IPSJ Transactions on System LSI Design Methodology Vol.5 106–117 (Aug. 2012)

[DOI: 10.2197/ipsjtsldm.5.106]

Regular Paper

Energy-efficient High-level Synthesis for HDR Architectures

Shin-ya Abe[1,a)] Masao Yanagisawa[2] Nozomu Togawa[1]

Received: November 14, 2011, Revised: February 27, 2012,
Accepted: April 19, 2012, Released: August 6, 2012

Abstract: As battery runtime and overheating problems for portable devices become unignorable, energy-aware LSI design is strongly required. Moreover, an interconnection delay should be explicitly considered there because it exceeds a gate delay as the semiconductor devices are downsized. We must take account of energy efficiency and interconnection delays even in high-level synthesis. In this paper, we first propose *a huddle-based distributed-register architecture* (*HDR architecture*), an island-based distributed-register architecture for multi-cycle interconnect communications where we can develop several energy-saving techniques. Next, we propose an energy-efficient high-level synthesis algorithm for HDR architectures focusing on multiple supply voltages. Our algorithm is based on iterative improvement of scheduling/binding and floorplanning. In the iteration process, *a huddle*, which is composed of functional units, registers, controller, and level converters, are very naturally generated using floorplanning results. By assigning high supply voltage to critical huddles and low supply voltage to non-critical huddles, we can finally have energy-efficient floorplan-aware high-level synthesis. Experimental results show that our algorithm achieves 45% energy-saving compared with the conventional distributed-register architectures and conventional algorithms.

Keywords: high-level synthesis, energy optimization, multiple supply voltages, interconnection delay, distributed-register architectures

1. Introduction

In recent LSI design, battery runtime and heat generation are becoming the two main problems. Energy-aware LSI designs are strongly required to solve both of these problems. An interconnection delay, which is a delay necessary for the communication between modules inside an LSI chip, is another problem. As device feature size decreases, the delay becomes the dominant factor of circuit total delay and it is predicted that this trend will continue over successive years. High-level synthesis is one of the LSI design automation techniques and it is quite necessary to deal with energy efficiency as well as interconnection delays there.

Several energy-aware high-level synthesis algorithms have been proposed which deal with multiple supply voltages [1], [6], [9], [10], [14], [15], [16], [19], [20]. Reducing the supply voltage, however, increases the circuit delay. To satisfy the overall throughput constraint, the high supply voltage should be assigned to elements on critical paths and the low supply voltage should be assigned to elements on the non-critical paths. These algorithms proposed so far, however, optimizes only power or energy, and they do not consider the interconnection delays at all.

Recently, several interconnection-aware high-level synthesis algorithms have been proposed [2], [5], [7], [12], [13]. They are not based on a traditional centralized-register architecture, but they are based on a distributed-register architecture which realizes multi-cycle interconnect communications by giving local registers to functional units.

A generalized distributed-register architecture (*GDR architecture*) is proposed in Ref. [13]. GDR realizes multi-cycle interconnect communications by preparing two kinds of registers: local registers and shared register groups. In Ref. [13], scheduling/binding as well as floorplanning are simultaneously optimized by using iterative synthesis flow. Since functional units, shared/local registers, and global/local controllers can be very flexibly arranged on a GDR architecture, it can obtain high performance design. However, the flexibility of GDR may become the disadvantage in terms of energy efficiency. Power reduction techniques including the ones using multiple supply voltages assume adding new modules such as a level converter. According to Ref. [13], the synthesis problem is difficult enough only to consider shared/local registers and global/local controllers. It is difficult to add any new modules to GDR architecture.

