IPSJ Transactions on System LSI Design Methodology Vol. 2 53–63 (Feb. 2009)

Regular Paper

Reducing Power Dissipation of Data Communications on LSI with Scheduling Exploration

Kazuhito Ito[†1] and Hidekazu Seto[†1]

Power dissipation by data communications on LSI depends on not only the binding and floorplan of functional units and registers but how data communications are executed. Data communications depend on the binding, and the binding depends on the schedule of operations. Therefore, it is important to obtain the best schedule which leads to the best binding and floorplan to minimize the power dissipated by data communication. In this paper a schedule exploration method is presented to search the best one which achieves the minimized energy dissipation of data communications.

1. Introduction

Data transfer over interconnects between modules such as functional units and registers for data communications is one of the major sources of power dissipation on LSIs [1]. In data transfer, each bit of data is transferred by charging or discharging the corresponding interconnect wire. Therefore the energy dissipation is proportional to the capacitance of the wire and usually it is strongly depends on the length of the wire. It is reported that for a moderate wire length, the energy dissipation is almost proportional to the wire length [1]. The length of wire necessary for a particular data communication can be minimized by using point-to-point interconnection scheme or by using bus architecture with bus splitting [2)–4)]. In order to minimize the energy dissipation of data communication on interconnects with these techniques, the physical locations of the data source and destination have the great importance.

Usually LSI design includes scheduling, binding, and floorplanning. Scheduling is to determine the execution start time for each computational tasks. Binding is to assign computational tasks to functional units (FUs). In addition, data intervals are to be bound to registers. Floorplanning is to determine the location and orientation of the functional units and registers so that these can be placed on a two dimensional LSI chip. Scheduling influences the binding because the concurrently executing tasks cannot be assigned to an identical functional unit. Binding influences the power dissipation of interconnects because the requirements of data communication among FUs and registers are determined by the result of binding.

In Ref. 1), energy dissipation is minimized by intersecting non-data lines in parallel wiring of bus to reduce coupling capacitances. Related to bus splitting, the best clustering of modules is proposed so that each cluster constitutes a split bus segment to minimize energy dissipation by the bus [3]. The work 4) is targeted to minimize data communication delay rather than minimizing energy dissipation. All of these do not consider scheduling and binding in minimization of energy dissipated by interconnects.

In this paper, we propose a method where the scheduling is explored so that the combination of the schedule, binding, and floorplan is optimized to achieve the minimized power dissipation of interconnects.

This paper is organized as follows. The hardware model is defined in Section 2. The motivational example is presented in Section 3. In Section 4, the schedule exploration technique is proposed and the necessary consideration for implementing energy dissipation minimization is described. Experimental results are shown in Section 5. Section 6 concludes this paper.

2. Hardware Model

2.1 Functional Unit and Register

Functional units (FU), such as adders and multipliers, are assumed to have no output register. A pipelined FU contains internal register(s) for its intermediate results of pipeline stages except for the last stage. To store the computation result, registers outside the FU is used. In addition, FUs do not have registers or latches at the input. FUs with input registers and/or output registers could be supported by modifying the binding algorithm shown in Section 4.3.

In the remaining of this paper, a module refers to either an FU or a register.

The shape of module type t is rectangle with width w_t and height h_t. The input

†1 Saitama University

© 2009 Information Processing Society of Japan

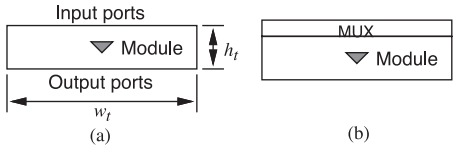

Fig. 1 The model of a module. (a) Size and input/output ports. (b) A module with multiplexor(s) on input port.

ports of a module are located on the middle of a wider edge of the rectangle and the output port is located on the edge opposite to the input ports. This is illustrated in **Fig. 1** (a). In the case a module receives data from several sources, one or more multiplexors are necessary. These MUXs are placed on the input side of the module as shown in Fig. 1 (b).

2.2 Module Interconnection and Its Energy Dissipation

The interconnects are implemented so that only the wire of the necessary length is charged or discharged to transfer data. Such a connection is implemented by a segmented bus (split bus [4]) or a point-to-point wiring. The interconnect energy dissipation is proportional to the wire length from the source to the destination [1]. The wire length can be computed as the shortest path between the source and the destination, with the constraint that the path does not go over any module. The source is an output port of a module and the destination is an input port of another module. The energy dissipation E_d by a data communication d is given by the following equation

$$E_d = WL_d \times K_d \tag{1}$$

where WL_d is the wire length between the source and the destination of d and K_d is a constant.

By taking into account that we generally have one or more data communication between identical pair of source and destination, the total interconnect energy dissipation, EC, is given by the following equation

$$EC = \sum_{d \in D} \{E_d \times M_d\} \tag{2}$$

where D is the set of all the pair of communication source and destination, and M_d is the multiplicity of the communication d.

For simplicity, we assume K_d is common to all the interconnects. Therefore we define the interconnect energy dissipation parameter EC as follows.

$$EC = \sum_{d \in D} \{WL_d \times M_d\} \tag{3}$$

Minimizing the interconnect energy dissipation of data communication is achieved by minimizing EC.

