Co-evaluation of FPGA Architectures and the CAD System for Telecommunication

Tsunemasa Hayashi, Atsushi Takahara, Ken-nosuke Fukami†
NTT System Electronics Laboratories
† NTT Science and Core Technology Laboratory Group
e-mail: {hayashi, taka, fukami}@aecl.ntt.co.jp

Abstract— **We propose an FPGA architecture for next generation B-ISDN telecommunications systems. Such a system requires an FPGA in which an over 10K gates circuit can be implemented and that has a clock cycle rate of 80MHz. While the FPGA architecture has been discussed in terms of its circuit structure, we consider the circuit structure of the FPGA with its CAD tools. We evaluate several FPGA logic-element structures with a technology mapping method. From our experiments, the Multiplexor based logic-element is found to be suitable for implementing such a high-speed circuit using the BDD-based technology mapping method.**

I. Introduction

The Broadband Integrated Services Digital Network (B-ISDN) based on the Asynchronous Transfer Mode (ATM) is aimed at providing various services such as multi-media telecommunication and inter-LAN connections. To do so, B-ISDN must support a wide variety of information sources and transport protocols. Consequently, there is a strong requirement for a "flexible transport device" that can support all protocols, both known and unknown [17].

A flexible FPGA-based telecommunications system has been proposed [13] that aims to support a wide variety of services, some of which exist now with the others intended for introduction in the future. General-purpose FPGAs can not be used in telecommunication because their operation speed is not high enough, so application specific FPGAs are required. Ohta et al. proposed a telecommunication-specific FPGA called "PROTEUS" [13]. The circuits implemented in PROTEUS operate at 20 MHz. However, considering that next-generation telecommunications systems such as B-ISDN will have a transmission rate of 622 Mbps, 80 MHz operation speed is required. This is hard to achieve with existing FPGAs.

When we consider speeding up FPGAs, it is important to closely examine the FPGA architecture and the CAD tools used in its development. An FPGA consists of reconfigurable logic and wiring elements. For logic elements, the two typical architectures are the Look Up Table (LUT) type and the Multiplexor (MUX) type. In general, the LUT type is more flexible, while a circuit implemented using a MUX type is faster [5]. This observation is not precise because the performance of the circuit implemented in an FPGA varies also with its functionality [5] and the implementation technology (such as SRAM, Fuse) is different. Moreover, the quality of the circuits largely depends on the performance of the CAD tools used to design the FPGAs. The quality also depends on the structure of the wiring elements. Even if each logic element can operate at high speed, the final circuit could be slow if the mapping tool maps the logic elements inefficiently. So, in FPGA design, it is very important to consider the relationship between the FPGA architecture and CAD tools. A number of previous works have discussed mapping technologies for specific architectures [11, 10, 12, 3, 1, 16], most of which are LUT based. However, there has been little discussion of FPGA performance in terms of how it is related to both the architecture and CAD tools.

There are numerous parameters that have to be considered in deciding the architecture of an FPGA. We examine FPGA architecture in two steps. Firstly, the logic elements of the architecture are considered with an evaluation a technology mapping method. Secondly, the wire structures are considered with a logic element structure and placement and routing methods. In each step, design candidates are evaluated using practical examples.

In this paper, we propose FPGA architectures developed on the basis of a co-evaluation of the architecture and CAD tools. In Section 2, we examine the architecture of the logic cell as well as the mapping technique. We compare LUT type to MUX type and choose a MUX type for telecommunications application. We propose several MUX-type logic cell architectures and present a suitable mapping method that is based on Binary Decision Diagrams (BDDs) [4]. In Section 3, we present results of an evaluation of the candidates that can be used efficiently with BDD-based mapping. The evaluation was done using ATM transmission circuits. In Section 4, we consider the possibility of speeding up FPGAs by reducing wire delay and present an advanced FPGA architecture that has MUX type logic cell and a clustered structure. Section 5 concludes the paper.

II. Co-Evaluation of Architecture and CAD System

The quality of the circuit implemented on the FPGA depends largely on the degree to which the characteristics of the FPGA architecture can be efficiently utilized as well as on the CAD system. We must first consider the logic elements, so we must examine them along with the mapping process to improve FPGA performance.

A. FPGAs in a Telecommunications System

Here, we are concerned with utilizing FPGAs in a telecommunications system. Ohta et al. reported that a digital communications system consists of submodules that can be categorized into a limited number of types and these modules are very simple and small [13]. In that report, they stated that in order to implement such modules efficiently, it is reasonable to prepare small and simple logic elements. Their logic cell architecture consists of four 3-input LUTs and a 5-input gate. The number of LUT inputs is smaller than that in a general-purpose FPGA [18] in which the LUT has 4 or more inputs.

B. Logic Cell Architecture and Mapping Algorithm

There are two typical logic cell structures, the LUT type and MUX type. Examples of LUT- and MUX-based logic cells are depicted in Fig. 1. In Table I, the area, delay and kinds of implementable logic functions are summarized.

