Vol. 40 No. 4 Transactions of Information Processing Society of Japan Apr. 1999

Regular Paper

Datapath-Layout-Driven Design for Low-Power FPGA Implementation

HIROYUKI OCHI,† TAKASHI NISHIKAWA† and TAKAO TSUDA†

The datapath is one of the main power-consuming parts in today's VLSIs. This paper proposes a new top-down field-programmable gate array (FPGA) design flow called *Datapath-Layout-Driven Design* (*DLDD*), which achieves low power by reducing the datapath wire length. While module partitioning and floor planning in the conventional design flow are based on the functional hierarchy, those in the DLDD flow are based on the "bit-slice" manner. After the module partitioning, the floor plan is designed and the optimal aspect ratio of the submodule is determined, followed by logic and layout synthesis. As an effect of DLDD flow, we can expect (1) *low power*, because most data nets are short intra-module wires rather than long inter-module data buses, and (2) *efficient layout*, because all submodules are the same in size and shape. As a case study, DLDD was applied to the design of a 16-bit RISC processor, ASAP-1.1, for implementation on an FPGA device, Xilinx XC5210-6 PG223. We implemented and compared seven designs based on DLDD flow with different aspect ratios of submodules, along with one conventional design. In the best case in the experiments, the power consumption was successfully reduced by 25%.

1. Introduction

In recent years, with the growing popularity of portable electronic equipment, it has become more important for VLSI design to achieve low power. Low-power design is also becoming important in the area of high-performance VLSIs, because overheating caused by high-speed and high-density degrades performance and reduces chip life-time. Various studies have been made of how to reduce power dissipation at every stage in the design flow, namely, the architecture[1], logic[2], circuit[3], and physical design[4].

Layout design has acquired more importance with recent advances in semiconductor technology: the size of features has decreased while the chip size has grown, and as a result, wire length has become a more significant factor in optimizing area, delay, and power. As systems implemented on a single chip become more sophisticated, it is becoming more difficult to generate an optimal physical layout by the conventional top-down design flow based on functional hierarchy.

In this paper, *Datapath-Layout-Driven Design* (*DLDD*) is proposed as a new top-down design flow that enables field-programmable gate arrays (FPGAs) to achieve low power by reducing the datapath wire length. In the DLDD flow, module partitioning is based on the "bit-slice" manner. After the module partitioning, the floor plan is designed and the optimal aspect ratio of the submodule is determined, followed by logic and layout synthesis. This top-down design flow makes it possible to generate a final layout with an optimized datapath wire length.

In the next section, we introduce DLDD and describe its general concepts. In Section 3, as a case study, DLDD is applied to the design of a 16-bit RISC processor, ASAP-1.1, for implementation on an FPGA device. In experiments, the power consumption was 25% lower in the best case than with the conventional design flow. Section 4 provides a summary, and outlines the direction of future work.

2. Datapath-Layout-Driven Design

2.1 A Power Dissipation Model

Let us denote by E_{gate} the amount of energy dissipated by a CMOS logic gate each time its output changes. E_{gate} is roughly given by:

$$E_{gate} = 0.5 C_l V_{DD}^2,$$

where C_l is the load capacitance of the gate and V_{DD} is the supply voltage. If V_{DD} is fixed, E_{gate} is affected only by C_l. C_l is given by:

$$C_l = C_d + C_w + C_g,$$

where C_d, C_w, and C_g are the capacitances of the drain (diffusion), wire, and gates, respectively. As the feature size is reduced, C_w becomes a relatively larger part of C_l. C_w is given by:

$$C_w = c_a l_{wire} w_{wire} + 2c_f(l_{wire} + w_{wire}),$$

† Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University

where c_a and c_f are the capacitances for unit area and the unit fringe length, respectively, of the metal, and l_{wire} and w_{wire} are the length and width of the wire. c_a, c_f, and w_{wire}☆ depend on the choice of metal layers.

From the above discussion, it is clear that reduction of the wire length l_{wire} is one of most important issues in low-power design.

2.2 Conventional Design Flow

A typical design flow of digital systems such as microprocessors is currently as follows:

(1) Module partitioning based on the functional hierarchy (e.g., register file, ALU, etc.)
(2) Logic synthesis and technology mapping for every module
(3) Layout design (floor planning for the chip level and layout design for submodules, followed by layout design for the chip level)

The final layout generated by the above design flow is something like **Fig. 1** (a). This design flow has the following problems:

- For example, if we design a 32-bit processor, many 32-bit data buses are needed to connect submodules. Hence the total wire length of data buses is very large, and buses can be among the heaviest power-consuming parts of a processor.
- Each module has a different physical size and shape, which makes it very difficult to optimize the floor planning.

