PROTEUS-Lite Project: Dedicated to Developing a Telecommunication-oriented FPGA and its Applications

Toshiaki Miyazaki, Member, IEEE, Atsushi Takahara, Member, IEEE, Takahiro Murooka, Member, IEEE, Masaru Katayama, Takuki Ichimori, Kazuhiro Shirakawa, Member, IEEE, Akihiro Tsutsui, Member, IEEE, and Kennosuke Fukami, Member, IEEE

Abstract—This paper describes a project dedicated to developing an improved (in terms of usability) version of our previous telecommunication-oriented field programmable gate array (FPGA), and its applications. To achieve this goal, we adopt several challenging design strategies. First, we determine the new FPGA architecture based on a quantitative evaluation carried out to optimize the interaction between the FPGA and CAD algorithms. In addition, we create a new chip design environment that allows semi-automatic test pattern generation and cross-checking between logic and layout design. Furthermore, a dedicated CAD system is developed based on a consideration of the evaluation results and the characteristics of the FPGA. As a result of these design strategies, the FPGA and CAD system are well-balanced, and even though the FPGA has very rich routing resources, the routing process can be finished quickly without sacrificing application-circuit performance. The FPGA is applied to several reconfigurable systems for telecommunication, and is found to offer the required functions and good performance.

Index Terms—Computer aided design (CAD), field programmable gate array (FPGA), hardware/software co-evaluation, reconfigurable-systems, telecommunications, test pattern generation.

I. INTRODUCTION

TODAY, more functions are needed in each protocol layer to provide flexible multimedia services in digital telecommunication networks. However, current telecommunication systems are often constructed by dedicated hardware. This is because data transmission requires high throughput and various bit-level manipulations must be performed, which CPUs or DSPs cannot handle well. Hence, current telecommunication circuits are far from rich in terms of flexibility. To help remedy this situation, we developed a telecommunication-based FPGA called PROTEUS [1] and reconfigurable telecommunication systems utilizing it [2]. PROTEUS was developed based on an analysis showing that telecommunication circuits have the following characteristics [3],[4].

- A relatively large number of flip-flops (FFs) or latches is needed compared to the amount of combinational logic due to the need to realize counters and data pipeline.
- A lot of pattern matching operations for extracting packet or frame structures from serialized and scrambled transmission data.
- Main data streams have a single direction, which makes it easy to construct a data pipeline structure.

We showed that application circuits related to communications can easily perform real-time operations if based on the PROTEUS FPGA. Unfortunately, the balance between logic and routing resources in PROTEUS was not optimal, and its dedicated CAD system [19], especially the router, suffered from a lack of routing resources.

To overcome these problems, we started a new project aimed at developing a well-balanced FPGA called PROTEUS-Lite. This time, we adopted a smart and consistent approach in all development phases. First, we constructed an FPGA/CAD co-evaluation framework called FACT [20] to determine the architecture quantitatively. Such a framework is needed because the balance between the FPGA and CAD system is very important in realizing both easy programming and good performance. Next, we fine-tuned the architecture by considering physical implementation issues such as chip size and propagation delay. An error-free design environment featuring semi-automatic test pattern generation was created for the physical implementation phase. Finally, we developed a dedicated CAD system called PLCAD, which features heuristic tuning based on the characteristics of the FPGA architecture [5]. Consequently, PROTEUS-Lite FPGA has been improved to the point where the CAD system can easily handle it without sacrificing the good features of the original PROTEUS and so assist the user in designing application circuits.

A. Related Works

Historically, both theoretical and practical approaches have been tried to develop good FPGA architectures [6]–[11]. Brown et al. discussed the flexibility of the FPGA, which has a simple symmetrical routing architecture. They evaluated some

1It was named after our expectation that the FPGA would reduce the user’s load.
switch block architectures using a few sample circuits [6]. Chang et al. proposed a linear programming model to evaluate switch block architectures [7], [8]. Darnauer et al. examined a routability measure for FPGAs using generated random circuits [9]. Although these works give us a lot of hints, it is difficult to apply their results to actual FPGA development because their approaches are too general. Ochotta et al. designed a delay-predictable FPGA and developed a test chip [10]. From a high-performance FPGA development point of view, it is a good approach, but they did not develop its dedicated CAD system simultaneously.

Bets et al. developed an environment called VPR to evaluate FPGA architectures [11]. The environment makes it practical to examine FPGA architectures, especially their routability. Unfortunately, compared to our architecture evaluation environment, i.e., FACT system [20], the algorithm of each tool in VPR is fixed. Thus, FPGA/CAD co-evaluation is impossible with VPR. Furthermore, to the best of our knowledge, no related work mentions a strategic FPGA chip design method and its environment.

Today, many types of FPGAs are commercially available, and some of them have on-chip memories [12], [13]. The memories are often very useful when realizing telecommunication-related circuits, especially for network terminals. However, to apply the commercial FPGAs to network node equipment such as a backbone router, we need much larger memory than is offered by the FPGAs. For example, to realize a routing table look-up function, we need at least several M-bits memory, which is impossible using today’s SRAM technology. We overcome this problem by using a sophisticated board-level architecture that combines discrete high-speed SRAMs with PROTEUS-Lite FPGAs.