A regular distributed-register architecture (*RDR architecture*) is proposed in Ref. [2]. RDR divides a chip into uniform-sized islands and arranges functional units, a register file, and a controller in each island. Inside an island, interconnection delay can be ignored by using a local register. By introducing uniform-sized islands, RDR realizes multi-cycle interconnect communication during inter-island data transfer. In RDR, it is very easy to predict interconnection delays even in high-level synthesis stage since an entire chip is divided into uniform-sized islands. In addition, it is easy to cope with the module addition/deletion in an is-

[1] Department of Computer Science and Engineering, Waseda University, Shinjuku, Tokyo 169–8555, Japan
[2] Department of Electronic and Photonic Systems, Waseda University, Shinjuku, Tokyo 169–8555, Japan
[a)] shinya.abe@togawa.cs.waseda.ac.jp

© 2012 Information Processing Society of Japan

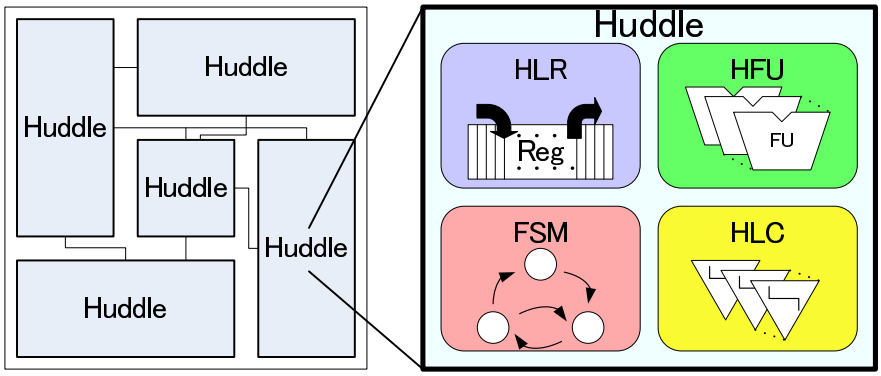


Fig. 1 A huddle configuration.

land since the island abstracts modules inside. RDR also has several improved architectures: RDR-Pipe [2] decreases the number of connections by adding a new module; DRFM [3] is proposed to implement the RDR architecture on FPGA devices; Ref. [17] proposes fault-secure high-level synthesis on RDR architectures by adding new modules to vacant islands. However, RDR has the area overhead since it divide an entire chip into uniform-sized islands. Area overhead causes useless modules and will increase interconnection delays.

Based on the above discussions, we propose a high-level synthesis algorithm considering energy-efficiency and interconnect delays simultaneously. First, we propose a *huddle-based distributed-register architecture* (*HDR architecture*), which is one of the distributed-register architecture focusing on energy-efficiency. In HDR, functional units, registers, controllers, and level converters are abstracted into a *non*-uniform island, which we call a *huddle*.

Second, we propose a high-level synthesis algorithm which is associated with an HDR architecture. The proposed algorithm can reflect floorplan information in scheduling/binding by using iterative synthesis flow. At that time, modules which are placed close to each other are *huddled* into a non-uniform island. Then each huddle shares functional units, registers, controllers, and level converters. Interconnection delays inside a huddle can be ignored by using local registers inside and HDR also support multi-cycle interconnect communications during inter-huddle data transfer. Every huddle has its own supply voltage; low supply voltages are assigned to huddles on non-critical paths and high supply voltages are assigned to huddles on critical paths. Experimental results show that our algorithm achieves 45% energy-saving compared with the conventional distributed-register architectures and conventional algorithms.

2. HDR Architecture

In this section, we briefly review recent distributed-register architectures, GDR and RDR, and point out that they are not suitable for applying energy-saving techniques. After that, we propose a new distributed-register architecture called HDR.

Generalized distributed-register (GDR) architectures proposed in Ref. [13] can synthesize high-performance and small-area circuits by introducing shared/local registers and global/local controllers. Since functional units, shared/local registers, and global/local controllers can be very flexibly arranged on a GDR architecture, it can obtain high performance design. However, this flexibly can become a disadvantage that its associated high-level synthesis problems are too complex [13] where we have to consider which register is shared or local as well as which controller is global or local. In order to realize power reduction using multiple supply voltages, it is necessary to add new modules, such as level converters when changing voltages. It definitely increases the complexity of the high-level synthesis problem and it must be very difficult to cope with energy-saving techniques in GDR architecture synthesis.

Regular distributed-register (RDR) architectures proposed in Refs. [2], [17] can predict interconnection delays in high-level synthesis accurately by dividing a chip into uniform-sized islands. It arranges functional units, local registers, and a controller inside an island, and multi-cycle interconnect communication during inter-island data transfer can be also realized. In RDR architectures, a module can be easily added to an island since it abstracts each module inside. Reference [17] realizes fault-secure high-level synthesis using RDR based on adding functional units to an island. On the other hand, RDR architectures have significant area overhead since they divide a chip into uniform-sized islands. Area overhead may lead the increase of useless modules as well as interconnection delays. We can say that RDR architectures are not suitable for applying energy-saving techniques, either.