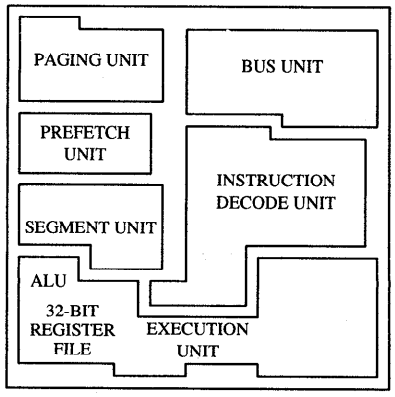
2.3 Proposed Design Flow

As a breakthrough solution to the problems discussed so far, we will propose *datapath-layout-driven design* (*DLDD*) as a new top-down design flow. The goal of DLDD is to minimize the total wire length of data buses. For this purpose, a new module partitioning method is introduced to allow floor planning at a very early stage in the design flow.

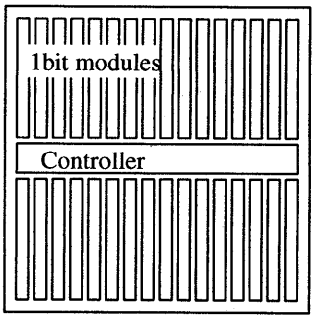
The DLDD flow is as follows:

(1) Module partitioning of the datapath in a "bit-slice" manner
(2) Floor planning to determine the optimal aspect ratio of the submodule
(3) Logic synthesis and layout design for the submodule to fit into a rectangular area with the expected aspect ratio
(4) Placement of submodules and global routing

☆ In usual digital design, the minimum width allowed by the design rules for the metal layer is applied for all wires except power and clock nets.



(a) Conventional design[5)]



(b) Datapath-Layout-Driven Design

Fig. 1 Typical layout of a 32-bit CPU.

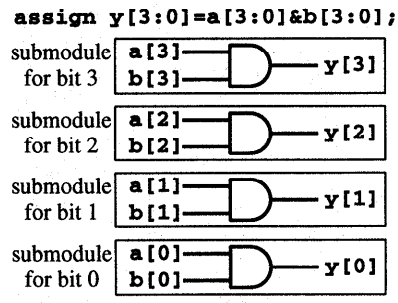
An example of the final layout generated by the proposed design flow is shown in Fig. 1 (b).

"Bit-slice" module partitioning (step (1)) is applied to datapath operations that satisfy the following conditions:

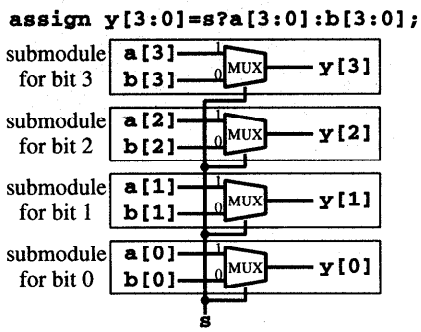
- Most inputs and outputs are bus with fixed word size.
- Each bit can be implemented by the same circuit.
- The number of wires between the sliced circuits is relatively small.

Figure 2 shows examples that meet the above criteria. Complex datapath components (e.g., floating-point units) are not suited to "bit-slice" partitioning. We assume that such components are excluded from the submodule and included in the controller.

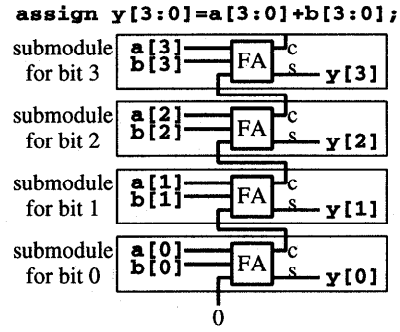
As an example, let us consider how to implement the circuit shown in **Fig. 3** (a). If we apply the conventional and the proposed methods of module partitioning, the numbers of inter-module wires are 17 (Fig. 3 (b)) and 6 (Fig. 3 (c)), respectively, where *inter-module wires* are ones that connect two or more sub-