From the viewpoint of developing reconﬁgurable systems, application speciﬁc IC (ASIC) emulators [14], [15] are the pioneers. They are widely used today in designing custom LSIs. Although their speed is around 1 MHz, which is much faster than software simulators, they cannot be applied to real-time telecommunication data processing, which often requires at least 20 MHz. On the other hand, many reconﬁgurable custom computing machines utilizing FPGAs have been reported [16], and they are good at ﬁne-grain parallel processing, such as number crunching and digital signal processing. For instance, “Splash-2” exhibits exceptional performance: it is more than 10 000 times faster than a conventional workstation in searching a genetic database [17]. However, its single-instruction stream, multidata stream (SIMD)-based architecture is not suitable for telecommunication in which data-driven operations are common. That is, the header information in each incoming packet decides the next action or state of the equipment; the instructions are not prescheduled as in a SIMD machine. Another example is the Teramac system [18]. Designed as a general purpose computing engine, it utilizes a lot of custom-FPGAs called PLASMA, which have a unique cross-bar routing resource architecture. The board- and system-level connections of the system also adopt cross-bars so that application circuits can easily be mapped to the system. However, the performance of Teramac is around 1 MHz, similar to that of the ASIC emulators, and so fails to achieve our target.

The rest of this paper is organized as follows: Section II introduces the PROTEUS-Lite architecture; how we decided upon and developed the new FPGA architecture is discussed in Sections III and IV. Next, the PLCAD system is presented along with some experimental results in Sections V and VI. Finally, some application systems are discussed in Section VII.

II. PROTEUS-Lite FPGA

A. Chip Overview

PROTEUS-Lite is a look-up table based FPGA [6]. A PROTEUS-Lite chip is illustrated in Fig. 1. The chip comprises basic cells (BCs), I/O blocks (IOBs) and routing resources. The BCs are for circuit logic, and they are placed regularly. As shown in Fig. 1(d), a BC has four 3-input/1-output look-up tables (LUTs) and one 5-input AND (5-AND) gate. There are 136 IOBs (28 on the top, 28 on the bottom, and 40 each on the left and right). Here, the 80 IOBs on the sides are arranged in groups of four, and they construct one I/O cell (IOC) as shown in Fig. 1(c). The specifications of PROTEUS-Lite are summarized in Table I, and Fig. 2 is a photograph of the chip. The chip was fabricated using 0.5 μm three-metal layer CMOS technology and packaged in three types of LSI packages: QFP, CSP, and PGA. There are several conﬁguration modes, including “synchronous,” “asynchronous,” and “memory clear.” In addition, the standard JTAG boundary-scan feature was implemented for testing.

In short, the PROTEUS-Lite architecture features:

- a BC structure that can easily realize basic functions such as pattern matching;
TABLE I
SPECIFICATIONS OF PROTEUS-LITE CHIP

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fabrication</td>
<td>0.5 ( \mu )m CMOS</td>
</tr>
<tr>
<td>Die size</td>
<td>17.1mm x 15.3mm</td>
</tr>
<tr>
<td>Package</td>
<td>QFP, CSP, PGA</td>
</tr>
<tr>
<td>Power supply</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Usable I/O pins</td>
<td>136 (TTL-compatible)</td>
</tr>
<tr>
<td>Number of basic cells</td>
<td>10 x 28 = 280</td>
</tr>
<tr>
<td>Number of LUTs</td>
<td>1120</td>
</tr>
<tr>
<td>Number of latches</td>
<td>1672 (including I/O latches)</td>
</tr>
<tr>
<td>Configuration</td>
<td>Synchronous/Asynchronous, Memory clear</td>
</tr>
<tr>
<td>Scan mechanism</td>
<td>JTAG</td>
</tr>
</tbody>
</table>

Fig. 2. PROTEUS-Lite chip.

- a lot of regularly placed latches that can perform pipelined data processing for high data throughput;
- local lines for connecting neighboring LUTs;
- an identical IOC (or IOB) and BC connection topology for the surrounding routing resources. This homogeneous wiring topology simplifies the routing algorithm.

All features except the last one are enhanced versions of those of PROTEUS-LITE.

B. Basic Cell

Placing one 5-AND gate after four LUTs in a BC [Fig. 1(d)] makes it easy to implement the basic functions needed for telecommunication circuits. For example, we can realize an 8-bit comparator by using neighboring two BCs, as shown in Fig. 3. This basic structure of the BC is based on the telecommunication circuit analysis made when designing our previous FPGA. Detailed discussions about this are given in [3] and [4]. In addition, a latch is located at each output of the LUTs and the 5-AND gate. These latches can be programmed to latch or not latch the output. There is a switch between a LUT output and one input of the 5-AND gate. In the “off” state, the input of the 5-AND gate is disconnected from the LUT and is pulled up. Thus, the 5-AND gate can be used as a one- to five-input AND gate by controlling the switches.

C. Data Latch and Clock Signal

As shown in Fig. 1(d), data latches are located at each output of the LUTs and 5-AND gates. Users can select whether the outputs should be latched or not, when the PROTEUS-Lite chip is programmed. Furthermore, the latching of a new signal in the current clock can be controlled by the enable lines. The total number of latches including those in the IOBs is 1672. Compared to commercial FPGAs, which often have one or two latches per basic logic block, the occupation ratio of latches in the usable gates in PROTEUS-Lite is about twice. This is a unique point of our FPGA, and is very important for constructing the counters and data pipeline structure often used in telecommunication circuits.