Distributed-register architectures suitable for applying energy-saving techniques are the ones which have small area and low power consumption and in which it is easy to add new modules. Based on the discussion above, we propose a huddle-based distributed-register (HDR) architecture, which is one of the distributed-register architecture but has energy-efficiency combining the advantages of RDR and GDR.

Our HDR architectures introduce a non-uniform sized island called a *huddle* into GDR architectures in which each module inside is abstracted. As seen in RDR architectures, it is very easy to add new modules into a non-uniform sized island. Our huddle has non-uniform rectangular area determined by clock period constraints which includes functional units, registers, controllers, and level converters. Since our huddles have non-uniform rectangular area, an HDR architecture can be synthesized with small area and small energy consumption.

Figure 1 shows a huddle configuration. A huddle h consists of the following components:

© 2012 Information Processing Society of Japan

Huddled Local Registers (HLRs): Dedicated local registers in h.

Huddled Functional Units (HFUs): Dedicated functional units in h. HFUs can only access the HLRs in h.

Finite State Machine (FSM): A dedicated controller in h. FSM controls the HFU and the HLR in h.

Huddled Level Converters (HLCs): Dedicated level converters in h. HLCs are used during inter-huddle data transfer across different voltage huddles.

Huddles are placed as in **Fig. 2**. Interconnection delays can be ignored by using HLRs inside a huddle and multi-cycle interconnect communication during inter-huddle data transfer can be realized. HLCs are used during multi-cycle interconnect communication across different voltage huddles.

Example 1. ***Figure 3*** *shows the high-level synthesis results of HDR, GDR* [13]*, and RDR* [2]*. DCT is used as a benchmark application. HDR drastically reduces synthesis complexity as compared to GDR because HDR abstracts modules by introducing huddles. HDR also reduces area overhead as compared to RDR because huddles have non-uniform rectangular area.* □

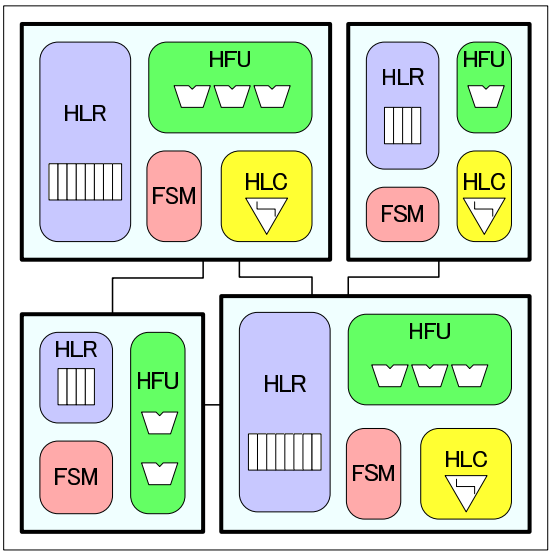


Fig. 2 HDR architecture.

3. Problem Definition

A control-data flow graph (CDFG) $G(N, E)$ is a directed graph, where a node set N is composed of an operation node set N_o and a branching control node set N_c (start and end nodes of conditional branches), and an edge set E is composed of a data-flow edge set E_d and a control-flow edge E_c set. T_{clk} refers to a clock period constraint and $S_{\max}$ refers to a control step constraint.

Let $F = \{f_1, \cdots, f_p\}$ be a set of functional units and $D_f(f_i)$ be a delay of the functional unit f_i in F. $S_f(f_i)$ shows the number of control steps required to execute the functional unit f_i and $S_f(f_i)$ is defined by $S_f(f_i) = \lceil D_f(f_i)/T_{clk} \rceil$. Let $E(f_i)$ be the energy consumed by the functional unit f_i in $S_f(f_i)$ steps.