(a) Bitwise logic operation

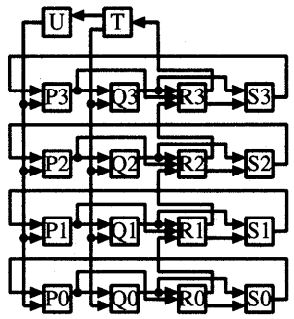


(b) Multiplexer

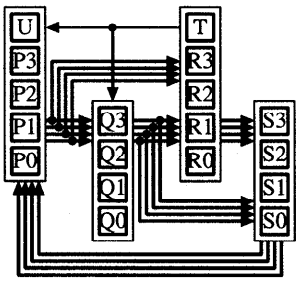


(c) Adder/Subtracter/Comparator

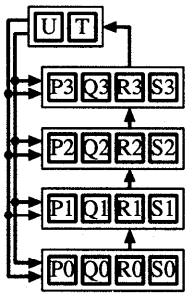
Fig. 2 Module partitioning in DLDD flow.



(a) Circuit to be implemented
(total no. of wires = 23)



(b) Conventional module partitioning
(no. of inter-module wires = 17)



(c) DLDD module partitioning
(no. of inter-module wires = 6)

Fig. 3 Module partitioning and inter-module wires.

modules. We call wires that are included in a submodule *intra-module wires*. This shows that the proposed module partitioning can reduce the number of inter-module wires.

If the number of inter-module wires is reduced, we can expect that the total wire length will be reduced, because:

(1) If the optimization effort for placement is very low, inter-module wires are usually longer than intra-module wires.

(2) Optimizing the placement so that inter-module wires are as short as intra-module wires takes a great effort.

(1) is illustrated in **Fig. 4**. As a model of low-effort placement, let us assume that the source and destination of each wire are "randomly" placed. As shown in Fig. 4, the expected lengths of intra-module and inter-module wires are $0.5(h+w)$ and $0.5(H+W)$, respectively, where h and w (H and W) are respectively the height and width of a submodule (chip). It follows that inter-module wires are much longer than intra-module wires, if $H \gg h$ and $W \gg w$.

Optimization of placement is possible, and inter-module wires may be as short as intra-module wires. However, to derive an optimal global layout, we have to refine the layouts of

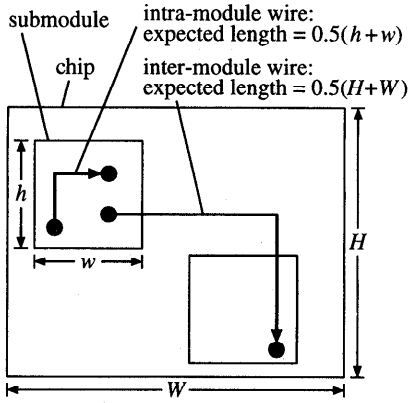


Fig. 4 Expected wire length if optimization effort is very low.

submodules to adapt them to the global layout. Such optimization requires iteration of submodules' layout and global layout in the design flow. This is the reason for (2).

The merits of DLDD can be summarized as follows:

- The number of inter-module data buses of one-word width has fallen considerably, because most data nets have become intra-module wires. This makes it possible to reduce the total wire length of data buses without a great effort.
- The size and shape of the submodule for each bit is the same. This makes it easy to solve the placement problem.

To make DLDD more practical, we have to answer the following questions:

- How can we reduce the wire length of inter-module critical paths (e.g., carry signal for addition)?
- How can we reduce the wire length of signals from the controller to each submodule?
- What is the best aspect ratio (width/height) for a submodule?

In the next section, the results of a case study will be given in answer to these questions.

3. Implementation and Evaluation

3.1 16-bit RISC Processor ASAP-1.1

A 16-bit RISC processor, ASAP-1.1, was designed and implemented on an FPGA device. ASAP-1.1 is upwardly compatible with ASAP-0[6], which is a DLX[7]-like educational 16-bit RISC processor. ASAP-1.1 has eight 16-bit general-purpose registers (`gr`), a 16-bit program counter (`pc`), and zero and negative flags (`z` and

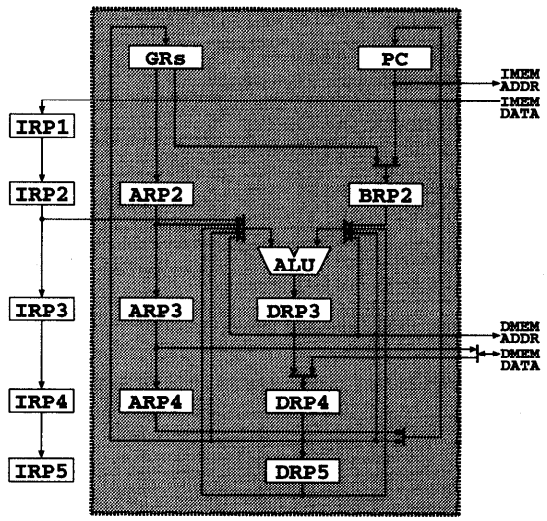


Fig. 5 Datapath diagram of ASAP-1.1.