A clock signal is delivered to all latches in a PROTEUS-Lite chip over dedicated clock lines. The clock lines are implemented in the fabrication process, and the clock-skew is less than 1 ns. In addition, each latch has a clock enable pin. Thus, we can easily design a circuit that controls a large number of subcircuits with the same timing pulse. This kind of circuit is often used in data transmission circuits.

Data pipelining is often used to achieve high data throughput in transmission circuit design. Retiming and latch insertion [21] are also known as performance-enhancing techniques that require no change in the original circuit except moving or adding latch locations. As mentioned above, PROTEUS-Lite has a lot of latches at the outputs of all LUTs and 5-AND gates, and the latches can be controlled without disturbing the routing. Thus, existing speed-boosting techniques are easily applied to our PROTEUS-Lite chip.

D. Routing Resources

The routing resource architecture was drastically changed from that in PROTEUS in order to improve routability. PROTEUS-Lite has three kinds of routing resources: local lines, middle lines, and long lines. Each routing resource plays a different role in achieving effective routing.

1) Local Lines: As shown in Fig. 4, local lines are used to connect neighboring LUTs directly. A mesh connection of all LUTs in the chip can be realized using the local lines without worrying about the boundary of each BC. The output of a LUT connects five adjacent LUTs; the LUT above, below, to the right, to the upper-right, and to the lower-right. We call this structure the “Sea of LUTs.” Compared to the other routing resources, the local lines have the smallest propagation delay. Thus, they...
should be used as much as possible to make high-performance circuits. Our preliminary experimental results showed that 33% of all nets in a circuit on average can be routed using only local lines.

2) Middle Lines and Long Lines: Details of the routing resource architecture are shown in Fig. 1(b). To connect BCs and IOBs, middle lines and long lines are provided. Long lines are used for long-distance routes. The length of each middle line is equivalent to the length of one BC, while a long line runs over half the length of the chip. A long line has several times larger delay than a middle line. Thus, middle lines should be used for shorter connections. However, long lines are better for long-distance routing.

Some commercial FPGAs have additional routing resources, such as double-length middle lines [12]. However, heterogeneous routing resources often make the routing process itself difficult. Thus, we simply concentrated on increasing the basic routing resource, i.e., the middle lines, instead of preparing other kinds of routing resources. Compared to PROTEUS, PROTEUS-Lite has many more middle lines which significantly improves routing efficiency. The switch patterns used to access the routing resources are described in Section II-D-3.

3) Switch Patterns: The switch patterns in each switch box were decided after considering the routability determined from the architecture evaluation process, which will be discussed in Section III. Fig. 5, is a simplified illustration of the essential switch patterns in PROTEUS-Lite. Each input or output line to/from a BC or an IOC is connected to both upper and lower middle lines through the switch boxes, and the switches in one pair of upper and lower switch boxes are located on different horizontal middle-line tracks. For example, the switches in switch box SB3 are located on odd tracks, and those in SB4 are located on even tracks. With these switch patterns, a signal on an odd track such as “a” in Fig. 5 can be fed to the BC or IOC through the upper switch box, and an even-track signal can be fed through the lower box. Furthermore, even-track signals, like “b” in Fig. 5, can be routed to the corresponding track in the lower middle lines via two switch boxes, SB1 and SB2, which connect vertical and horizontal middle lines. This is because the switches are placed diagonally in each switch box. These switch pattern characteristics greatly contribute to fast routing because appropriate tracks can be easily selected by simply checking the switch patterns or by referring to some pattern tables. In fact, our router with the line-search algorithm described in Section V reduces the routing time by more than 70%, compared to the basic Dijkstra-based router in the FACT system, and the routing process for a circuit, which occupies more than 50% of the FPGA area, often finishes within 15 minutes.

In addition, wires W1 and W2 in Fig. 5 can be used as bypass lines to connect upper and lower middle lines by opening switches SW1 and SW2, if the BC or IOC is unused.

III. ARCHITECTURE DESIGN

A. FACT System

The PROTEUS-Lite architecture described in Section II was assessed quantitatively using the FACT system [20]. FACT is an FPGA/CAD co-evaluation system. It can simulate ordinary circuit-design processes for FPGAs, such as technology mapping, placement, and routing. In addition, a new FPGA architecture can be defined using the architecture definition format (ADF) provided in the FACT system.

An overview of the FACT system is shown in Fig. 6. The system consists of a CAD development environment and an FPGA evaluation tool set. Each tool was created using the development environment.

The CAD development environment provides middleware incorporating various kinds of data access methods. Thus, one can easily develop original tools using the middleware by simply coding the core part.

The FPGA evaluation tools are divided into two categories: actual evaluation tools and post-processors that visualize or reorganize the results produced by the evaluation tools. Technology mapping, placement, and routing tools belong to the
former category, and they become actual CAD tools after the FPGA is realized. The latter contains graphical viewers and statistical analyzers.