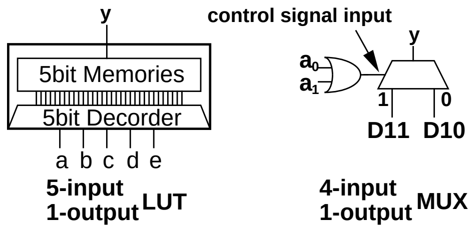

Fig. 1. Examples of LUT- and MUX-based FPGAs

TABLE I
Characteristics of LUT type and MUX type

Cell type	area (# of Tr.)	delay (ns)	implementable logic
5-In 1-Out LUT type	582	4.0	all logic expressed by 5 variables
4-In 1-Out MUX type	25	0.25	$y = (a_0 + a_1)D01 + \bar{a_0}\bar{a_1}D00$

Gould et al. reported, however, that when the circuit is implemented in a LUT-based FPGA, each LUT implements simple logic with few inputs [7]. This means the LUT type is used redundantly by existing mapping algorithms. Comparing [15] and [16], we found that the MUX type is more effective than the LUT type. So, the MUX type is suitable for telecommunications systems comprising small fan-in logic functions.

C. MUX-based logic cell architecture

The next problem is how to map the given circuit on MUX-type logic cells. Recently, Binary Decision Diagrams (BDDs) have been used to handle Boolean functions efficiently [4]. As Fig. 2 shows, the BDD representation is very similar to a MUX circuit. Here, we use

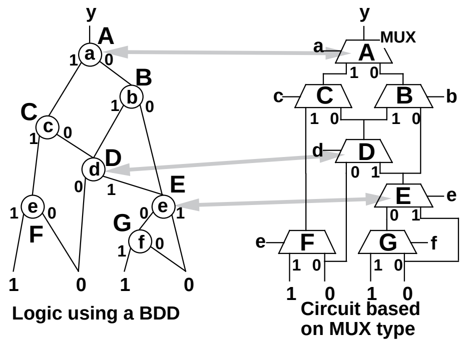

Fig. 2. Relationship of a BDD node and a MUX

BDD representation to express a circuit based on MUX-type architecture because this can represent the correlation between the structure of MUX-based circuits and the logic functions naturally. As a result, when mapping, the logic implemented on the FPGA can be processed easily and smoothly.

If a simple MUX is used, a BDD node is realized by a MUX. Here, we discuss how to improve the performance of the MUX-type logic cell. If more BDD nodes can be implemented in a cell, the critical path can be shortened and the number of MUXs can be reduced, so the delay and the area could be improved. As shown in Fig. 3(a), each logic cell (A∼G) implements a single BDD node, and the number of MUX levels in the critical path is 5 (G→E→D→B→A). If two BDD nodes are implemented in a MUX-type logic cell as in cell H and J in the Fig. 3(b), the number of logic cell levels in the critical path is reduced to 3 (J→H→A), and the number of logic cells is also reduced from 7 to 5.

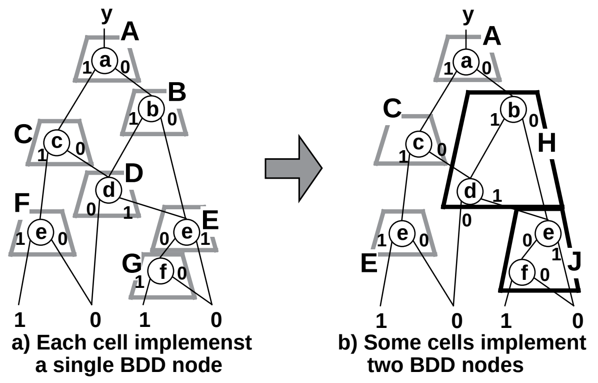
a) Each cell implemenst a single BDD node
b) Some cells implement two BDD nodes

Fig. 3. Advantaged logic cell based on MUX type

However, an increase in the delay of a logic cell is undesirable. There are two design policies for improving the architecture of logic cells.

1. Use a logic cell architecture which consists of a single MUX with an optional circuit at the control signal

part of the MUX.

2. Use a logic cell architecture in which two MUXs are stacked. This type is called two-level MUX.

Accordingly, we designed four architectures, that use either single MUX type or two-level MUX type logic cells. Figure 4 shows single MUX type T0 and its corresponding BDD representation. We designed three kinds of single

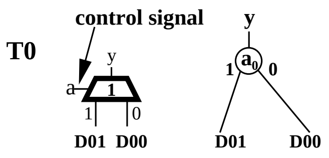

Fig. 4. Architecture of T0

MUX type logic cells. T0 has the simple structure but is less flexible because the 0-edge and 1-edge are fixed and only one BDD node can be implemented. T1 in Fig. 5 is more flexible than T0. A 2-input OR gate is added

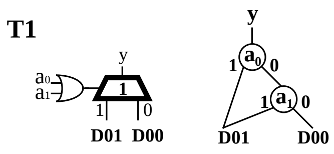

Fig. 5. Architecture of T1

to the control signal part of the MUX. Two BDD nodes can be implemented in a T1 cell but the implementable structure is either an OR structure or a single BDD node. Hence, we consider a logic cell that can implement two BDD nodes whose relationship is more kinds of logic. T2 in Fig. 6 has two control MUXs and a 2-input NAND gate at the control signal part. Signals S_{a0} and S_{a1} are control signals used by the control MUX to change the functionality of this cell.