n, respectively), and has 13 essential instructions. A diagram of the datapath and an instruction set summary are shown in **Fig. 5** and **Table 1**, respectively. ASAP-1.1 is a 5-stage pipelined processor with bypass paths, which are needed for forwarding.

3.2 Implementation

ASAP-1.1 was described in Verilog-HDL, then compiled into schematics by Cadence Synergy 2.2, and implemented into a target FPGA device by Xilinx Xact 5.2.0 on a Sun SPARCstation 5 workstation with 64 MB of memory. The target FPGA device is XC5210-6 PG223, a 10,000-gate-class look-up-table-based FPGA device. It has an 18 × 18 array of configurable logic blocks (CLBs), each of which has four flipflops and four look-up-tables[8].

For comparison, conventional design flow was performed as well as DLDD flow.

3.2.1 Conventional Design Flow

For the conventional design flow, ASAP-1.1 was described in Verilog-HDL with 5 modules: `top`, `phase`, `pc`, `regfile`, and `alu`. `pc` implements the program counter, `regfile` implements the register file, and `alu` implements the ALU. These modules correspond to `PC`, `GRs`, and `ALU` in Fig. 5, respectively. `phase` is a sequential machine with 6 flipflops. `top` is the highest module, and implements the other components. The total length of the description is 259 lines. **Figure 6** (a) shows inter-module connections of the description. In this figure, there are 144 and 24 nets for inter-module data and control signals, respectively.

The description is processed by Synergy 2.2

Table 1 Instruction set summary of ASAP-1.1.

Mnemonic	(MSB) Instruction Code (LSB)	Verilog-HDL Pseudocode

LD Ra, d(Rb)	00 | Ra | Rb | d	// LoaD from memory
		gr[Ra] <= mem[gr[Rb]+d];
		pc <= pc+1;

ST Ra, d(Rb)	01 | Ra | Rb | d	// STore to memory
		mem[gr[Rb]+d] <= gr[Ra];
		pc <= pc+1;

LI Rb, d	10 | 001 | Rb | d	// Load Immediate
		gr[Rb] <= d; // (Sign Ext.)
		pc <= pc+1;

LHI Rb, d	10 | 011 | Rb | d	// Load High Immediate
		gr[Rb] <= d<<8 | gr[Rb]&255;
		pc <= pc+1;

B d	10 | 111 | 000 | d	// Branch pc <= pc+1+d;
BNE d	10 | 111 | 100 | d	// Branch Not Equal pc <= (!z) ? pc+1+d : pc+1;
BN d	10 | 111 | 110 | d	// Branch Negative pc <= (n) ? pc+1+d : pc+1;

ADD Rd, Rs	11 | Rs | Rd | 0000 0000	// ADD arithmetic
		gr[Rd] <= gr[Rd]+gr[Rs];
		z <= ~|(gr[Rd]+gr[Rs]);
		n <= (gr[Rd]+gr[Rs])>>15;
		pc <= pc+1;

NOR Rd, Rs	11 | Rs | Rd | 0010 0000	// NOR bitwise
		gr[Rd] <= ~(gr[Rd]|gr[Rs]);
		z <= ~|(~(gr[Rd]|gr[Rs]));
		n <= ~(gr[Rd]|gr[Rs])>>15;
		pc <= pc+1;

SRL Rd	11 | 000 | Rd | 1010 0000	// Shift Right Logically
		gr[Rd] <= gr[Rd]>>1;
		z <= ~|(gr[Rd]>>1);
		n <= 0;
		pc <= pc+1;

JAL Ra, Rb	11 | Ra | Rb | 1111 0101	// Jump And Link
		gr[Rb] <= pc+1;
		pc <= gr[Ra];

NOP	11 | 111 | 111 | 1111 1110	// No OPeration pc <= pc+1;

HLT	11 | 111 | 111 | 1111 1111	// HaLT
		run <= 0;
		pc <= pc+1;

and Xact 5.2.0 without any given floor plan, i.e., layout optimization is performed automatically by Xact 5.2.0.