The architecture is evaluated using the input circuit data, which are usually netlists structured by primitive gates and nets connecting the gates. The input circuit data are actually configured into the architecture specified in the ADF through the technology mapping, placement, and routing processes. In other words, our system can completely simulate the ordinary process in the dedicated CAD environment for an FPGA. This means the FACT system can be used as a CAD system when an evaluated FPGA is actually developed. Of course, each evaluation tool can be replaced independently, so the FACT system can also be used as a CAD algorithm evaluation environment if the architecture is fixed.

The most interesting feature of this system is that the architecture can be defined without worrying about the evaluation tools. The ADF represents an FPGA, and is open to the user. Thus, users can freely specify their own FPGA architectures. Each evaluation tool reads the ADF whenever it is invoked, and gets the architecture information. This mechanism contributes to reducing the time taken to rebuild the evaluation environment; it is unnecessary to modify each tool even if the FPGA architecture is changed slightly.

The evaluation results can be visualized or summarized using the post-processors. The post-processors are also application programs, which means users can add original post-processors to the FACT system.

B. Routing Resource Evaluation

A routing bottleneck often occurs at the inputs or outputs of each logic block, which is called a basic cell, or BC, in PROTEUS and PROTEUS-Lite. Thus, we first concentrated on improving the routing topology around the BCs.

We considered the three architecture types shown in Fig. 7. Each type has a symmetrical array structure, but with different input and output directions to/from each BC. In type A, the inputs come from both the left vertical and top horizontal routing channels, and the outputs go to both the right vertical and top horizontal channels. In type B, the inputs come only from the top horizontal channel, while the outputs return to it. In type C, both the inputs and outputs are connected to the top and bottom horizontal channels. Six circuits were realized using these architectures while the numbers of tracks in the vertical and horizontal channels and the switch pattern in each switch box were changed.

We evaluated a total of 23 architectures in the three categories. The results are shown in Fig. 7. In the graph, the x-axis, i.e., the number of switches in each cyclic pattern in the FPGA, means the switches in several switch boxes, and the y-axis represents the number of unrouted nets. The two numbers in parentheses are the numbers of horizontal and vertical tracks in each routing channel. The target area had less than 450 switches per cyclic pattern and less than ten unrouted nets. These were not strict constraints; they were decided from the standpoints of hardware cost and CAD efficiency.

In one type-A architecture, all nets were completely routed, i.e., the number of unrouted nets is zero. The architecture has completely connected switches that connect any two tracks coming into the same switch box. Thus, we only had to concentrate on the width of each channel, and the routing results indicate that the architecture has enough channel width (in this case, 20) to route the circuit data. However, we found that this architecture requires too many switches; i.e., it is too difficult to realize the desired architecture.

Type B is aimed at row-based routing because the inputs and outputs of each BC are connected to the top horizontal channel. Unfortunately, the routing results are not so good. However, there is still room to apply channel routing algorithms developed for ASICs to improve the routing. In fact, the routing tool we used is a variant of the Dijkstra-based algorithm, not a channel routing algorithm.

The type-C architecture is a modification of type B. That is, the inputs and outputs of each BC are connected to both the top and bottom horizontal channels. The type-C architecture provides relatively good routing results compared to types A and B, without increasing the number of switches.

In the end, we chose the type-C architecture with 32 horizontal and 16 vertical tracks. It is expected to offer much higher routability compared to the PROTEUS architecture, which has the type-A architecture minus the top channel connections.

IV. PHYSICAL DESIGN

Physical design aspects were not considered in the above discussions. Increasing the number of switches and wires can in-
crease chip size and the propagation delay of signals. In addition, design verification and chip testing are very important, and time consuming, because our FPGA is a full custom LSI.

A. Area

Since the maximum chip size is limited, it is inevitable that the number of logic blocks will have to be decreased when more wires and switches are added to enhance routability.

Table II shows how the area increases when more wires and switches are implemented in PROTEUS and PROTEUS-Lite chips. The numbers for “Switches,” “Wires,” and “Memories” are the numbers of each item in a BC, which is the basic layout block in each chip. Here, “Memories” includes LUT memories and control memories for switches. “Area” is the ratio of the two chips. The area of PROTEUS-Lite is twice that of PROTEUS.

The area-increase ratio is slightly lower than that of switches and memories, but it is higher than that of wires. “Wires” in the table is the number of wires visible to users in each chip. In an actual layout, additional wires are needed to connect wires to switches or switches to memories. The area taken up by these wires should also be considered in the area penalty.

Based on the above results, we reduced the number of BCs from 512 to 280 under the assumption that the final die size is the same as that of PROTEUS. However, this is not so problematic, because, as shown in detail in Section VI, the number of usable BCs is substantially increased by balancing the wiring resources against the number of BCs.

B. Delay

Another important aspect is the signal propagation delay of wires and switches. While it is true that the signal propagation delay depends on the number of switches that the signal passes through [22], unused switches or switches that are “off” also affect the propagation delay.

Using the wiring structure shown in Fig. 8, we simulated the effect of unused switches on propagation delay. We calculated the propagation delay from \( I \) to \( O \) assuming 1) 16 lines are connected to a wire carrying the input signal and 2) the number of switches connected to these lines varies. Fig. 9 shows the relationship between the propagation delay and the number of unused switches. The propagation delay is normalized for the case where the number of fanouts is 1 with no unused switches. Each line corresponds to a different number of fanouts. If the number of unused switches is 0, the propagation delay increases with the number of fanouts. If the number of unused switches increases, the propagation delay also increases.