Let $H = \{h_1, \cdots, h_q\}$ be a set of huddles in our HDR architecture. Each functional units are bound to any one of the huddles. $Hud(f_i)$ is the huddle to which f_i is bound. $F(h_j)$ is a set of functional units which are bound to h_j. $D_{reg}(h_j)$ is a delay of HLRs in h_j.

We consider the three supply voltages, v_l, v_m, and v_h ($v_l < v_m < v_h$), which are assigned to each huddle. $V(h_j)$ is a supply voltage which are assigned to the huddle h_j. $D_{lc}(v_l, v_m)$ is a delay of a level converter which changes the voltage from v_l to v_m. Likewise, we can define $D_{lc}(v_l, v_h)$, $D_{lc}(v_m, v_l)$, and so on.

$Slack(f_j)$ is defined by:

$$Slack(f_j) = T_{clk} \cdot S_f(f_i) - D_f(f_i). \quad (1)$$

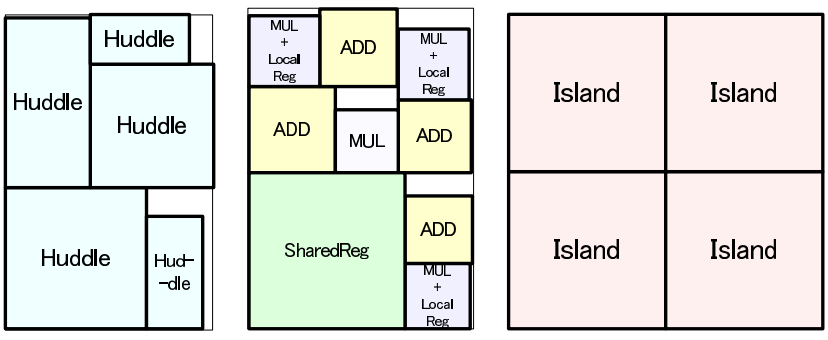
$Slack(f_j)$ shows the slack time which can be used by data transfer for succeeding operations.

According to the island defined by RDR [2], The width and height of each huddle must satisfy the following *huddle size constraint*:

$$2 \cdot D_w(W(h_j) + H(h_j)) + D_{reg}(h_j) \leq \min_{f_i \in F(h_j)} \{Slack(f_i)\} \quad (2)$$

where $W(h_j)$ and $H(h_j)$ are the width and height of the huddle h_j, respectively. $D_w(x)$ is an interconnection delay whose length is x. In our proposed algorithm, we obtain the value of $(W(h_j)+H(h_j))$ so that it satisfies the huddle size constraint and determine $W(h_j)$ and $H(h_j)$ by using the aspect ratio predefined for each huddle.

Let $Dist(h_j, h_k)$ be the Manhattan distance between the huddles h_j and h_k. Then $D_w(Dist(h_j, h_k))$ shows the interconnection delay between them. Let f_i be a functional unit bound to the huddle h_j,



(a) HDR. (b) GDR. (c) RDR.

Fig. 3 Experimental results of HDR, GDR [13], and RDR [2] when DCT is synthesized.

© 2012 Information Processing Society of Japan

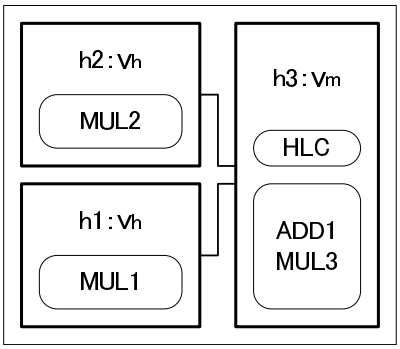


Fig. 4 An example of HDR architecture configuration.

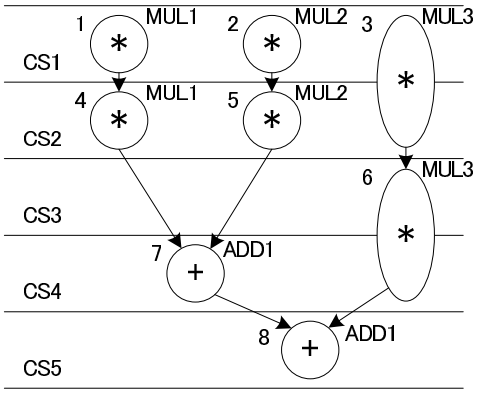


Fig. 5 An example of scheduling/binding for HDR architecture.