The obtained layout is shown in **Fig. 7** (a). The elapsed times for Synergy and Xact were 19 min and 57 min, respectively.

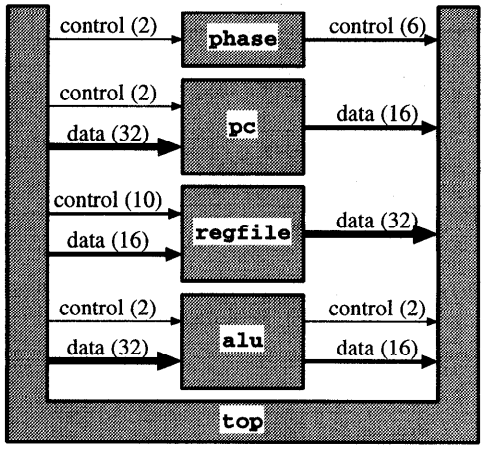
The maximum circuit delay from flipflops to flipflops estimated by Xact is 219.2 nsec, as shown in **Table 2** (a).

3.2.2 Datapath-Layout-Driven Design Flow

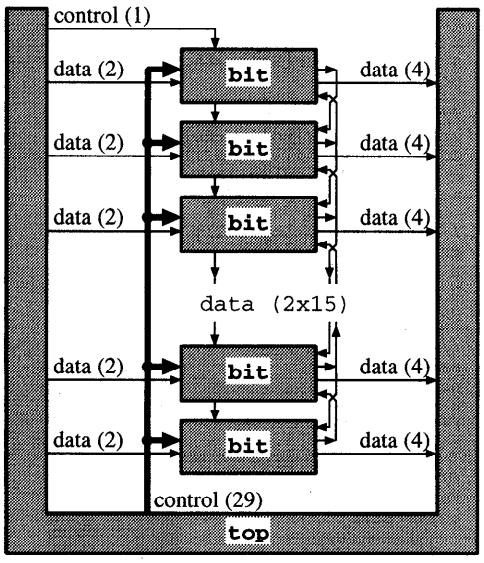
For the DLDD flow, ASAP-1.1 was described in Verilog-HDL with 2 modules, `top` and `bit`. The `bit` module implements one of 16 bit datapaths of ASAP-1.1. The `bit` module is instantiated 16 times in the module `top` to realize the shaded portion of Fig. 5, while the other portions (including miscellaneous control logics) are included in the `top` module. The lengths of the descriptions for `top` and `bit` are 186 and 91 lines, respectively. Figure 6 (b) shows the inter-module connections of the description. In this figure, there are 126 and 30 nets for inter-module data and control signals, respectively. A comparison of Fig. 6 (b) with Fig. 6 (a) shows that we successfully reduced the number of inter-module data nets.

After netlists for `top` and `bit` modules are generated, the design flow for final layout is as follows:

(1) Design a floor plan for the chip level.
(2) Generate a layout for the submodule, `bit`.
(3) Given the results of (1) and (2), generate a final layout.