To avoid the increases in propagation delay caused by unused switches, we put a buffered switch consisting of two trisate drivers between cascading switch matrixes. This localizes the load and limits the propagation delay increase.

Consider the propagation delay of the switch matrix shown in Fig. 10. There are 16 \( \times 64 \) normal switches, which are divided by buffered switches. Fig. 11 shows the propagation delay for this matrix when the number of buffered switches is varied from 0 to 15. In this example, if the number of buffered switches is less than four, the propagation delay decreases because the buffered switches can effectively drive the signal against the load of the unused switch. On the other hand, when that number exceeds four, the propagation delay increases because the intrinsic delay of the buffered switches becomes dominant. Again, we consider the balance between the number of buffered switches and normal switches, and also the fact that buffered switches have another function; they divide a wire into several segmented wires. This provides more alternative routes. Accordingly, we inserted several buffered switches into the wires, which are located around BCs and IOBs; each such wire has more than 60 normal switches.

C. Design Verification

PROTEUS-Lite is a full custom chip, and it was designed manually. Verification of the design is critical due to the complicated switch patterns. Moreover, it is hard to provide correct specifications for layout design. To overcome these difficulties, we developed a design verification environment for generating the correct specifications. We use the architecture definition format (ADF) in the FACT system to describe the logical structure of the switches, wires and their connections as well as the structure of the logic blocks. In the evaluation phase, the
ADF is verified by examining many example circuits, in order to obtain the correct specifications of the PROTEUS-Lite chip.

Our design verification flow is shown in Fig. 12. Netlist information on PROTEUS-Lite is generated from the ADF as well as the library in which LUTs, FFs, and switches are defined. The ADF holds no information on the configuration of memory structures, so these structures are defined in the memory definition file (MDF). The MDF stores the relationship between each switch and its memory addresses. PROTEUS-Lite is verified by comparing the netlist extracted from the layout patterns to the netlists generated from the ADF, MDF, and library.

D. Semi-Automatic Test Pattern Generation

Another important aspect of designing FPGAs is the testing of memories, logic blocks, wires, and switches. The configuration memories themselves can be tested by the standard memory testing method, and there are several techniques for testing the logic blocks. However, it is difficult to develop a general test method for the wires and switches because the test method largely depends on their structure. Our approach is to perform semi-automatic testing using the information in the ADF.

Our test method is explained here using the 3 x 3 switch matrices of Fig. 13. The fault patterns of the wires and switches are categorized into “stuck-at faults” in switches, and “bridge/open faults” in wires.

These faults are tested by feeding the patterns that satisfy the below conditions.

**Condition 1:** Any input of the switch matrix is fed by patterns 0 and 1.

**Condition 2:** Any input of the switch matrix is fed by patterns in which there are signal transitions $1 \rightarrow 0$ and $0 \rightarrow 1$.

**Condition 3:** Each input signal pair $(a, b)$ is fed by two patterns $(a, b) = \{(1, 0), (0, 1)\}$.

The above patterns can be obtained by generating m-out-of-n code, where m is the number of inputs of a switch matrix and n is equal to $m/2$ (or $m/2$ if you want). The patterns in which $m/2$ 1s and 0s are continuous are not used because they do not include both signal transitions ($0 \rightarrow 1$, $1 \rightarrow 0$).

To test switches and wires, specific circuits in which test patterns are fed to the switch/wire under test should be configured in the FPGA. Each switch should be set “on” and “off.” For efficiency of testing, the circuit should cover as many switches as possible, but only one of the switches in horizontal and vertical wires can be tested at a time. Thus, we have to find the patterns that satisfy these requirements. For the case in Fig. 13, four circuits are required to test the nine switches/wires in this switch matrix.

Our developed tool generates the circuits for testing switches and wires from both the ADF and a typical switch matrix pattern library in which templates of position-independent switch patterns are defined. This tool finds the specific switch patterns that match one of the templates in the library and generates the circuits for testing switches in the target switch matrix. A total of 25130 circuits were generated to test all the switches of PROTEUS-Lite. A chip is tested by repeating the cycle of configuring the circuit and feeding the test pattern.

The number of circuits, i.e., test patterns is relatively large. However, the configuration memory in PROTEUS-Lite is partially reconfigurable, and it can be randomly accessed bit by bit. Thus, each circuit is configured by writing only the bits of the configuration memory that differ from those of the previous circuit. This reduces testing time.

V. PLCAD

A. System Overview

A dedicated CAD system called PLCAD was developed for PROTEUS-Lite. An overview of the system is shown in Fig. 14. PLCAD supports the top-down design methodology. Using either the Synopsys’ Design Compiler™ or the in-house logic synthesis system, PARTHENON [23], the programming data downloaded into PROTEUS-Lite can be obtained directly from RTL descriptions.

In addition, a visual design editor was developed to allow the user to interfere freely at any design level. Using the design editor, any part of a designed circuit can be saved at any time, and can be re-used as a hardmacro in other design phases. An actual programming data stream is created after design rule checking (DRC). The static delay calculator is based on a modified Elmore model and can estimate critical path delays to an accuracy of within 4% on average [24]. Furthermore, we created a netlist converter, which produces Verilog-HDL descriptions from our original netlist format. It can also generate delay information in the OVI SDF (Standard Delay File) format if necessary. Thus, we can link to other CAD tools and use the application circuit in
the ASIC design process. Some of the tools in PLCAD system are described in detail below.