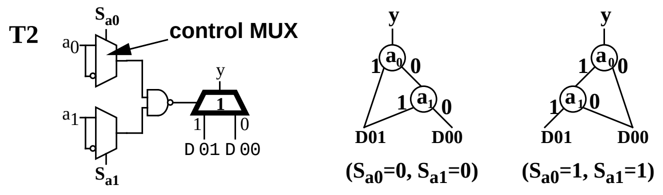

Fig. 6. Architecture of T2

Next, we consider two-level MUX type logic cells to implement more BDD nodes in a cell. T3 in Fig. 7 consists of three MUXs. T3 is a two-level MUX type in which 4 BDD nodes in three levels can be implemented. This logic cell is identical to the Act1 cell proposed by Actel [2]. Finally, we consider a logic cell in which many more BDD nodes can be implemented. T4 in Fig. 8 can implement 6 BDD nodes in four levels. This logic cell consists of three T2 logic cells.

The characteristics of these five candidates are summarized in the Table II, which shows area as the number of transistors along with two kinds of delay, DI, whose path is from input (D11, D10, D01, D00) to output (y) of a cell, and DS, whose path is from the control signal (ex. a0, a1) to the output (y).

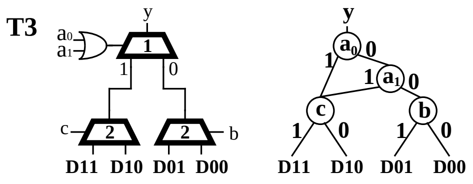

Fig. 7. Architecture of T3

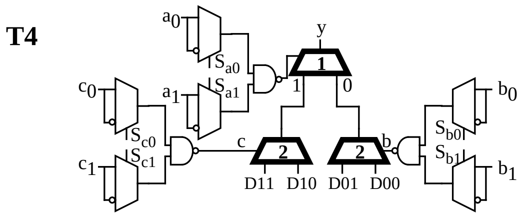

Fig. 8. Architecture of T4

TABLE II
CHARACTERISTIC OF BASIC CELL

Basic cell		T0	T1	T2	T3	T4
Circuit Area		13	25	41	52	123
Circuit Delay	DI	0.25	0.25	0.25	0.50	0.50
(ns)	DS	0.25	0.50	0.75	0.50	1.00

D. Technology Mapping for MUX-type logic cells

In our mapping method, the given logic is expressed using BDDs, and some BDD nodes among a BDD representation are implemented in each logic cell. Conventionally, the given logic is transformed to a multi-level logic representation first, but this process is eliminated in our method because the given logic is represented by BDDs, which is multi-level logic expression. Our mapping algorithm is based on BDD graph covering. In the covering process, all steps are started from the deepest BDD node to reduce the number of cells in the critical path. We also introduce the duplicate algorithm in the covering process to reduce the number of fanouts and cover BDD nodes efficiently. This algorithm is same as that used in "Chortle-crf" [6]. The covering procedure is as follows:

1. Find the deepest node among BDD representations.
2. If that node and its parent node (the out put side) have a common fanin and
   (a) If there is a relationship by which these two nodes can be implemented in one logic cell, implement them in one cell.
   (b) Otherwise, implement deepest node in one logic cell.

When the logic is implemented on two-level MUX type cells, after covering by a single MUX type cell, three single MUX type cells are implemented in a two-level MUX type cell as shown in the Fig. 9. This algorithm can be used for each proposed MUX cell type.

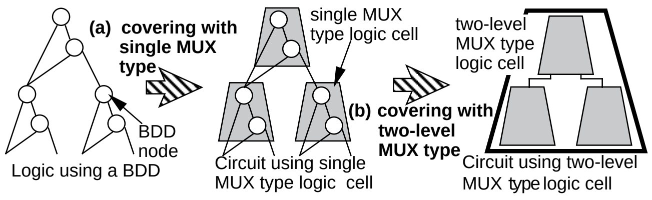
(a) covering with single MUX type
single MUX type logic cell
two-level MUX type logic cell
BDD node
(b) covering with two-level MUX type
Logic using a BDD
Circuit using single MUX type logic cell
Circuit using two-level MUX type logic cell

Fig. 9. Covering Process

III. Experimental Results

We evaluated the five proposed cells in terms of area, delay, fanout and mapping ability using practical circuits of an ATM transmission system. To evaluate the logic cell, we examined the effect of using the additional circuit at the control signal part and the effect of the two-level MUX type logic cell. Our mapping algorithm is implemented using NTT BDD tools [8, 9] in C++. Table III shows the number of logic cells and the number of cell levels in the critical path of the generated circuit. Table IV shows the exact number of transistors used in each cell and the critical path delay value according to the results in Table III.