i.e., $Hud(f_i) = h_j$. $Tr(f_i, h_k)$ shows the inter-huddle data transfer delay from f_i to HLRs in h_k which is defined by:

$$Tr(f_i, h_k) = D_w(Dist(h_j, h_k)) + D_{lc}(V(h_j), V(h_k)) + D_{reg}(h_k). \tag{3}$$

$DT(f_i, h_k)$ shows the number of clock cycles required to transfer data from f_i to h_k which is defined by:

$$DT(f_i, h_k) = \begin{cases} 0, & (Slack(f_i) \geq Tr(f_i, h_k)) \\ \lceil Tr(f_i, h_k)/T_{clk} \rceil. & (Slack(f_i) < Tr(f_i, h_k)) \end{cases} \tag{4}$$

In the case of $Slack(f_i) \geq Tr(f_i, h_k)$, the functional unit f_i directly stores its output in the register file of huddle h_k. Thus, the data transfer requires no extra cycles. In the case of $Slack(f_i) < Tr(f_i, h_k)$, the functional unit f_i first stores its output into the local register file in its huddle h_j (= $Hud(f_i)$). In the next cycle, we perform the data transfer from the huddle h_j to the huddle h_k. The data transfer requires $\lceil Tr(f_i, h_k)/T_{clk} \rceil$ cycles. Then the data transfer table DT can be defined by a $p \times q$ matrix whose (i, k)-element is expressed by $DT(f_i, h_k)$.

Example 2. ***Figure 4*** *and* ***Fig. 5*** *show an example of HDR architecture when a clock period constraint T_{clk} = 3 ns and a control step constraint $S_{\max}$ = 5 are given.* ***Table 1*** *shows delay and energy consumption of each functional unit.* ***Table 2*** *shows delay of each level converter.*

As in Fig. 4, the input functional units are f_1 = MUL1, f_2 = MUL2, f_3 = MUL3, and f_4 = ADD1. In this example, we have huddle configurations of $F(h_1) = \{f_1\}$, $F(h_2) = \{f_2\}$, and $F(h_3) = \{f_3, f_4\}$. Voltages are assigned to the huddles as in Fig. 4: $V(h_1) = V(h_2) = v_h$ and $V(h_3) = v_m$.

Let $D_w(Dist(h_1, h_3)) = 1$ ns and $D_{reg}(h_3) = 0.5$ ns. $Slack(f_1)$ can be calculated by:

$$Slack(f_1) = 3\,\text{ns} \times 1 - 2\,\text{ns} = 1\,\text{ns}.$$

$Tr(f_1, h_3)$ is calculated by

$$Tr(f_1, h_3) = 1\,\text{ns} + 0.5\,\text{ns} + 0.5\,\text{ns} = 2\,\text{ns}.$$

Since $Slack(f_1) < Tr(f_1, h_3)$, $DT(f_1, h_3)$ can be calculated by:

$$DT(f_1, h_3) = \lceil 2/3 \rceil = 1.$$

Data transfer from the functional unit f_1 to the huddle h_3 requires 1 clock cycle. As in Fig. 5, the data transfer from the node 4 (*) to the node 7 (+) requires extra control step (CS3) for multi-cycle interconnect communication. □

Table 1 Delay/energy of functional units.

	Adder	Multiplier
v_h	1 ns/144 fJ	2 ns/1,440 fJ
v_m	2 ns/100 fJ	4 ns/1,000 fJ
v_l	4 ns/64 fJ	8 ns/640 fJ

Table 2 Level converter delays.

	v_h	v_m	v_l
v_h	-	0.5 ns	1.0 ns
v_m	0.5 ns	-	2.0 ns
v_l	1.0 ns	2.0 ns	-

Based on the above definitions, our high-level synthesis problem is defined as follows:

Definition 1. *Our high-level synthesis problem is, for a given CDFG, a clock cycle constraint, a control step constraint, and a set of functional units, to assign each operation node to a control step and a functional unit, to bind each functional unit to each huddle, and to assign a supply voltage to each huddle so that the given CDFG is executed correctly considering multi-cycle interconnect communications. The objective is to minimize the total energy consumption.*

4. An Energy-efficient High-level Synthesis Algorithm for HDR Architectures

In this section, we propose a new high-level synthesis algorithm targeting HDR architectures which deals with multiple supply voltages and multi-cycle interconnect communication simultaneously.