B. Mapping

Mapping is the process that assigns an input netlist to logic and routing resources in a PROTEUS-Lite chip. IOBs and latches are assigned one by one. Remaining combinational portions of the input circuit must be covered with LUTs and 5-AND gates. To achieve this, a heuristic algorithm was adopted. The algorithm first creates a Boolean network from the input netlist. Next, the following two steps are executed alternatively until all logic gates are covered with LUTs or 5-AND gates: 1) graph transformation, including inverter elimination by adding a negative-edge attribute to the appropriate edge, along with logic duplication, if necessary; and 2) actual covering applied from the outputs using LUTs and 5-AND gates.

C. Placement

Our algorithm optimizes the placement of each LUT, considering the routing process. First, 42 \times 30 slots, which correspond to possible LUT and IOB locations are assumed, and the placement problem is formulated as a slot allocation problem. A simulated annealing method is used in the algorithm. The evaluation function considers the destination between current LUT location and the next location, usability of local lines, the number of feedback nets, etc. If we use hardmacros, their locations are initially determined by topological sorting and some heuristics.

D. Routing

Similar to other commercial FPGAs, routing is critical when implementing a circuit in PROTEUS-Lite.

Fig. 15 shows an overview of the plroute routing process for our FPGA. It makes the best of the unique structure of PROTEUS-Lite, i.e., hierarchical routing resources and switch patterns in the switch boxes.

The plroute process involves four routing procedures as follows.
I/O Routing Procedure (IRP): The connections between IOB ports and BCs are processed. This procedure tries to find the shortest path using all available resources. This is done first to give I/O connections the highest priority. This procedure utilizes a Dijkstra-based algorithm.

Local-line Routing Procedure (LRP): The connections between adjacent LUTs are routed using a greedy algorithm. This procedure only uses local-line resources, which provide the shortest connections with the smallest propagation delay.

Middle-line and Long-line Routing Procedure (MLRP): The connections between LUTs not routed by LRP are processed. This procedure uses middle-lines and long-lines as the routing resource. It is based on the line search algorithm. However, candidate wire selection is extremely fast, because it is done by simply referring to a switch pattern table, as described in Section II-D3.

Gleaner Routing Procedure (GRP): The connections that cannot be routed by the above procedures are processed. This procedure finds paths to realize complete routing using all of usable routing resources, including unused LUTs.

Our routing process adopts the greedy strategy. IRP is executed first, followed successively by LRP, MLRP, and GRP.

E. Design Editor

It is effective to use hardmacros when programming a large circuit in a PROTEUS-Lite chip. Here, hardmacros are the function parts in which BC locations are mutually fixed. The routing of nets connecting the BCs has already been finished. For example, function parts, such as counters and pattern matching circuits, which are often used in telecommunication circuits, are suitable for hardmacros. An X-Window based design editor was developed to handle them. In the editor, all resources, i.e., BCs, IOBs, and routing resources, are visible. The user can set or program the resources using a mouse. One of the most unique features is that any BC and routed net can be grouped as in a conventional drawing tool, and the grouped part can be handled as one primitive such as a BC. A snapshot of the design editor is shown in Fig. 16.

F. Hierarchical Design Flattening

Logic level flattening can be done using the logic synthesis system. However, if there are hierarchically designed circuits using hardmacros, it is necessary to flatten the circuit hierarchy with partially routed network information. Thus, we also provide a flattening tool that can handle the hardmacros.

VI. EXPERIMENTAL RESULTS

Using the PLCAD system, some telecommunication circuits were evaluated. The results are shown in Table III. The results
for the same circuits implemented using PROTEUS are also shown for comparison. In the table, \( \# \text{LUTs} \) and \( \# \text{Nets} \) indicate the number of used LUTs and the number of nets that should be routed, respectively. \( \text{Time} \) represents the total execution time for technology mapping, placement, and routing in seconds, and \( \# \text{UN} \) is the number of un routed nets. In PROTEUS-Lite, 100\% routing is performed for every circuit, while the execution time is comparable to that of PROTEUS. This indicates that our architecture is enhanced from the routability point of view, or creating a CAD-oriented FPGA.

Fig. 17 compares the routability of PROTEUS and PROTEUS-Lite. The \( x \)-axis represents the number of used LUTs compared to that of LUTs implemented in a chip, and the \( y \)-axis represents the routability ratio between routed nets and all nets for each circuit.

From the results in Fig. 17, nets of circuits that use up to 80\% of the LUTs are almost completely routed by the dedicated CAD system. In PROTEUS, nets in circuits which use up to 20\% of the LUTs are routed by its dedicated CAD system. Consider these results from the view of wasted LUTs. There are 2048 LUTs in PROTEUS and 1120 LUTs in PROTEUS-Lite. Let’s calculate the number of realistically usable LUTs according to the above results. PROTEUS’s realistic LUT usability is \( 2048 \times 20\% = 410 \) while that of PROTEUS-Lite is \( 1120 \times 80\% = 896 \).