A. Circuit Area

The circuit area is small for T0 and T1. In the two-level MUX types, the circuit area constructed by T3 cells is smaller than that constructed by T4 cells. The required conditions for getting a small circuit area are as follows.

1. The BDD nodes for expressing a given circuit by a logic cell must be covered efficiently.
2. The number of transistors in a logic cell has to be small.

Table V shows the covering ability of each cell. We con-

TABLE V
Covering ability of each logic cell

MUX level	Single			Two	
Cell name	T0	T1	T2	T3	T4
Max. # of BDD	1	2	2	4	6
Max # of cell levels	1	2	2	3	4
CE	1.00	0.52	0.57	0.42	0.35

sider first the maximum number of BDD nodes and the maximum number of cell levels that each cell can cover. We define covering efficiency (CE) as

$$CE = \frac{N_{imp}}{N_{max}}, \tag{1}$$

where N_{imp} is the number of nodes actually implemented in one logic cell and N_{max} is the maximum number of nodes that can be used in one logic cell. If CE is 1, every logic cell is used to its limit and this means these cells are used as efficiently as possible. Of course, this is the ideal case except T0 because T0 is an exact implementation of a BDD node and its CE is always 1. Single MUX types are superior to two-level MUX types in terms of covering efficiency.

Next, we consider the required area to implement a BDD node. The required area for T4 is

$$\frac{Area\ of\ T4}{N_{max}} = \frac{123}{6} = 20.5 \tag{2}$$

and that for T3 is

$$\frac{Area\ of\ T3}{N_{max}} = \frac{52}{4} = 13.0. \tag{3}$$

These results show that T4 requires 1.6 times more area than T3 when a given BDD representation is covered by logic cells. This is the theoretical ratio. In our experimental results, the ratio is 2.1, which means T4 requires more area than that theoretically estimated for these examples. This is because the covering efficiency of T4 is lower than that of T3 and T4 requires more cells to realize a given circuit. Consequently, T4 has a redundant structure in terms of the circuit area.

For single MUX type cells T0, T1 and T2, the covering efficiency of T2 is better than that of T1, but the area of the circuits constructed by T1 cells is smaller than that of circuits constructed by T2 cells. This means that the logic cell area is small and the circuits added at the control signal part makes the covering efficiency higher. This additional circuits dose not result in a penalty in terms of area.

From the area evaluation, T1 and T0, which are single MUX types, are effective. As for the two-level MUX types (T3 and T4), the covering efficiency is low, so they are not effective in terms of area efficiency.

B. Delay

In Table IV, the delay of the circuit constructed by single-MUX type cells is smaller than the delay of the one constructed by two-level MUX types because the covering result of two-level MUX types is sometime redundant as shown in Fig. 10. When two-level MUX type cells are

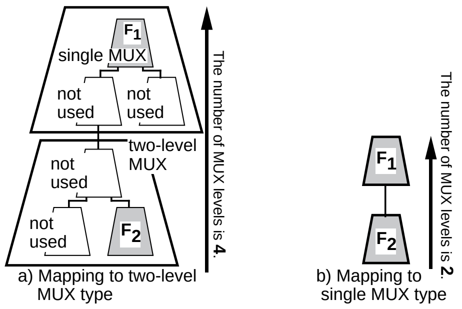
F1
single MUX
not used
not used
two-level MUX
not used
not used
F2
The number of MUX levels is 4.
a) Mapping to two-level MUX type
F1
F2
The number of MUX levels is 2.
b) Mapping to single MUX type

Fig. 10. Redundant mapping model using two level MUX

used, there are 4 MUX levels. On the other hand, when single-MUX type cells are used, the number of levels is 2. The circuit delay using two-level MUX type is twice that for single MUX type in the worst case. This indicates that single MUX types are effective in terms of circuit delay.

TABLE III
Mapping results (1)

Circuit	Inputs	BDD nodes	Number of Basic Cells					Critical path length				
			T0	T1	T2	T3	T4	T0	T1	T2	T3	T4
atm_g	43	464	464	409	320	280	212	17	15	13	9	8
sa4_t_s	53	400	400	362	302	230	211	35	31	29	19	17
shpt4_g	71	640	640	561	495	354	283	30	24	20	18	11
tim_gen	25	194	194	178	161	118	104	19	16	12	10	7
v_ais	116	273	273	272	258	174	128	9	8	7	5	5
vpoam1_g	69	708	708	643	592	422	349	33	21	19	17	11
vpoam1ts	88	790	790	742	673	464	394	53	43	34	26	22
atm_t_s	93	594	594	558	523	296	286	55	52	52	33	34