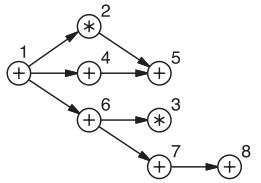

Fig. 2 An example DFG.

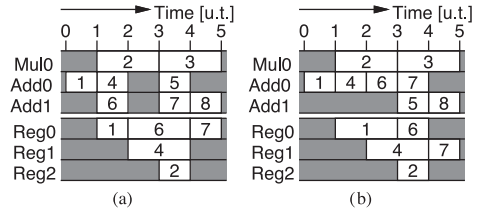

Fig. 3 The schedule and binding. (a) A schedule and its binding A. (b) Another schedule and its binding B.

3. Minimizing Interconnect Energy Dissipation

3.1 Motivational Example

Here we show a simple example where the energy dissipation of data communication is minimized not only by binding and floorplanning, but also by scheduling.

Figure 2 shows the example data-flow graph (DFG). It consists of two multiplications (nodes 2 and 3, marked as '$*$' inside the node) and 6 additions (other nodes, marked as '+' inside the node). The addition takes 1 unit of time (u.t.) and the multiplication takes 2 u.t. The requirement is to execute all the computations within 5 u.t.

Figure 3 (a) shows a schedule which minimizes the required number of modules. The binding, also shown in the figure, is obtained so that the number of

© 2009 Information Processing Society of Japan

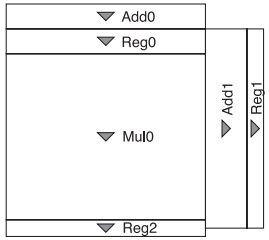

Fig. 4 Floorplan for schedule and binding A.

Table 1 Communication analysis for design of Fig. 4.

Communication d	WL_d	M_d
Add0 → Reg0	29	1
Add0 → Reg1	0	1
Add1 → Reg0	0	2
Mul0 → Reg2	0	1
Reg0 → Add0	21	1
Reg0 → Add1	30	3
Reg0 → Mul0	0	2
Reg1 → Add0	31	1
Reg2 → Add0	25	1
EC	196	

communication source and destination pairs is minimized.

Based on the binding, floorplanning is performed so that the communication energy dissipation parameter EC is minimized. The resultant floorplan is shown in **Fig. 4**. The communication requirement d, its wire length WL_d, and the multiplicity M_d are shown in **Table 1**. For example, the communication from the output of register 'Reg0' to input of adder 'Add1' (the sixth row) uses the wire of length 30 and this communication is done three times. The minimized parameter EC for this design is 196.

For the same DFG, another schedule shown in Fig. 3 (b) is obtained by exploring feasible schedules so that the parameter EC would be minimized after floorplanning. The binding result is also shown in Fig. 3 (b). The floorplanning is performed similarly to minimize EC. The result of floorplanning is shown in **Fig. 5** and **Table 2**. The minimized parameter EC for this design is 138. Therefore about 30% reduction of energy dissipation of data communication may be

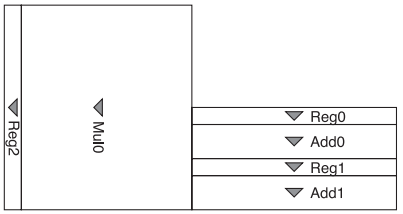

Fig. 5 Floorplan for schedule and binding B.

Table 2 Communication analysis for design of Fig. 5.

Communication d	WL_d	M_d
Add0 → Reg0	30	2
Add0 → Reg1	0	2
Mul0 → Reg2	0	1
Reg0 → Add0	0	3
Reg0 → Mul0	14	2
Reg1 → Add1	0	2
Reg2 → Add1	50	1
EC	138	

achieved.

This example suggests that the interconnect energy dissipation of data communication can be reduced by scheduling, binding, and floorplanning which are oriented to minimizing the energy dissipation.

It is also suggested that EC may depend on how data communication requirements are distributed among modules. In the design shown in Fig. 3 (a) and Fig. 4, the data communications are distributed among 9 pairs of modules. The smaller EC is obtained by the design shown in Fig. 3 (b) and Fig. 5 where the data communications are distributed among only 7 pairs of modules. EC computed by Eq. (3) can be minimized by minimizing the wire length WL_d, which is achieved by placing communicating modules near to each other. However, such placement is difficult when the data communications are distributed among many pairs of modules. If the data communications are concentrated onto rather small number of pairs of modules, it is possible to place modules so that the wire lengths between communicating modules achieve their minimum. Hence reducing the distribution of data communications is effective in minimizing the energy

© 2009 Information Processing Society of Japan

dissipation of data communication.

3.2 The proposed Method and Related Works

In Refs. 5), 6) (**Fig. 6** (a)), the optimum binding which leads to the minimized power dissipation is searched as follows: starting from the initial binding, (1) change the current binding, (2) reschedule operations if necessary, (3) estimate the interconnect lengths and compute the power dissipation, (4) accept the change if the power is reduced, and (5) repeat (1)–(4) with the current power dissipation information fed back to the binding changed in (1). In this method, it is difficult to explicitly take into account the time constraint such as the iteration clock cycle or the execution time. In Ref. 7) (Fig. 6 (b)), scheduling, binding, and floorplanning are simultaneously considered to more directly minimize the interconnect power dissipation. Given a combination of scheduling and binding, a floorplan is derived and the interconnect power dissipation is computed. By using the simulated annealing (SA), the best combination of scheduling and binding is explored which leads to the minimum interconnect power dissipation. It is anticipated, however, that the combinations of scheduling and binding are enormous for larger designs and it takes long time for exploration. In addition, since the employed floorplanning method is simple to quickly obtain the floorplan, the optimality may be sacrificed.

