This work is supported by the Grant-in-Aid for JSPS Fellows No.17–6203 of the Ministry of Education, Science, Sports and Culture (MEXT). This work is supported by the Grant-in-Aid for Scientific Research (A) No.19200004 of the Ministry of Education, Science, Sports and Culture (MEXT). This research was supported by JST, CREST. The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration by Hitachi Hokkai Semiconductor Ltd. and Dai Nippon Printing Corporation.

References

[1] J. Um and T. Kim, "An optimal allocation of carry-save-adders in arithmetic circuits," *IEEE Trans. on Computers*, vol. 50, no. 3, pp. 215–233, Mar. 2001.

[2] B. Landwehr and R. D. P. Marwedel, "Oscar: Optimum simultaneous scheduling, allocation and resource binding based on integer programming," in *Proc. of the conference on European design automation*, 1994, pp. 90–95.

[3] P. Marwedel, B. Landwehr, and R. Dömer, "Built-in chaining: Introducing complex components into architectural synthesis," in *Proc. of the Asia South Pacific Design Automation Conference*, 1997, pp. 599–605.

[4] M. R. Corazao, M. A. Khalaf, L. M. Guerra, M. Potkonjak, and J. M. Rabaey, "Performance optimization using template mapping for datapath-intensive high-level synthesis," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 877–888, Aug. 1996.

[5] O. Bringmann and W. Rosenstiel, "Cross-level hierarchical high-level synthesis," in *Proc. of the conference on Design, Automation and Test in Europe*, 1998, pp. 451–456.

[6] D. D. Gajski, *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Pub, 1992.

[7] N. Dutt, "Current status of hlsw benchmarks and guidelines for benchmark submission," in *HLSynth'92 Benchmark*, 1992.

[8] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "Mediabench: a tool for evaluating and synthesizing multimedia and communications systems," in *Proc. of the ACM/IEEE international symposium on Microarchitecture*, 1997, pp. 330–335.

TABLE I

FUNCTIONAL UNIT INFORMATION

Function	DC = 3 ns		DC = 6 ns		Used or not		
	D (ns)	A (μm^2)	D (ns)	A (μm^2)	diffeq	bdist2	fdct
$a+b$ or $a-b$	2.48	24284	4.60	19384	*	*	*
$a+b$	2.44	10460	4.83	7994	*	*	*
$a+b$ (8 bit)	1.79	1567	1.79	1567		*	
$a-b$	2.22	11927	4.43	9155	*	*	*
$a*b$	3.82	93036	5.58	77821	*	*	*
$a<b$	0.95	4662	0.95	4662	*	*	
$a \geq b$	1.00	4685	1.00	4685			*
$a>>b$	2.12	10829	2.12	10829		*	*
$a<<b$	1.77	10045	4.43	9155			*
$a-b*c$	4.18	97881	5.69	81116	*		
$a-b-c$	2.46	16397	4.65	14070	*		
$a*b+c$	4.12	93327	5.64	79764	*	*	*
$a-b*c-d$	4.62	98704	5.58	88758	*		
$(a-b)-c*d$	4.37	101268	5.48	86592	*		
$(a-b*c)-d*e$	4.99	169789	6.09	159437	*		
$a+b+c$	2.40	14976	4.87	122296		*	*
$a+b+c$ (8 bit)	1.86	2650	1.86	2650		*	
$a+b+c+d$ (8 bit)	2.11	3802	2.23	3694		*	
$a+b+c+d+e$ (8 bit)	2.21	4631	2.32	4524		*	
$a*b+c+d$	4.48	95578	5.49	84541			*
$a+b+c+d$	2.32	24660	4.84	16827			*
$a*b+c+d+e$	4.82	100877	5.86	90770			*
$a-b+c$	2.16	16818	4.74	13363			*

DC: Maximum delay constraint given to Synopsys Module Compiler

D: Maximum delay, A: Total cell area, w/o: Results without SFUs, w: Results with SFUs
ALL: Results with the proposed method excluding a heuristic module selection algorithm
OUR: Results with the proposed method including a heuristic module selection algorithm

TABLE II

# OF ENUMERATED UNIT/MAXIMAL FMSVS (BDIST2)

Max area (μm^2)	UNIT		MAXIMAL	
	w/o	w	w/o	w
110,000	0	11	0	11
120,000	6	224	6	327
130,000	20	1415	31	3773
140,000	41	4699	88	25468
150,000	65	11001	195	122562
160,000	86	20116	360	464922
170,000	100	30802	595	1473367
180,000	109	41124	897	4052582

TABLE III

# OF ENUMERATED UNIT/MAXIMAL FMSVS (FDCT)

Max area (μm^2)	UNIT		MAXIMAL	
	w/o	w	w/o	w
120,000	0	5	0	5
130,000	6	47	7	55
140,000	23	206	39	299
150,000	44	628	124	1201
160,000	68	1552	310	3974
170,000	88	3174	659	11292
180,000	102	5712	1251	28975
190,000	110	9080	2234	68458
200,000	113	13193	3763	151268
210,000	114	17603	6051	315858
220,000	119	22032	9356	629034

TABLE IV

# OF CYCLES (BDIST2)

Max area (μm^2)	CCT = 6 ns				CCT = 9 ns			
	ALL		OUR		ALL		OUR	
	w/o	w	w/o	w	w/o	w	w/o	w
110,000	-	29	-	29	-	29	-	29
120,000	28	18	28	18	27	16	27	16
130,000	19	16	19	16	16	13	16	13
140,000	17	14	17	14	13	12	13	12
150,000	16	13	16	14	12	11	12	11
160,000	15	13	15	13	11	10	11	11
170,000	15	12	15	13	11	10	11	11
180,000	14	12	14	12	11	10	11	10

TABLE V

COMPUTATIONAL TIME (SEC) (BDIST2, CCT = 6 NS)

Max area (μm^2)	ALL		OUR	
	w/o	w	w/o	w
110,000	1	1	1	1
120,000	1	1	1	1
130,000	1	7	1	7
140,000	1	48	1	18
150,000	1	219	1	40
160,000	1	848	1	72
170,000	1	2736	1	111
180,000	1	7588	1	149

TABLE VI

# OF CYCLES (FDCT)

Max area (μm^2)	CCT = 6 ns				CCT = 9 ns			
	ALL		OUR		ALL		OUR	
	w/o	w	w/o	w	w/o	w	w/o	w
120,000	-	68	-	68	-	66	-	66
130,000	51	46	51	46	50	44	50	44
140,000	44	37	44	37	40	36	40	36
150,000	40	36	40	36	38	34	38	34
160,000	38	36	38	36	38	32	38	32
170,000	38	34	38	34	36	30	36	30
180,000	38	34	38	34	36	30	36	30
190,000	38	34	38	34	36	30	36	30
200,000	38	34	38	34	36	30	36	30
210,000	38	34	38	34	36	30	36	30
220,000	38	34	38	34	36	30	36	30

TABLE VII

COMPUTATIONAL TIME (SEC) (FDCT, CCT = 6 NS)

Max area (μm^2)	ALL		OUR	
	w/o	w	w/o	w
120,000	1	1	1	1
130,000	1	1	1	1
140,000	1	3	1	7
150,000	1	13	1	23
160,000	1	45	1	45
170,000	1	133	1	85
180,000	1	357	1	151
190,000	2	869	1	253
200,000	3	1970	1	373
210,000	5	4121	1	677
220,000	8	8218	1	857