PROTEUS-Lite offers nearly twice as many as usable LUTs than PROTEUS, while the total number of LUTs implemented in PROTEUS-Lite is half that in PROTEUS. A chip with fewer LUTs can realize larger circuits. This indicates that it is very important to consider the relationship between the architecture and the CAD system because flexibility is somewhat limited due to the number of wires, LUTs, and IOBs compared to the flexibility of traditional gate arrays. The optimal design never emerges without co-evaluating the FPGA and its CAD system. Our results suggest that if the number of LUTs increases, the number of wires should also increase to maintain coherence between the number of LUTs and wires and switches. It is not enough to evaluate these things statistically: Circuits analogous to those intended to be implemented should be used in the evaluation. Our design methodology is superior in the sense that all aspects of FPGAs are considered, and we have shown one globally optimal point in our own application field.

We also compared PROTEUS-Lite to a commercially-available Xilinx FPGA. The gate capacity of PROTEUS-Lite is about 19.8K gates using Xilinx’s calculation method where a 3-input LUT has 3.2 gates, a 4-input LUT 7 gates and an FF (or latch) 10 gates. XC4010E, which has 18.1K gates, is similar to PROTEUS-Lite according to this calculation method. Table IV shows the critical path delay of several telecom circuits implemented in PROTEUS-Lite and XC4010E-5. Here, the number of FFs in XC4010E-5 is the same as that in PROTEUS-Lite, so the corresponding column is omitted in the table. These results were obtained using their specific CAD tools.

As shown in Table IV, the number of used LUTs in PROTEUS-Lite is about 1.5 times than that in XC4010E-5. However, given that a 4-input LUT has more than twice the logic capability of a 3-input LUT, this result indicates that the technology mapping of PROTEUS-Lite was more efficient. In other words, this fact proves that our logic block, i.e., BC, design is, according to the telecommunication circuit analysis described in [3] and [4], suitable for realizing telecommunication circuits.

In terms of performance, PROTEUS-Lite circuits are faster or equal to those in XC4010E-5. The delay calculated by our method is accurate compared to the actual delay [24] while Xilinx’s delay calculation tool (xdelay) sometimes estimates critical path delay to be shorter than the actual delay. Furthermore, as mentioned in Section II-D-3, our FPGA has a clever switch pattern in each switch box, and the key to shorten the critical path delay is how the characteristics of the switch pat-
tern are utilized effectively. To do so, the placement process as well as the routing is very important. Unfortunately, our current placement tool does not consider the characteristics directly, so there is still room to improve circuit performance by enhancing the placement algorithm.

VII. APPLICATIONS

We are in the process of developing a variety of telecommunication equipment containing PROTEUS-Lite FPGAs. Two pieces that have been completed are a programmable ATM adapter and a transmutable telecom system.

A. Programmable ATM Adapter

As shown in Fig. 18, the programmable ATM adapter mainly consists of 155-Mbps (OC-3) x3 I/O interface cards, a CPU and PROTEUS-Lite FPGAs [25]. A photograph is shown in Fig. 19. Pre/post ATM processing is performed by the I/O interface cards. Thus, PROTEUS-Lite can handle ATM cells directly. A typical application of this ATM adapter is as a traffic shaper, which is necessary when we connect two data traffic lines having different data throughputs. The shaping rate is easily and rapidly changed by reconfiguring PROTEUS-Lite in the programmable ATM adapter appropriately. A one-to-two way line shaper requires about 2K gates, and it is easily implemented in a PROTEUS-Lite. The FPGA reconfiguration and other parameter settings are performed by the CPU, and the data can be saved in the flash memory in the ATM adapter.

B. Transmutable Telecom System

We have developed a reconfigurable or transmutable system for telecommunications, called ATTRACTOR, which features both flexibility and high performance [26]. There are two key innovations. One is a board-level modularity concept which allows different functions to be implemented on different boards. The other is a high-speed serial link mechanism that provides excellent interboard communications without sacrificing performance. Our system is distinguished from conventional ASIC emulators by its ability to provide a real-world execution environment, which enables us to connect the system to other telecommunications systems directly.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>PROTEUS-Lite</th>
<th>XC4010E-5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of FPs</td>
<td># of LUTs</td>
</tr>
<tr>
<td>fcount</td>
<td>13</td>
<td>45</td>
</tr>
<tr>
<td>rts,mng</td>
<td>17</td>
<td>46</td>
</tr>
<tr>
<td>v_atgp</td>
<td>15</td>
<td>49</td>
</tr>
<tr>
<td>svtg</td>
<td>13</td>
<td>50</td>
</tr>
<tr>
<td>sc,mng</td>
<td>11</td>
<td>70</td>
</tr>
<tr>
<td>augen</td>
<td>21</td>
<td>75</td>
</tr>
<tr>
<td>hec</td>
<td>48</td>
<td>85</td>
</tr>
<tr>
<td>vclgen</td>
<td>13</td>
<td>89</td>
</tr>
<tr>
<td>cellgen</td>
<td>54</td>
<td>136</td>
</tr>
<tr>
<td>aal1,t</td>
<td>59</td>
<td>214</td>
</tr>
<tr>
<td>stn1gen</td>
<td>93</td>
<td>228</td>
</tr>
<tr>
<td>Average</td>
<td>33</td>
<td>99</td>
</tr>
</tbody>
</table>
Fig. 20. Overview of ATTRACTOR.