In this paper, we propose a method (Fig. 6 (c)) where floorplanning is separated from scheduling and binding so that larger DFGs can be treated. First, the scheduling is explored by using SA. For each scheduling candidate, a heuristic binding is performed and the cost is obtained by evaluating the binding result. The purpose of scheduling and binding is to concentrate data communications to the small number of interconnects. The detail is presented in the next section. Then the floorplanning is explored by using the combination of SA and the sequence-pair [8]. The objective is to minimize the energy dissipation parameter EC. By the floorplan exploration, a sufficient level of optimality would be achieved.

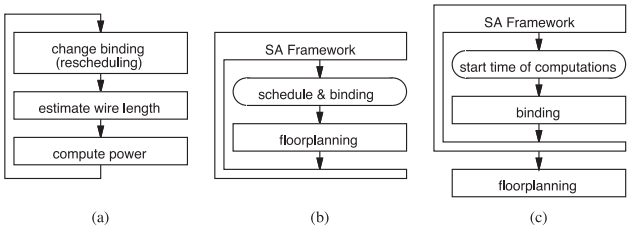

Fig. 6 Design methods. (a) Refs. 5), 6). (b) Ref. 7). (c) Proposed.

4. Scheduling and Binding for Minimizing Interconnect Energy Dissipation

4.1 Schedule with Struts

Scheduling is the problem to determine the execution start time of each computation in a given DFG so that all the precedence relations between computations are satisfied. As soon as possible (ASAP) schedule is a solution to the scheduling problem. It is obtained by solving the longest path problem for the graph where a node corresponds to a computation in the DFG and an edge corresponds to a data dependency between nodes. By ASAP scheduling, the execution start time t_j of each computation j is determined to satisfy

$$t_j = \max_{(i,j)\in E}\{t_i + w_{ij}\}. \tag{4}$$

In Eq. (4), E is the set of edges and w_{ij} is the original weight of edge (i, j) given as

$$w_{ij} = q_i - d_{ij}Tr, \tag{5}$$

where q_i is the computation duration of computation i, d_{ij} is the delay count on the edge (i, j), and Tr is the iteration clock cycles. For every directed edge (i, j), t_j is not earlier than $t_i + q_j - d_{ij}Tr$, that is the execution end time of computation i, hence the precedence is satisfied.

Figure 7 shows an example DFG (a) and its ASAP schedule (b). We assume that the computation durations q_i, q_j, and q_k are 1 and the iteration period $Tr = 2$. The node i is assumed to start at $t_i = 0$ and the end time is $t_i + q_i = 1$. A delay on the edge (i, j) ($d_{ij} = 1$) means the execution of j depends on the output of i executed in the previous iteration cycle. Because $Tr = 2$, the previous execution of i (denoted as i_{-1} in Fig. 7) starts at $t_i - d_{ij}Tr = -2$ and ends at $t_i + q_i - d_{ij}Tr = t_i + w_{ij} = -1$. The start time of j, t_j, is no earlier than the end of i, hence $t_j = -1$. The node k depends on i and j, and the earliest start time $t_k = \max\{t_i + w_{ik}, t_j + w_{jk}\} = \max\{1, 0\} = 1$.

© 2009 Information Processing Society of Japan

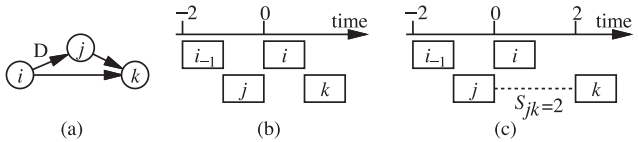

Fig. 7 ASAP schedule and strut. (a) example DFG. (b) ASAP schedule of (a). (c) ASAP schedule of (a) with strut $S_{jk} = 2$.

When a non-negative value is added to the weight of edge (i, j) and ASAP scheduling is performed, the precedence between computations i and j is still satisfied. Such a value is called *strut* and denoted by S_{ij}. The effect of adding struts is that the execution start times of j and subsequent computations are delayed. Therefore, a schedule different from the original ASAP schedule can be obtained by introducing struts. For example, if the strut $S_{jk} = 2$ is added on the edge (j, k) in Fig. 7 (a), the node k is delayed by 2 from the end of j. Therefore, $t_k = \max\{t_i + w_{ik}, t_j + w_{jk} + S_{jk}\} = \max\{1, 2\} = 2$ as shown in Fig. 7 (b).

The algorithm of scheduling with struts is as follows.