TABLE IV
Mapping results (2)

Circuit	Inputs	BDD nodes	Circuit Area (Number of Transistors)					Delay (ns)				
			T0	T1	T2	T3	T4	T0	T1	T2	T3	T4
atm_g	43	464	6032	10225	13120	14560	26076	4.25	4	3.75	4.5	4.5
sa4_t_s	53	400	5200	9050	12382	11960	25953	8.75	8	7.75	9.5	9
shpt4_g	71	640	8320	14025	20295	18408	34809	7.5	6.25	5.5	9	6
tim_gen	25	194	2522	4450	6601	6136	12792	4.75	4.25	3.5	5	4
v_ais	116	273	3549	6800	10578	9048	15744	2.25	2.25	2.25	2.5	3
vpoam1_g	69	708	9204	16075	24272	21944	42927	8.25	5.5	5.25	8.5	6
vpoam1ts	88	790	10270	18550	27593	24128	48462	13.25	11	9	13	11.5
atm_t_s	93	594	7722	13950	21443	15392	35178	13.75	13.25	13.5	16.5	17.5

For T1 and T2, the critical path of the implemented circuits is shorter than for T0. This is because T1 and T2 can implement 2 BDD nodes in a cell but T0 can implement only one BDD node. In summary, the single level MUX types (T1 and T2), which can cover multi-level BDD nodes, are better than the two-level MUX types for decreasing the delay of the critical path length.

C. Fanout

The larger the number of logic cell- or primary input-fanouts is, the more difficult it becomes to connect elements, and the wiring delay increases. Hence, we evaluate fanouts in terms of wire delay. From the results, in terms of logic cells, two-level MUX types are more effective than single MUX types. The average number of fanouts of single MUX-types is less than 3, which is a small value. In terms of the fanout of primary inputs, the single MUX type is more effective than two-level types because many nodes are duplicated when two-level MUX types are used. From the fanout evaluation, single MUX types are effective.

D. Ability of Mapping Method

Here, we compare our mapping algorithm to a previous method to evaluate our method's mapping ability. We choose ASYL [14] for the comparison. ASYL is known to generate good results for the MUX-based logic cell Act1 [2], which has the same structure as T3. Both the covering efficiency of our method and that of ASYL are 0.58[1]. This means their mapping ability is the same.

Our mapping method starts from the BDDs. The initial BDD size effects our mapping results. If the initial BDD size is small, we expect the number of logic elements to decrease. It is known that the BDD size is dependent on the order of its variables. In Fig. 11, the relationship among initial BDD size, the number of logic cells and the delay of the critical path of circuit atm_g is depicted. The sum of the primary input fanouts and the sum of logic cell fanouts are shown in Fig. 12. Both increase in proportion to initial BDD size, but the number of cell levels does not largely change. The circuit area also increases in proportion to initial BDD size, but again the number of cell levels does not change.

In this method, the quality of the circuit implemented on an FPGA depends on how much the initial size of the BDD nodes can be reduced. In general, there are many cases in which the size of BDDs grows like multipliers or other arithmetic functions. Fortunately, the circuits of transmission systems are always constructed with some simple functions and the initial size of the BDDs is typically small. So, our BDD-based approach is reasonable for applications to transmission systems.

To summarize the co-evaluation of logic cell architecture and our mapping method:

[1]This covering efficiency was calculated using the MCNC benchmark circuits used in [14]

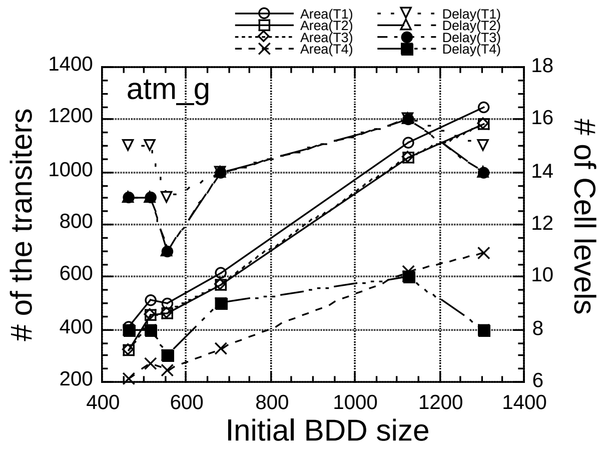

Fig. 11. The dependency of area and critical path on initial BDD size in the circuit atm_g