Generally, high-level synthesis algorithms considering multi-cycle interconnect communication are composed of schedulings, bindings, and floorplannings and classified into the following two types:

Type 1: Schedulings, bindings, and floorplannings are executed a predetermined number of times in a predetermined order.

Type 2: Schedulings, bindings, and floorplannings are executed repeatedly as an iterative refinement flow.

In Type 1, a required time to synthesize a chip can be expected easily since how many times each synthesis step is executed and its execution order are determined. If we know in ad-

© 2012 Information Processing Society of Japan

vance how many times we need to perform each high-level synthesis step as well as its best execution order, Type 1 will be the best choice. MCAS [2], one of the RDR architecture synthesis algorithms, uses an approach based on Type 1 above. Since RDR architectures have uniform-sized islands, inter-island delays are unchanged even if RDR island configurations are changed. Then we can execute a predetermined design flow as in Type 1 above.

In Type 2, several informations such as scheduling results and placement results are fed back to each other since each synthesis step is executed repeatedly as many times as needed. A GDR architecture synthesis algorithm proposed in Ref. [13] uses an approach based on Type 2. By iteratively executing scheduling/binding steps and floorplanning steps, a current scheduling/binding step can consider interconnection delays obtained in a previous floorplanning step. The shape and size of each module are determined in a scheduling/binding step and a floorplanning step is done using these module informations. Because each synthesis step affects each other, an iterative refinement flow as in Type 2 must be the best choice targeting GDR architectures.

As far as we know, all the existing high-level synthesis approaches based on Type 1 just ignore level converter delays or assume that level converters are inserted between all the two modules [1], [6], [9], [10], [14], [15], [16], [19], [20]. This is because it is very difficult to insert level converters and other required components when they are needed. On the other hand, we can consider level converters and other required components in scheduling and binding according to module configurations determined at the previous iteration in Type 2. It is very natural that we develop an algorithm based on Type 2 when we consider multiple supply voltages in HDR architectures. Totally, we can say that Type 2 will be the best choice in HDR architecture synthesis algorithms.

An HDR architecture synthesis algorithm based on Type 2 must have the following four steps in each iteration:

- **scheduling/binding**
- **register/controller synthesis and floorplanning**
- **huddling, and**
- **unhuddling.**

In a scheduling/binding step, each operation node in a CDFG is assigned to a control step and a functional unit considering multiple supply voltages and multi-cycle interconnect communications. In a register/controller synthesis and floorplanning step, register file and controller configuration in each huddle are determined using a scheduling/binding result and every huddle is placed on a chip. In a huddling step, adjacent huddles are merged into a single huddle. In an unhuddling step, a single huddle is partitioned into several huddles.

Now we face a problem when and how many times each of the above four synthesis steps is executed in each iteration (see **Fig. 6**). Assume that we have initial huddle configurations somehow. Then we can execute (i) a scheduling/binding step based on them. After a scheduling/binding step is done, (ii) a register/controller synthesis and floorplanning step must be executed since an operation scheduling to a control step and binding to a function unit may be changed. After that we can merge two