Algorithm for Scheduling with Struts

Input: DFG $= (N, E)$

N: the set of computation nodes

E: the set of dependency edges between nodes

w_{ij}: the weight of the edge $(i, j) \in E$

S_{ij}: the strut of edge $(i, j) \in E$

s: the reference node in N

Output: the execution start time t_i of all the nodes $i \in N$

(1) Set $t_i = -\infty$ for every node i. Set $t_s = 0$ for the reference node s.

(2) For every node $j \in N$, compute t'_j as follows.

$$t'_j = \max_{(i,j)\in E} \{t_i + w_{ij} + S_{ij}\}$$

(3) For each node $j \in N$, update t_j as t'_j if $t_j < t'_j$. While there exists an update on any of the node, return to (2). End otherwise.

This algorithm solves the longest path problem by Bellman-Ford method [9]. Therefore, it is guaranteed that the algorithm terminates if the values of struts are appropriately selected so that no positive cycle exists, and that the set of obtained t_j satisfies all the precedence relations. By assigning various values to the struts S_{ij}, we can explore the feasible schedule.

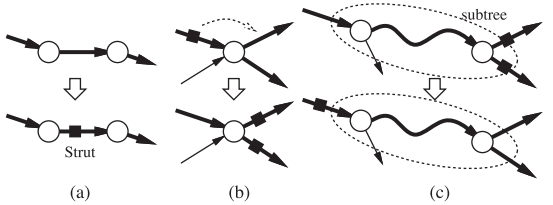

Fig. 8 Change of strut. (a) Insert single strut. (b) Move forward over node input to node output. (c) Move backward from the output to the input of a sub-tree.

4.2 Schedule Exploration by Simulated Annealing

Simulated annealing (SA) is used to determine the best combination of the struts. The optimization algorithm is described below.

Scheduling Exploration

(1) Set the strut $S_{ij} = 0$ for every edge $(i, j) \in E$.

(2) Determine the execution start time t_j for every node $j \in N$ by using the algorithm for scheduling with struts. This is the first scheduling candidate SC and evaluate the initial objective cost $Cost$ of SC.

(3) Set the temperature $T = T_{start}$.

(4) Generate a new scheduling candidate by performing one of the following manipulations (**Fig. 8**) to derive new strut values S'_{ij} of edge $(i, j) \in E$ so that no precedence relations are violated. Let E_X denote the set of edges where the value of strut changes.

(a) insert or decrease a strut

(b) move struts over single node

(c) rewind struts for a sub-tree

(5) Determine the execution start time t_j for every node $j \in N$ by using the values of struts S'_{ij} and the algorithm for scheduling with struts. This is the new scheduling candidate SC'.

(6) Evaluate the objective cost $Cost'$ of SC'.

(7) Accept the new schedule candidate SC' by assigning $S_{ij} = S'_{ij}$ for all the edges $(i, j) \in E_X$ if $Cost' < Cost$ or at the probability

© 2009 Information Processing Society of Japan

$$p = \exp\left(-\frac{Cost' - Cost}{T}\right). \quad (6)$$

(8) Repeat steps (4) to (7) for M times where M is a predetermined number.

(9) Decrease the temperature T by multiplying a constant $\alpha < 1$. If $T < T_{end}$, then terminate this procedure. Otherwise, go to step (4).

The manipulations in step (4) are described as follows. The execution start time t_j for every node $j \in N$ is obtained as the solution of the longest path problem. The set of edges used as the longest paths constructs a tree. The manipulation increases or decreases only the struts of tree edges. In Fig. 8, the tree edges are shown in bold. Note that some manipulations may not be possible. The manipulation is allowed only when no precedence relation of nodes is violated.

(a) *Insert or decrease a strut*

A value of strut is inserted to an edge with no strut or an existing strut is decreased. In the case of inserting a strut, its value is chosen so that any positive cycle is not introduced in the DFG. The maximum possible value $Slack_{ij}$ of the strut for edge (i, j) is given as

$$Slack_{ij} = U^j(j) - \left(U^j(i) + w_{ij}\right) \quad (7)$$

where $U^j(k)$ is the ASAP schedule of node k with node j being the reference node (hence $U^j(j) = 0$). A value $0 < S_{ij} \leq Slack_{ij}$ is selected and added to the edge (i, j). In the case of decreasing a strut, a value v is selected so that $0 < v \leq S_{ij}$ and S_{ij} of the edge (i, j) is decreased by v. This manipulation has an effect to schedule j and subsequent nodes in later or earlier control steps.

(b) *Move struts over single node*

A node j is chosen, a value u is removed from the strut on the tree edge (i, j), and adds u to all the struts on tree edges outgoing from node j. The value u is arbitrarily selected within the range between 1 and the maximum movable strut, MMS, given as

$$MMS = \min\left\{S_{ij}, \min_{(k,j)\in E, k\neq i} Slack_{kj}\right\}. \quad (8)$$

The second term of the right hand side gives the minimum of slack time of cotree edges incoming to j. This manipulation has an effect to schedule node j in earlier control step.