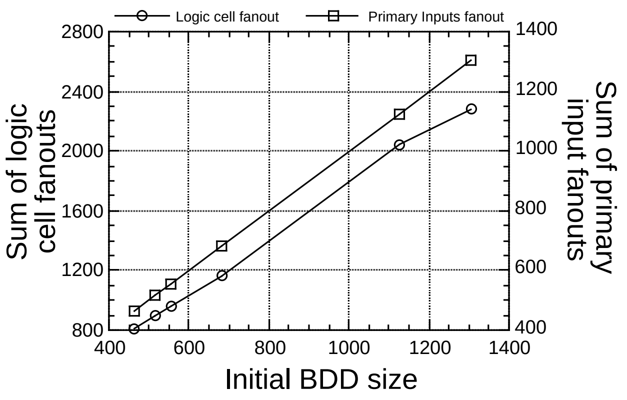

Fig. 12. Dependency of mapping result on initial BDD size in the circuit atm_g

From the area evaluation, T0 and T1, which are single MUX types, are effective because both logic cells are small and the covering efficiency is better than with the others. From the delay evaluation, T1 and T2 are effective because these cell types can implement two BDD nodes, T0 can't do this. From the fanout evaluation, single MUX types (T0, T1, and T2) are effective in terms of the number of primary input fanouts. T2 is the best cell because all evaluation results are good, and especially, the delay is the smallest among all types.

IV. An Advanced FPGA Architecture for High-speed Systems

We have evaluated both the logic cell architecture and mapping method. In this section, we take into account the wiring structure and evaluate our cell-type candidates.

A. Wiring Area Delay of FPGA

In general, the wiring area delay accounts for 50% or more of the delay in the circuit implemented on an FPGA. We estimated the wiring area delay of our example circuits using the delay of PROTEUS. Here, we assume the connection delay between cells is 2.0 ns. This is the delay of the line that connects LUTs in PROTEUS. The wiring area delays calculated under this assumption are shown in the Table VI. None of the generated circuits achieved an operation speed of 80 MHz. Therefor, we consider the reduction of the wire delay based on the above discussions in terms of the speed-up the circuit implemented on the FPGA.

B. Wiring structure for MUX type logic cells

We found that the delay of the wiring area is larger than that of the logic cell, as shown in the Table VI. The circuit implemented on the FPGA didn't run at the desired speed because we assumed that the wiring structure is the same as that in LUT-based architecture. So, we devised a suitable wiring structure for the MUX-based logic cells.

The wire delay is dependent on the wire length and the number of switches. Many switches have to put in the FPGA chip when a circuit that has many fanouts in itself is implemented. But, we don't need to put so many switches in a wire because the average number of fanouts between cells is 3 or less in our experiments. So, the low-fanout wires can be used for connections between cells. Another important point is the difference in the cell size between a LUT and a MUX type cell. We can put a lot of MUX type cells in an area equal to one LUT. For instance, the area of T2 is $\frac{1}{10}$ that of a 5-input LUT. Considering these two features, we propose the clustered FPGA structure shown in Fig. 13. In this structure, several MUX type cells are put into a logic area and are connected by a short wire and a small number of switches. So, the wiring area delay within a cluster is smaller than the wiring area delay between clusters. If cells in the critical path are put in a cluster as shown in the Fig. 13(b), the delay of the circuit is shortened.

C. Evaluation

The problems are how to place the cells in a cluster and how to connect cells in a cluster to achieve the required performance. To solve these problems, we examined the cluster size and the delay between cells in a cluster when the example circuits are operated at 80-MHz clock speed.

For this evaluation, we introduce the cell utilization ratio (CUR) which is defined as

$$CUR = \frac{m}{q}, \tag{4}$$

where m is the number of cell levels that have to be implemented in a cluster to achieve the desired clock speed and q is the total number of logic cells that can be implemented in a cluster whose size is equal to the area of the 5-input LUT. Notice that CUR must be below 1 for

TABLE VI
Delay in the wiring area

Circuit	Delay(ns)											
	T0		T1		T2		T3		T4		LUT*	
	Logic	Wire	Logic	Wire	Logic	Wire	Logic	Wire	Logic	Wire	Logic	Wire
atm_g	4.25	32	4	28	3.75	24	4.5	16	4.5	14	24	90.4
sa4_t_s	8.75	68	8	60	7.75	56	9.5	36	9	32	44	99.8
shpt4_g	7.5	58	6.25	46	5.5	38	9	34	6	20	46	132
tim_gen	4.75	36	4.25	30	3.5	22	5	18	4	12	18	76.1
v_ais	2.25	16	2.25	14	2.25	12	2.5	8	3	8	14	95.4
vpoam1_g	8.25	64	5.5	40	5.25	36	8.5	32	6	20	26	113
vpoam1ts	13.25	104	11	84	9	66	13	50	11.5	42	22	91.3
atm_t_s	13.75	108	13.25	102	13.5	102	16.5	64	17.5	66	24	106

$*$: The actual delay of the circuit in PROTEUS.