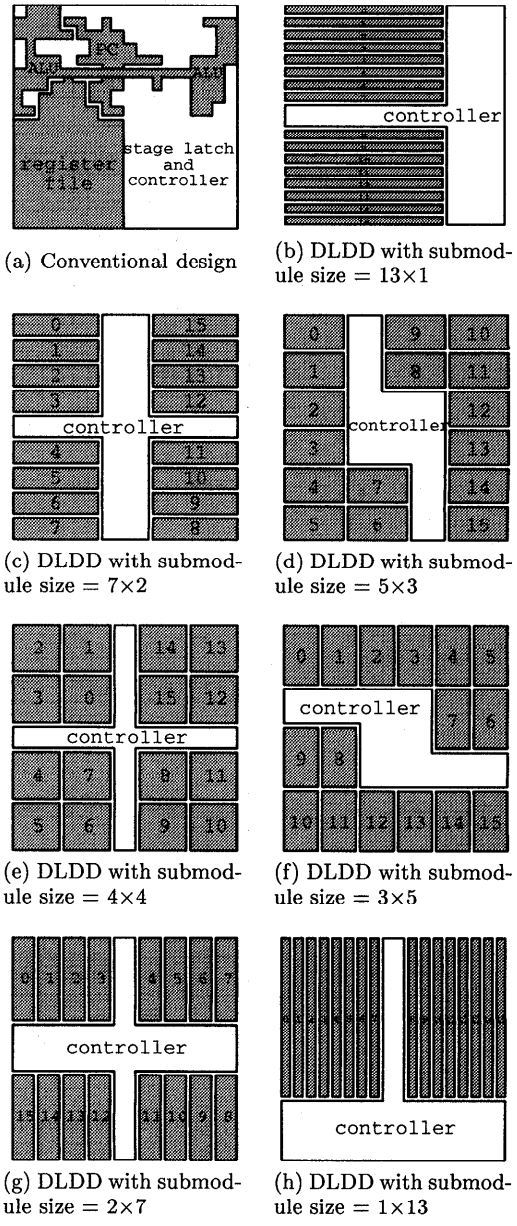


(a) Conventional design
(data: 144 nets, control: 24 nets)



(b) DLDD design (data: 126 nets, control: 30 nets)

Fig. 6 Inter-module connections.



(a) Conventional design
(b) DLDD with submodule size = 13×1
(c) DLDD with submodule size = 7×2
(d) DLDD with submodule size = 5×3
(e) DLDD with submodule size = 4×4
(f) DLDD with submodule size = 3×5
(g) DLDD with submodule size = 2×7
(h) DLDD with submodule size = 1×13

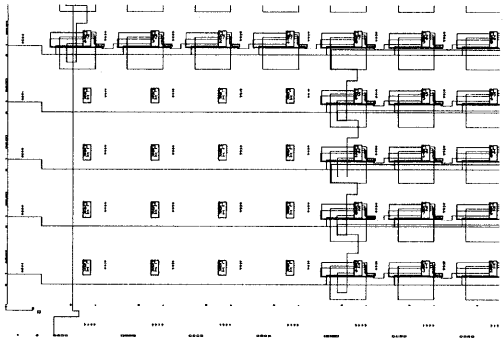
Fig. 7 Implemented layouts.

Assume that the available array of CLBs in the target FPGA device is 18 × 18 and that each submodule requires at least 13 CLBs. Figure 7 (b)–(h) shows possible floor plans for layouts of submodules with width×height of 13×1, 7×2, 5×3, 4×4, 3×5, 2×7, and 1×13. We implemented these seven floor plans for comparison.

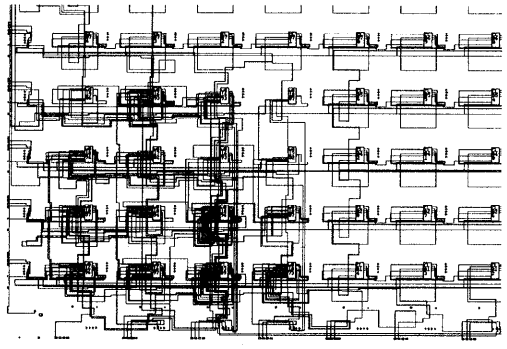
To generate an optimal layout for the submodule, `bit`, it is desirable to use existing automated FPGA layout tools (e.g., Xact) rather than to do full-custom layout manually. To do it properly, we developed a method using dummy components. Before applying a netlist of the submodule, `bit`, to Xact, we added dummy components whose locations are specified so that all LUTs and FFs except the desired area are occupied by the dummy components. **Figure 8** (a) shows a layout of dummy components. In this case, all LUTs and FFs except 4×4 CLBs in the left-bottom corner are occupied by the dummy components. If we append these dummy components to the netlist for the submodule, `bit`, the layout shown in Fig. 8 (b) is generated automatically by Xact, whose left-

Table 2 Performance (circuit delay from FF to FF in nsec).

(a) Conventional Design	Datapath-Layout-Driven Design						
	(b) 13×1	(c) 7×2	(d) 5×3	(e) 4×4	(f) 3×5	(g) 2×7	(h) 1×13
219.2	288.9	323.6	322.0	317.3	326.4	320.1	291.3



(a) Layout of dummy components



(b) Generated layout

Fig. 8 Generating a layout for a submodule.

bottom corner represents an optimal layout for the submodule, `bit`, with area 4×4. From this layout, relative placement information for components in the submodule, `bit`, is automatically extracted for the final layout, using `awk` and `sed` scripts.

The elapsed time for Synergy is 8 min, while the elapsed times for Xact for the submodule with size = 1×13 and for Xact for the final layout are 10 min and 47 min, respectively.

The maximum circuit delay from flipflops to flipflops estimated by Xact is shown in Table 2 (b)–(h).