(c) *Rewind struts for a sub-tree*

A node j is chosen and a sub-tree rooted at j is identified. Let $ST(j)$ denote the sub-tree. The sub-tree is identified so that every edge outgoing from the sub-tree has the non-zero strut if it is a tree edge or has the non-zero slack time, $Slack_{vw}$, if it is a cotree edge. The maximum movable strut, MMS, is calculated as the smallest value of these struts and slack times as below.

$$MMS = \min\left\{\min_{(m,n)\in E_t(j)} S_{mn}, \min_{(m,n)\in E_{ct}(j)} Slack_{mn}\right\}, \quad (9)$$

where $E_t(j)$ and $E_{ct}(j)$ are respectively the set of tree and cotree edges outgoing from $ST(j)$. Then a value $0 < u \leq MMS$ is selected and u is subtracted from the struts of tree edges outgoing from the sub-tree and added to the tree edge (i, j) incoming to j. This manipulation has an effect to schedule nodes in the sub-tree in later control steps.

4.3 Interconnect Energy Dissipation Oriented Binding

Once the schedule is given, the binding is done so that computations executed simultaneously are not bound to the same FU (similarly, data to be stored at the same time are not bound to the same register). The objective of the binding is that the floorplan with minimized interconnect energy dissipation can be obtained. As suggested by the motivational example in Section 3, the desired binding is that the data communications are concentrated on a small number of module pairs. To derive such a binding, the following method is used.

The computation group (CG) is defined as the subset of computations which are simultaneously executed. Let $CG(t)$ denote the CG where its computations are executed at clock cycle t. The binding is processed CG by CG. Assume that some CGs have been processed. That is, computations in the processed CGs have been bound to FUs and the data output by those computations also have been bound to registers. Among not yet processed CGs, a CG is selected and all the possible bindings are enumerated. For example, let the CG contains two computations c_1 and c_2 and three FUs A_1, A_2, and A_3 are available. Six ($= {}_3P_2$) possible bindings are ($c_1 \to A_1$, $c_2 \to A_2$), ($c_1 \to A_2$, $c_2 \to A_1$), ($c_1 \to A_1$, $c_2 \to A_3$), ($c_1 \to A_3$, $c_2 \to A_1$), ($c_1 \to A_2$, $c_2 \to A_3$), ($c_1 \to A_3$, $c_2 \to A_2$). The computations communicate with registers to receive input data or send output data. For each possible bindings, the data communications between the FUs and

© 2009 Information Processing Society of Japan

registers are evaluated and the best binding is chosen. The best binding of the CG is accepted and the CG is marked as processed. The evaluation of the data communications is described later.

To select a CG among not yet processed ones, the parameter $B(t)$ is used. $B(t)$ is the number of data which are already bound to registers and consumed or produced by the computations in $CG(t)$. With larger $B(t)$, that is, more source or destination of data communications are fixed, data communication between registers and the computations in $CG(t)$ can be evaluated more precisely. Therefore, the $CG(t)$ with the largest $B(t)$ is selected to process next.

Once the binding is accomplished, the score of the binding is computed. Many kinds of scores which may lead to the desired floorplan are tried in the preliminary experiments [10] and the following scores are used here.

- Fanout

$$S_1 = \sum_{m \in SM} \left(\sum_{n \in SM} L_d(n, m) \right)^2 \tag{10}$$

- Square-sum of communication counts

$$S_2 = \sum_{n,m \in SM} \left(M_d(n, m) \right)^2 \tag{11}$$

- Sum of interconnects

$$S_3 = \sum_{n,m \in SM} L_d(n, m) \tag{12}$$

Let SM denote the set of modules, and $M_d(n, m)$ denote the number of data communications from the module n to the module m. In addition, $L_d(n, m)$ indicates the requirement of an interconnect from n to m. If $M_d(n, m) > 0$, $L_d(n, m) = 1$, otherwise $L_d(n, m) = 0$. In the case of Fanout, S_1 is minimized. It means that data communications from module n are concentrated onto the small number of interconnects. In the case of Square-sum of communication counts, S_2 is maximized. For example, module n outputs 5 data to two modules m_1 and m_2. When m_1 receives 4 data and m_2 receives 1 data, S_2 is 17. When m_1 receives 3 data and m_2 receives 2 data, S_2 is 13. The former is preferred because placing n and m_1 closer to each other and making the wire length shorter for this communication may lead to smaller interconnect energy dissipation, even when the wire length between n and m_2 is made longer. Minimizing Sum of interconnects S_3 is often used as the score in conventional area oriented bindings.

Data communication evaluation for the binding of a CG is described here. Let $M_d^0(n, m)$ ($M_d^1(n, m)$) denote the number of data communications from the module n to the module m before (after) the CG binding is done. Assume that a computation i in the CG is bound to an FU f. If the input data j of i is already bound to a register $r(j)$, then $M_d^1(r(j), f) = M_d^0(r(j), f) + 1$. If j is not yet bound to any register, choose a register r to which j can be bound without conflicting other data. Increase a register if it is impossible to bind j without conflicts. In the case of S_1 score, r is chosen so that the increase of fanout of r is minimum. In the case of S_2 score, r with the largest $M_d^0(r, f)$ is chosen. In the case of S_3 score, r is chosen among $M_d^0(r, f) > 0$. If there exists a tie in S_1 and S_2 cases or there is no register with $M_d^0(r, f) > 0$ in S_3 case, then choose r arbitrarily. Similarly, if the output data k of i is already bound to the register $r(k)$, then $M_d^1(f, r(k)) = M_d^0(f, r(k)) + 1$. If k is not yet bound to any register, choose r similarly as the input data j. Then, based on M_d^1 values, the score S_1, S_2, or S_3 is computed and it is the data communication evaluation of the CG binding. The smaller score is the better for the case of S_1 and S_3 scores, and the larger score is the better for S_2 score.