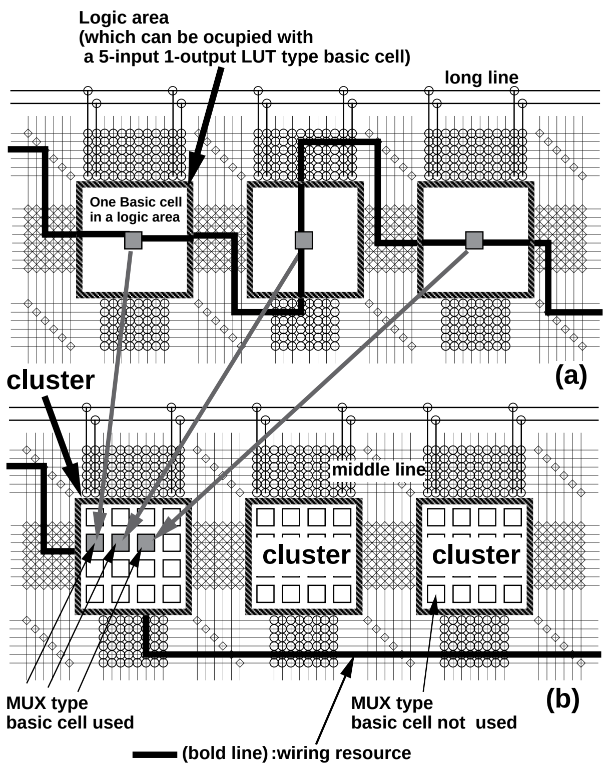

Fig. 13. Clustered FPGA structure

the circuit implemented to run at the desired speed. The smaller CUR is, the easier it becomes to design the low-delay wiring structure.

When the logic cells are placed a short distance apart, the wire delay between them becomes small. So, we introduce another ratio to express how much speed is improved by decreasing wire delay between cells in a cluster compared to the delay between clusters. This ratio is called the "speed-up ratio (SUR)" and is defined as

$$SUR = \frac{t_E}{t_I}, \tag{5}$$

where t_E is the delay between clusters and t_I is the connection delay between cells in a cluster. The larger SUR is, the less flexible the wiring between cells in the logic area becomes. This is because the wiring architecture between cells in a cluster becomes complicated when the value of t_I becomes small. Therefore we should use a cell type in which the values of both CUR and SUR are small.

Here, we evaluated two typical transmission circuits: atm_g, whose critical path length is large, and vpoam1_g, whose critical path length is medium. The results for atm_g are shown in Fig. 14. We must consider that the

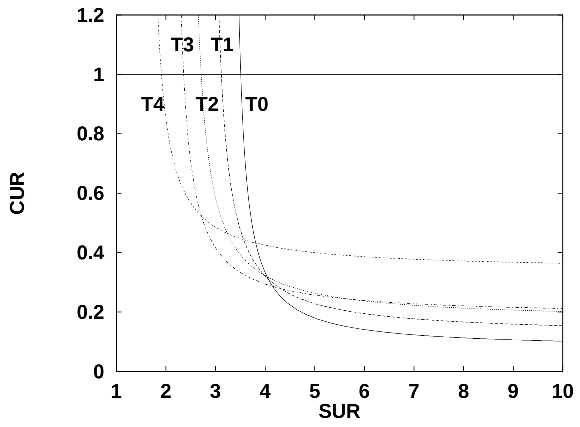

Fig. 14. $\frac{m}{q}$ calculated by mapping results on the atm_g circuit

value of CUR has to be less than 1 for the circuits to run at 80 MHz or more. In the results for the atm_g circuit, all cell types satisfy the above condition when SUR is around 4 or more. We can't see any difference among the candidate cell types because the critical path length of the atm_g circuit isn't so large.

The results for the vpoam1_g circuit are shown in Fig. 15. Obviously, T4, T2, and T1 are easy to implement in a cluster. The number of cell levels of T1 and T2 is larger so the reduction of each wire delay affects the total delay. In the case of T4, the reduction of each wire delay doesn't affect the total delay so much. In other words, the CUR is not reduced even if the SUR is large. This is

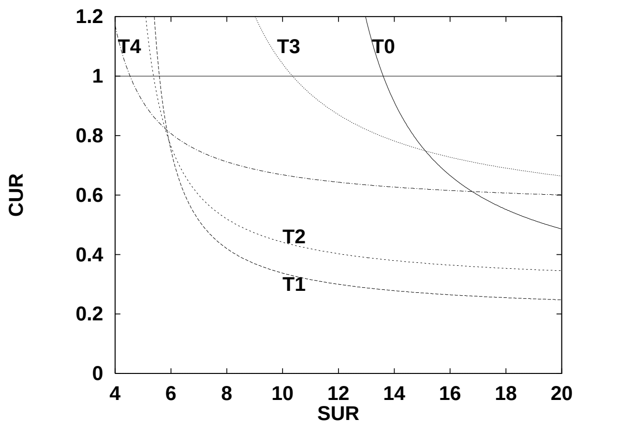

Fig. 15. $\frac{m}{q}$ calculated by mapping results on the vpoam1_g circuit

because the number of cell levels in T4 is much smaller than in the others. On the other hand, T0 has so many cell levels that the value of SUR has to be large to realize the desired speed. In the case of T3, the intrinsic delay of the logic cells is large, so the value of SUR should be large.