3.3 Experimental Results

We developed an FPGA board for power measurement (**Fig. 9**). It consists of an FPGA device (Xilinx XC5210-6 PG223), two 32-KB SRAM devices for instruction memory, two 32-KB SRAM devices for data memory, a 3-MHz clock oscillator, RESET and START switches

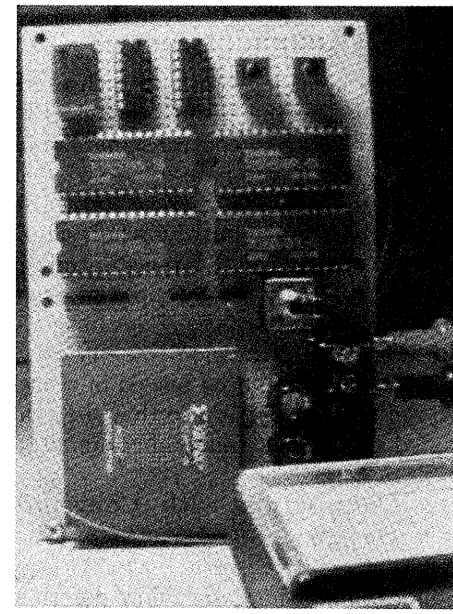


Fig. 9 FPGA board for power measurement.

with RS-latches, a power switch, and a connector for the XChecker download cable. Separate power supplies were provided, one for the FPGA device and the other for the other devices, to measure the VDD current of the FPGA device accurately.

Table 3 shows the measured VDD current. We measured the VDD currents in the following four modes:

- `clk`: A 3-MHz clock is supplied to the halting CPU.
- `nop`: The CPU is executing a program with a long sequence of `NOP` instructions.
- `circle`: The CPU is executing a typical application program.
- `toggle`: The CPU is executing a program that toggles almost all datapaths in every clock cycle.

3.4 Discussion

3.4.1 Aspect Ratio of Submodules

Comparing the DLDD for different submodule sizes, we can see that the design with submodule size = 1×13 achieved the best results in both power consumption (Table 3) and clock cycle time (Table 2).

We can guess that the difference in power

Table 3 Power (VDD current in mA/3.0 MHz).

	(a) Conventional Design	Datapath-Layout-Driven Design						
		(b) 13×1	(c) 7×2	(d) 5×3	(e) 4×4	(f) 3×5	(g) 2×7	(h)1×13
`clk`	15.3	15.5	17.0	16.4	16.0	17.6	16.2	14.9
`nop`	22.5	21.7	24.3	23.6	24.2	25.4	24.2	20.8
`circle`	100.2	102.0	107.1	107.4	108.3	112.2	108.5	93.6
`toggle`	82.5	67.1	67.9	70.5	66.2	70.9	69.8	61.9

consumption is caused by the length of global control signals. In the case of the "1×13" design, all fanins of every control signal are in the same row, while in the case of "3×5", "4×4", or "5×3" designs, fanins of every control signal are distributed throughout the whole chip. We can also guess that the difference in minimum clock cycle time is caused by the wire length of the ripple carry signal of the adder in the ALU.

From the above discussion, we can conclude that submodules should be placed in a narrow-shaped rectangular pattern so that they can all be located side by side (one-dimensionally) in a chip.

3.4.2 Comparison with Conventional Design

3.4.2.1 Design Effort

Let us examine the design flow described in Section 3.2.2:

(1) Design a floor plan for the chip level.
(2) Generate a layout for the submodule, `bit`.
(3) Given the results of (1) and (2), generate a final layout.

As described in Section 3.4.1, we have established the best floor plan for the chip level in the DLDD flow; therefore, from now on, Step (1) is performed in a straight-forward manner. As described in Section 3.2.2, Steps (2) and (3) are automated by using existing tools and developed scripts. The times taken by Xact for Steps (2) and (3) are 10 min and 47 min, respectively, while the total time taken by Xact in the conventional flow is 57 min.

From the above observation, we can see that layout design effort required for the DLDD flow is comparable with that for the conventional design flow (Section 3.2.1) without interactive optimization.

3.4.2.2 Power Consumption

In comparison with the conventional design in Table 3, the "1×13" design reduces the power consumption by 25% for the `toggle` mode. This result proves that the datapath is one of the main power-consuming parts, and thus it is possible to reduce the power consumption by adopting the DLDD flow.

3.4.2.3 Speed

From Table 2, the clock cycle time in the "1×13" design is 32.9% larger than in the conventional design. This means that the conventional design is a better choice when we require the best performance possible with the target FPGA, while DLDD may be a better choice if the clock cycle time is given as fixed.