The binding algorithm is summarized as follows.

Binding Algorithm

Input: $CG(t)$ (the set of computations which start executions in clock cycle t)

Output: the binding of computations to FUs and data to registers

(1) Choose a clock cycle in which the most data overlap with each other. Bind these data to distinct registers.

(2) Among not yet processed CGs, choose $CG(t)$ with the largest $B(t)$.

(3) Enumerate all possible bindings of computations in $CG(t)$ to FUs. Compute the M_d^1 values for each binding. Choose the best binding according to the employed binding score. Fix the binding of computations. Mark CG as processed.

(4) Let $D(t)$ denote the set of data output by computations in $CG(t)$. Enumerate all possible binding of data in $D(t)$ to registers. Compute the M_d^1 values for each binding similarly as computations. Choose the best binding

© 2009 Information Processing Society of Japan

and fix the binding of data.

(5) If all CGs are processed, then end. Otherwise $M_d^0 = M_d^1$ and go to step (2).

By using the binding score, the *Cost* is calculated as follows and minimized in the schedule exploration,

$$\begin{array}{ll} Cost = \beta|SM| + S_k & (k = 1, 3) \\ Cost = \beta|SM| + W - S_k & (k = 2) \end{array} \tag{13}$$

W is a sufficiently large value and introduced to maximize the score S_2. $|SM|$ is the number of modules. When the weight β is positive, the data communications are optimized while the number of modules is minimized. By setting $\beta = 0$, only the data communications are considered in binding. Although the number of modules may not be minimized, the design with minimized interconnect energy dissipation would be obtained.

5. Experimental Results

The proposed method is implemented by using the programming language C++. The software is run on a PC with a 2.4 GHz processor and 512 MB memory. The CPU time ranged over 100 to 1,000 seconds for the scheduling exploration and over 50 to 1,000 seconds for the floorplanning.

In Ref. 6), it is reported that the CPU time for high-level synthesis (HLS) of the DFG with 56 nodes is 207 seconds on a PC with 1.3 GHz processor. Before HLS, a DFG simulation (SIM) is done. The CPU time is not presented precisely but reported as comparable to the CPU time of HLS. Therefore we assume the total CPU time is 400 seconds. Because the obtained output of the method [6] is the RTL description, the actual floorplan must be done as the separate design process. The CPU time for the floorplan is not included in the above CPU time (400 seconds). It is appropriate to compare the CPU time for HLS and SIM with the CPU time for scheduling and binding in the proposed method. The CPU time for the DFG with 102 nodes is 887 seconds (the average of runs for three different iteration periods Tr). Taking into account the CPU clock difference, the proposed method takes 4.1 times longer CPU time for the 1.8 times larger DFG. The CPU time for HLS generally grows much faster than the growth of DFG size. Consequently, the CPU time of the proposed method is comparable to the method [6].

The modules are designed for 0.5 μm, two-metal CMOS technology. The module width $w_M = 24$ and height $h_M = 20$ for multipliers, $w_A = 24$ and $h_A = 3$ for adders, and $w_R = 24$ and $h_R = 2$ for registers. The width of multiplexor (MUX) is $w_{MUX} = 24$. The height of MUX depends on the input count and assumed as $h_{MUX} = 1$ for 1 to 4 inputs, $h_{MUX} = 2$ for 5 to 7 inputs, $h_{MUX} = 3$ for 8 to 10 inputs, and so on. All of these length are noted in unit of length. The multiplier is two stage pipelined. Therefore, a new multiplication can be started one clock cycle after the previous multiplication was started.

First, the results of the proposed method were compared with those by the conventional method [5),6)]. Because the different technology is assumed and different modules (FUs and registers) are used, directly comparing the energy dissipation of the methods seems meaningless. Instead, energy dissipation saving (how much energy dissipation is reduced) is used to compare the performances of the methods. Two benchmark DFGs are used. The 5th order wave elliptic filter (WEF) [11] (referred as Elliptic in Refs. 5), 6)) consists of 8 multiplications and 26 additions. The differential equation (Diffeq) consists of 6 multiplications and 5 additions. Three iteration clock cycles (Tr) are tried for each DFG. The SA parameters are $T_{start} = 100$, $T_{end} = 1$, $\alpha = 0.95$, and $M = 1{,}000$ for schedule exploration, and $T_{start} = 100$, $T_{end} = 1$, $\alpha = 0.99$, and $M = 1{,}000$ for floorplanning. **Figure 9** compares the interconnect energy dissipation saving of the energy optimized design against the area optimized design. In the proposed method, the energy optimized design is obtained by using $k = 2$ in Eq. (13) with $\beta = 0$. The area optimized design is obtained by using $k = 3$ in Eq. (13) with $\beta = 100$ and the floorplanning is done to minimize the area of the rectangle bounding all the modules. Figure 9 shows that more interconnect energy dissipation can be saved by the proposed method.