From the above evaluation, the single MUX-type structure (T1 and T2) is better than others considering the wire structure. Here, we don't take the primary input wiring into account. The BDD-based mapping method makes the number of fanouts of primary inputs huge, so we should consider the special wiring structure for primary inputs. We have to consider the primary-input fanouts and I/O structures in designing the precise architecture of our FPGA.

From Table VI, in the case of the MUX type, it is easy to reduce the total delay because the logic area delay of any MUX type is less than that of the LUT type. On this point, the MUX type cell is more effective than the LUT type cell.

V. Conclusions

We proposed an FPGA architecture for telecommunications systems based on the co-evaluation of FPGA logic cell architecture and its CAD system. From our evaluation using practical circuits, the combination of a MUX-type logic cell structure and BDD-based mapping is usable for telecommunications systems. The architecture of the FPGA is a set of clusters containing several MUX-type cells, a low number of fanouts and short wire elements.

Previous discussions have treated only one feature, either the FPGA architecture or the CAD tools. Therefore, the results have not always been convincing, because only part of the FPGA features were considered. Our results are interesting because the co-evaluation was done from the point of view of both architecture and CAD tools using practical examples.

Acknowledgments

We would like to thank K.Hayashi(NTT Optical Network Systems Laboratories) and N.Yamanaka(NTT Network Service System Laboratories) for their instruction.

References

[1] P. Abouzeid, L.Bouchet, K.Sakouti, G.Saucier, and P.Sicard. Lexicographical expression of boolean function for multilevel synthsis of high speed circuits. In *Proc. of SASIMI 91*, pp. 31–39, 1991.

[2] Actel. *FPGA DATA BOOK AND DESIGN GUIDE*, 1995.

[3] B. Babba and M. Crastea. Automatic synthesis on table lookup-based pgas. In *Proc. of EURO ASIC 92*, pp. 25–31, 1992.

[4] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE TRANSACTION ON COMPUTERS*, Vol. C-35, No. 8, pp. 677–691, 1986.

[5] Stephen D.Brown, Robert J.Francis, Jonathan Rose, and Zvonko G.Vranesic. *FIELD-PROGRAMMABLE GATE ARRAYS*. Kluwer Academic Publishers, 1992.

[6] Robert Francis, Jonathan Rose, and Zvonko Vranesic. Chortle-crf:fast technology mapping for lookup table-based fpgas. In *Proc. of 28th DAC*, pp. 227–233, 1991.

[7] Scott Gould, Brian Worth, Kim Clinton, Eric Millham, Fran Keyster, Ron Palmer, Steve Hartman, and Terry Zittritsch. An sram-based fpga architecture. *CICC*, pp. 243–246, 1996.

[8] Shin ichi Minato. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. *27th DAC*, pp. 52–57, 1990.

[9] Shin ichi Minato. Minimum-width method of variable ordering for binary decision diagram. *IEICE Trans. FUNDAMENTALS*, Vol. E75-A, No. 3, pp. 392–399, 1992.

[10] K. Karplus. Amap: A technology mapper for selector-based field-programable gate arrays. In *Proc.28th DAC*, pp. 244–247, 1991.

[11] Yung-Te Lai, Massoud Pedram, and Sarma B.K Vrudhula. Bdd based decomposition of logic functions with application to pga synthesis. In *Proc.30th DAC*, pp. 642–647, 1993.

[12] Rajeev Murgai, Narendra Shenoy, Robert K.Brayton, and Albert Sangiovanni-Vincentelli. Improved logic synthethis for table look up architectures. In *Proc. of ICCAD91*, pp. 564–567, 1991.

[13] N. Ohta, H. Nakada, K. Yamada, A. Tsutsui, and T. Miyazaki. Proteus: Programmable hardware for telecommunication systems. *ICCD*, 1994.

[14] Gabriele Saucier, Micchel Crasters De Paulet, and Pascal Sicard. Asyl: A role-based system for controller synthesis. *IEEE TRANSACTIONS ON COMPUTER-ADED DESIGN*, Vol. CAD-6, No. 6, pp. 1088–1097, 1987.

[15] Hiroshi Sawada, Takayuki Suyama, and Akira Nagoya. Logic synthesis for look-up table based fpgas using functional decomposition and support minimization. *ICCAD*, pp. 353–358, 1995.

[16] T.Besson, H.Bouzouzou, M.Crastes, and G.Saucier. Synthesis on multiplexer-based programmable devices using (ordered) binary descision diagrams. In *Proc.EURO ASIC*, pp. 8–13, 1992.

[17] Akihiro Tsutsui and Toshiaki Miyazaki. An efficient design environment and algorithms for transport processing fpga. *ASP-DAC*, pp. 677–691, 1995.

[18] Xilinx. *The Programmable Logic Data Book*, 1994.