4. Conclusion

In this paper, we have proposed *Datapath-Layout-Driven Design* (*DLDD*) as a new top-down FPGA design flow that enables a physical layout to be generated with power-optimal datapath routing. We applied DLDD to the design of a 16-bit RISC processor, ASAP-1.1, for implementation on an FPGA device with several floor plans. The experimental results suggested that a long, narrow rectangular shape is a good choice for the submodule, rather than a rectangle with a 1:1 aspect ratio. In the best case in the experiments, the power consumption was successfully reduced by 25%.

As explained in Section 3, DLDD increases the wire length of the critical path as a trade-off for reducing the total wire length. This problem may be solved if buffer sizing for the critical path is applied, although buffer sizing is impossible when the target device is FPGA.

Instead of reducing the total wire length of the datapath, it seems that DLDD increases the total wire length of the control signals. To get better results, it is necessary to develop a better interface between the controller and the submodules.

In the experiments described in Section 3, we developed several scripts and programs to enable us to use existing CAD tools in the DLDD flow. It is strongly desirable to develop a better CAD framework and environment to enhance the DLDD flow.

In the experiments described in Section 3, we used a Xilinx XC5200-series FPGA device; however, we believe that similar results can be obtained if target technology provides wiring resources that do not severely restrict the floor

plan. Currently, we are planning to apply DLDD to a cell-based LSI design, to prove its capability.

Acknowledgments The authors thank the reviewers for their constructive comments.

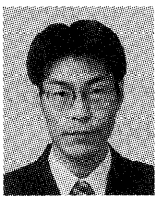
This work is supported in part by a Grant in Aid for Scientific Research from the Ministry of Education, Science, Sports and Culture of Japan (no. 09780299).

References

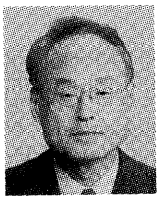
1) Sparsø, J., Nielsen, L.S., Niessen, C. and Berkel, K.: Low-Power Operation Using Self-Timed Circuits and Adaptive Scaling of the Supply Voltage, *IEEE Trans. VLSI System*, Vol.2, No.4, pp.391–397 (1994).
2) Tamiya, Y., Matsunaga, Y. and Fujita, M.: LP-Based Cell Selection with Constraints of Timing, Area, and Power Consumption, *Proc. ICCAD'94*, San Jose, California, IEEE/ACM, pp.378–381 (1994).
3) Konishi, K., Kishimoto, S., Lee, B.-Y., Tanaka, H. and Taki, K.: A Logic Synthesis System for the Pass-Transistor Logic SPL, *Proc. SASIMI'96*, Fukuoka, pp.32–39 (1996).
4) Chao K.-Y. and Wong, D.F.: Low Power Considerations in Floorplan Design, *Proc.1994 Intl. Workshop on Low Power Design*, Napa, California, pp.45–50 (1994).
5) Bakoglu, H.B.: *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Reading, MA (1990).
6) Ochi, H., Kamidoi, Y. and Kawabata, H.: ASAver.1: An FPGA-Based Education Board for Computer Architecture/System Design, *Trans. IEICE*, Vol.E80-A, No.10, pp.1826–1833 (1997).
7) Hennessy, J.L. and Patterson, D.A.: *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Mateo, CA (1990).
8) Xilinx, Inc., Xilinx XC5200 Logic Cell Array Family Technical Data v3.0, Xilinx, San Jose, CA (1995).

(Received September 4, 1998)
(Accepted February 8, 1999)

Hiroyuki Ochi received the B.E., M.E., and Ph.D. degrees in Information Science from Kyoto University, Kyoto Japan, in 1989, 1991, and 1994, respectively. Since 1994, he has been an Associate Professor of Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University, Hiroshima, Japan. His current research interests include low-power design, binary decision diagram (BDD), field-programmable gate array (FPGA), and pass-transistor logic. He is a member of IEICE.



Takashi Nishikawa received the B.E. degree in Information Science from Hiroshima City University, Hiroshima, Japan, in 1998.



Takao Tsuda received the B.E., M.E., and Ph.D. degrees in Electrical Engineering from Kyoto University, Kyoto Japan, in 1957, 1959, and 1964, respectively. Currently, he is an emeritus professor of Kyoto University and a professor of Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University, Hiroshima, Japan. His current research interests include vectorizing/parallelizing compiler, parallel numerical analysis, and Monte Carlo method. He is a member of ACM and SIAM.