Next, the binding scores of the proposed method were compared. Two more DFGs, WEF3 and 8-point 1DDCT [12], are used in addition to WEF. WEF3 is derived from the WEF by unfolding [13] it by factor 3. WEF3 consists of 24 multiplications and 78 additions, and 1DDCT consists of 11 multiplications and 29 additions. Again three iteration clock cycles (Tr) are tried for each DFG. In

© 2009 Information Processing Society of Japan

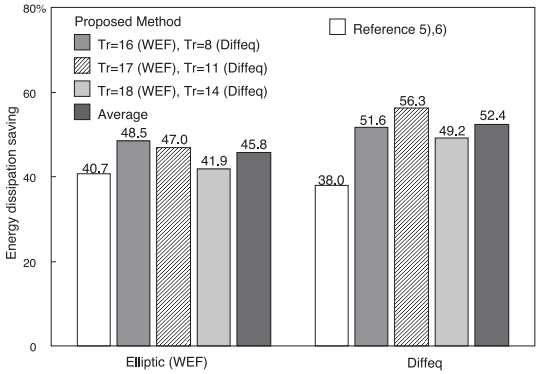

Fig. 9 Comparison of interconnect energy dissipation savings against the area-optimized results.

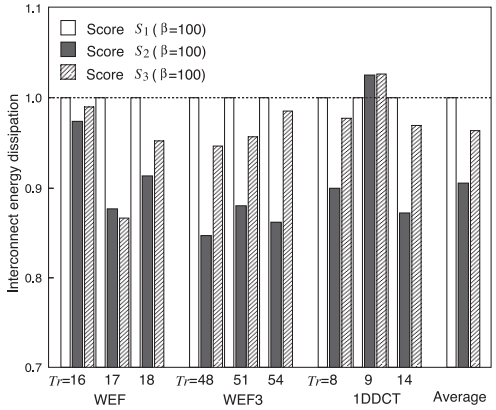

Fig. 10 Comparison of the binding scores with module minimization.

this experiment, $\beta = 100$ and $\beta = 0$ were tried. The SA parameters are the same as the first experiment.

Figure 10 shows the results for $\beta = 100$ where the minimization of the number of modules as well as the interconnect energy dissipation is considered. **Figure 11** shows the results for $\beta = 0$ where only the minimization of the interconnect power dissipation is considered. In these figures, the relative energy dissipations are shown with the case of score S_1 being the unity. In both cases, maximizing

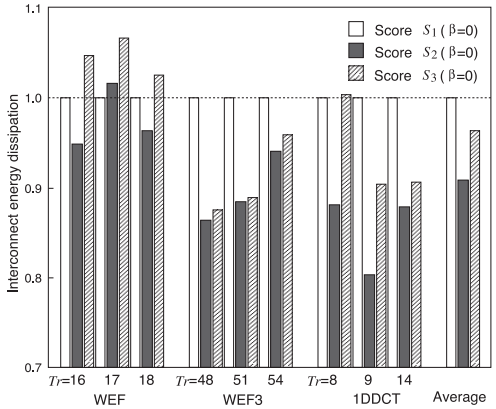

Fig. 11 Comparison of the binding scores without module minimization.

the score S_2 (Eq. (11)) provides the best results on average.

Finally, **Table 3** shows the power dissipation and the area of the designs. Shown in Table 3 are the iteration clock cycles Tr, the power dissipation shown in μW including the computations and data communications (in parenthesis is the relative value against the area optimized design), the area of the rectangle bounding all the modules relative to the area optimized design, and the content of SM (the numbers of adders, multipliers, and registers are shown from left to right). 'PO' indicates that only the interconnect power dissipation is minimized. 'WA' indicates that the area is also minimized by setting $\beta = 100$ in the binding score. In these designs, the *Cost* in Eq. (13) with $k = 2$ is used. 'AO' indicates that only the area is minimized. On average, 'PO' achieves about 15% reduction of power dissipation at the expense of 87% increase in area and 'WA' achieves about 14% reduction of power dissipation at the expense of 58% increase in area. By using modules designed for lower power, reduction of power dissipation by data communication may give larger impact on reduction of total power.

6. Conclusions

In this paper, a schedule exploration method for minimization of power dissipation of data communications on LSI is proposed. By the exploration, a best schedule is searched so that the optimized binding and floorplan are obtained,

© 2009 Information Processing Society of Japan

Table 3 Comparison of Designs @ 100 MHz.