Fig. 21. Photograph of ATTRACTOR.

on-the-fly. We connected an ATTRACTOR system transmuted as the IP cut-through router to a 155-Mbps ATM-LAN (local area network) environment, and confirmed that Motion-JPEG data streams could be transferred with no frame loss and no delay at full-rate traffic.

VIII. CONCLUSION

We have introduced our project dedicated to developing a telecommunication-based FPGA that features both ease of use and high performance. To realize this FPGA, we carefully considered how to best use a CAD system to handle the FPGA in the architecture design phase, and evaluated the balance between logic elements and routing resources using a newly developed evaluation system. In addition, we created an integrated FPGA design environment in order to accelerate the development and validation of the FPGA, i.e., a full custom LSI design. As a result, we obtained a well-balanced FPGA and a dedicated CAD system. The project is now shifting to developing application systems. One future task is to construct a system-level hardware-software codesign environment in order to be able to handle complicated systems utilizing MPUs as well as our FPGA.

ACKNOWLEDGMENT

The authors would like to thank K. Ishii and H. Tsuboi for providing some of the experimental results. They also wish to thank K. Matsuhiro, K. Hayashi, and N. Ohta for their helpful suggestions.

REFERENCES

Toshiki Miyazaki (M’88) received the B.E. and M.E. degrees in electronic engineering from the University of Electro-Communications, Tokyo, Japan, in 1981 and 1983, respectively, and the Dr.Eng. degree from the Tokyo Institute of Technology in 1994.

He is a Senior Research Engineer, Supervisor of Nippon Telegraph and Telephone Corporation (NTT) Network Innovation Laboratories, Atsugi, Japan. Since joining NTT in 1983, he has been engaged in research on VLSI CAD systems including high-level synthesis, logic design tools, and CAD frameworks.

His research interests are in high-level synthesis, hardware/software codesign, FPGA-related systems, and evolvable hardware.

Dr. Miyazaki is a member of the IEICE and IPSJ.

Masaru Katayama received the B.E. and M.E. degrees from Hokkaido University, Japan, in 1990 and 1992, respectively.

Currently, he is a Research Engineer at Nippon Telegraph and Telephone Corporation (NTT) Electrical Communication Laboratories, Atsugi, Japan. He has been engaged in research on molecular beam epitaxy technology for Si-HBTs, functional design methodology for telecommunication LSIs, and multi-FPGA systems. Currently, his research interests include new service planning and promotion over next-generation broadband networks.

Kazuhiro Shirakawa (M’95) received the B.E. and M.E. degrees in electrical engineering from Osaka University, Osaka, Japan, in 1985 and 1987, respectively.

He joined Nippon Telegraph and Telephone Corporation (NTT) Asahi Electrical Communication Laboratories, Atsugi, Japan, in 1987. He is currently a Manager of the multimedia business department. His present interest is programmable digital transport systems and their design methodology.

Takaki Ichimori received the B.S. and M.S. degrees from Hokkaido University, Japan, in 1984 and 1986, respectively.

He joined the Nippon Telegraph and Telephone Corporation (NTT) Network Innovation Laboratories, Kanagawa, Japan in 1986. He has been engaged in research on molecular beam epitaxy technology for Si-HBTs, functional design methodology for telecommunication LSIs, and multi-FPGA systems. Currently, his research interests include new service planning and promotion over next-generation broadband networks.

Dr. Takahiro Murooka is a member of the IEICE and ACM.

Atsushi Takahara (M’95) received the B.E., M.E. and the Dr.Eng. degrees from Tokyo Institute of Technology, Tokyo, Japan, in 1983, 1985, and 1988, respectively.

He joined Nippon Telegraph and Telephone Corporation (NTT) LSI Laboratories in 1988. He is currently a Senior Research Engineer, Supervisor of NTT Network Innovation Laboratories, Kanagawa, Japan. His current research interests are in formal design verification and formal methods of system design.

Dr. Takahara is a member of the IEICE, ACM, and IPSJ.
Akihiro Tsutsui (M’95) received the B.E. and M.E. degrees in systems engineering from Kobe University, Hyogo, Japan, in 1988 and 1990, respectively. He is a Research Engineer at Nippon Telegraph and Telephone Corporation (NTT) Network Innovation Laboratories, Kanagawa, Japan. Since joining NTT in 1990, he has been engaged in research on transport data processing. He has been involved in the project for developing telecommunications-oriented FPGAs and their dedicated CAD systems. His research interests are in hardware architecture for high-speed network devices, FPGA-related systems, and their design environment including hardware/software codesign approach. Mr. Tsutsui is a member of the IEICE and ACM.

Kennosuke Fukami (M’85) received the B.S. and M.S. degrees from Keio University, Japan, in 1977 and 1979, respectively. He is an Executive Manager, research planning department, Nippon Telegraph and Telephone Corporation (NTT) Lifestyle and Environmental Technology Laboratories, Tokyo, Japan. He has been engaged in research on system LSI design and its design methodologies. He has accumulated considerable experience in development of optical transmission system and switching system LSIs since joining NTT Laboratory in 1979. His recent effort has been focused on the strategy of the new lifestyle and environmental technology research and development. Mr. Fukami is a member of the IEICE.