Diffeq	AO		PO			WA		
Tr	Power	SM	Power	Area	SM	Power	Area	SM
8	402	1,1,5	360(89.6%)	108%	1,1,5	360(89.7%)	146%	1,1,5
11	294	1,1,5	260(88.5%)	102%	2,1,5	262(89.3%)	105%	1,1,5
14	219	1,1,5	207(94.3%)	182%	1,1,5	207(94.4%)	114%	1,1,5
WEF	AO		PO			WA		
Tr	Power	SM	Power	Area	SM	Power	Area	SM
16	406	3,1,10	338(83.3%)	216%	3,2,10	340(83.9%)	175%	3,2,9
17	377	2,1,9	317(84.2%)	183%	2,2,10	320(84.9%)	127%	2,1,9
18	344	2,1,9	299(86.9%)	213%	3,1,9	302(87.6%)	114%	2,1,9
WEF3	AO		PO			WA		
Tr	Power	SM	Power	Area	SM	Power	Area	SM
48	415	3,1,10	343(82.8%)	217%	3,2,10	340(82.1%)	183%	3,2,10
51	386	3,1,9	323(83.8%)	195%	3,2,9	324(84.1%)	221%	3,2,9
54	361	2,1,9	305(84.5%)	216%	3,2,10	306(84.9%)	240%	3,2,9
1DDCT	AO		PO			WA		
Tr	Power	SM	Power	Area	SM	Power	Area	SM
8	1,090	4,2,9	857(78.4%)	149%	4,3,10	870(79.6%)	175%	4,2,9
9	945	4,2,9	748(79.2%)	227%	4,3,11	780(82.5%)	130%	4,2,9
10	826	3,2,9	678(82.1%)	233%	4,2,11	692(83.8%)	165%	3,2,9
Average			84.8%	187%		85.6%	158%	

which in turn result in the minimized power dissipation of data communications. Experiments show the effectiveness of the proposed method.

Reexamination of the objective cost and the algorithm of binding, tuning the SA parameters, support of multi-destination data communication in bus architecture, consideration for chip area necessary for wire and switches, remain as future work.

Acknowledgments This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc. and Cadence Design Systems, Inc.

References

1) Taylor, N., Dey, S. and Zhao, Y.: Modeling and Minimization of Interconnect Energy Dissipation in Nanometer Technologies, *Proc. DAC 2001*, pp.754–757 (2001).
2) Raghunathan, V., Srivastava, M.B. and Gupta, R.K.: A Survey of Techniques for Energy Efficient On-Chip Communication, *Proc. DAC 2003*, pp.900–905 (2003).
3) Hsieh, C.T. and Pedram, M.: Architecural Energy Optimization by Bus Splitting, *IEEE Trans. Comput.-Aided Des. Intergrated Circuits & Syst.*, Vol.21, pp.408–414 (2002).
4) Lu, R. and Koh, C.-K.: A High Performance Bus Communication Architecture through Bus Splitting, *Proc. ASP-DAC 2004*, pp.751–755 (2004).
5) Zhong, L. and Jha, N.K.: Interconnect-aware high-level synthesis for low power, *Proc. ICCAD*, pp.110–117 (2002).
6) Zhong, L. and Jha, N.K.: Interconnect-aware low-power high-level synthesis, *IEEE Trans. Comput.-Aided Des. Intergrated Circuits & Syst.*, Vol.24, pp.336–351 (2005).
7) Prabhakaran, P., Banerjee, P., Crenshaw, J. and Sarrafzadeh, M.: Simultaneous Scheduling, Binding and Floorplanning for Interconnect Power Optimization, *12th Int. Conf. VLSI Design*, pp.423–427 (1999).
8) Murata, M., Natakake, S., Fujiyoshi, K. and Kajitani, Y.: VLSI Module Placement based on Rectangle Packing by Sequence-Pair, *IEEE Trans. Comput.-Aided Des. Intergrated Circuits & Syst.*, Vol.15, pp.1518–1524 (1996).
9) Lawler, E.L.: *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston (1976).
10) Seto, H. and Ito, K.: A Resource Binding Method for Reducint Power Consumption of LSI Data Communications (in Japanese), *IEICE Tech. Report*, Vol.VLD2007-86, pp.25–30 (2007).
11) Heemstra de Groot, S.M., Gerez, S.H. and Herrmann, O.E.: Range-Chart-Guided Iterative Data-Flow Graph Scheduling, *IEEE Trans. Circuits Syst.-I: Fund. Theory & Appl.*, Vol.39, pp.351–364 (1992).
12) Loeffler, C., Ligtenberg, A. and Moschytz, G.S.: Practical Fast 1-D DCT Algorithms with 11 Multiplications, *Proc. IEEE ICASSP '89*, pp.988–991 (1989).
13) Parhi, K.K. and Messerschmitt, D.G.: Static Rate-Optimal Scheduling of Iterative Data-Flow Programs via Optimum Unfolding, *IEEE Trans. Computers*, Vol.40, pp.178–195 (1991).

(Received May 23, 2008)
(Revised August 22, 2008)
(Accepted October 9, 2008)
(Released February 17, 2009)

(Recommended by Associate Editor: *Shigetoshi Nakatake*)

© 2009 Information Processing Society of Japan

Kazuhito Ito received the B.E., M.E., and Ph.D. degrees in Electrical Engineering from Tokyo Institute of Technology, Japan, in 1987, 1989, and 1992, respectively. He is an associate professor of the Graduate School of Science and Engineering, Saitama University, Japan. His research interests include high-level synthesis in VLSI design, VLSI signal processing, and design automation of system LSIs. He is a member of IEICE, IPSJ, and IEEE.

Hidekazu Seto received the B.E. degree in Electrical and Electronic Engineering from Saitama University, Japan, in 2007. He is currently a master degree student of the Graduate School of Science and Engineering, Saitama University, Japan. His research interests include high-level synthesis in VLSI design and design automation of system LSIs.

© 2009 Information Processing Society of Japan