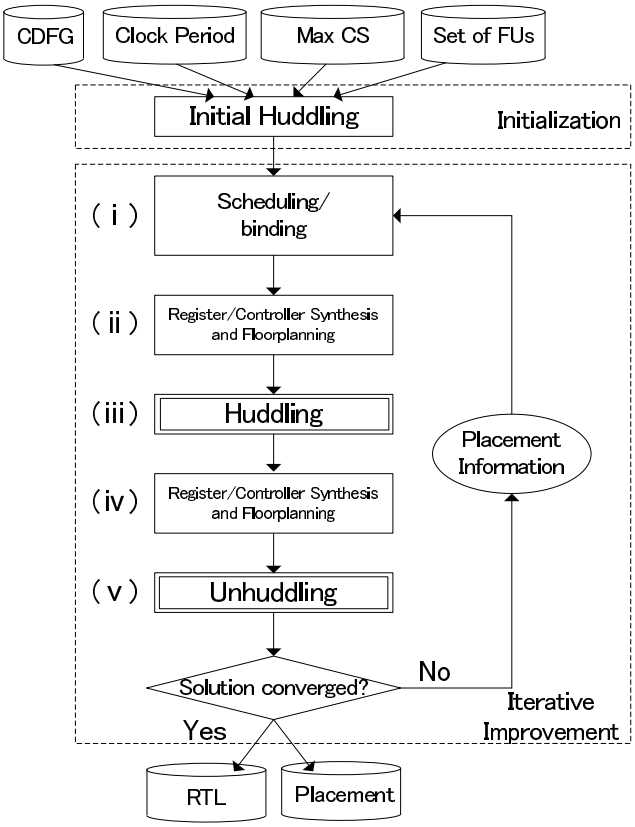


Fig. 6 Energy-efficient high-level synthesis algorithm targeting HDR architectures.

or more huddles into a single huddle since floorplanning may be changed. This means that (iii) a huddling step can be done after Step (ii). If several huddles are merged into a single huddle, a register/controller synthesis is needed since each huddle has a single register file and a controller. We need (iv) a register/controller synthesis and floorplanning step again. Finally, we try (v) an unhuddling step and if no huddles are unhuddled, we can finish the loop or we continue Steps (i)–(v) again.

The remaining problem is how to obtain initial huddle configurations. We can assume the following two initial huddle configuration options:

Option 1: As an initial state, we assume a single huddle which contains all the given functional units.

Option 2: As an initial state, we assume several huddles, each of which includes only a single functional unit.

In Option 1, our synthesis flow is roughly based on unhuddling a single huddle into multiple huddles. However, we cannot find out which part in a huddle will cause a multi-cycle commutation since we have only a single huddle or two in an early iteration stage. Moreover, we cannot assign multiple supply voltages to a single huddle since each huddle has its own supply voltage.

In Option 2, our synthesis flow is roughly based on huddling two or more huddles into a single huddle. If two or more huddles are placed close to each other, they should be merged into a single huddle unless they cause interconnection delay errors. We can also deal with multiple supply voltages by considering multiple huddles and assigning an appropriate supply voltage to each of them.

© 2012 Information Processing Society of Japan

Based on the above discussion, we can say that Option 2 is the best choice as our initial huddle configurations. Overall, we can summarize that Fig. 6 shows the best synthesis flow targeting HDR architectures. In initial huddling, initial huddle configuration and placement are determined by given functional units. If p functional units are given as input, we prepare just p huddles in which each functional unit is assigned to each huddle. We merged huddles by huddling during Steps (i)–(v) iteratively. When no huddles are partitioned in Step (v) (in other words, no timing violations occur in Step (iv)), the iteration is finished [*1]. In the rest of this section, we will propose each process in Fig. 6. Note that we deal with only DFG for simplicity as a motivated example but we can deal with CDFG similarly. In the same way, we only deal with adders and multipliers with fixed bit width as functional units for simplicity.

4.1 Initial Huddling

In initial huddling, initial huddle configuration and placement are determined by given functional units. If p functional units are given as input, we prepare just p huddles in which each functional unit is assigned to each huddle and the supply voltage v_h is assigned to each huddle. All the huddles are overlapped with each other where we can ignore interconnection delays between huddles here. $F(h_i)$, $V(h_i)$, and $Dist(h_i, h_j)$ are set to be:

$$F(h_i) = \{f_i\}, \qquad (1 \le i \le p)$$
$$V(h_i) = v_h, \qquad (1 \le i \le p)$$
$$Dist(h_i, h_j) = 0. \qquad (1 \le i, j \le p)$$

After initial huddling, we will start the first iteration.

4.2 Scheduling/Binding

When the supply voltage assigned to operations is changed, the execution timing and the energy consumption of the operation are also changed. When low supply voltage is assigned to an operation, its execution time will increase and its energy consumption will decrease. If an operation execution timing is changed, we may change the operation scheduling and/or operation binding. Our algorithm has five steps (i)–(v) but only Step (i) determines operation execution timings. It is very natural that we assign supply voltages to huddles in the scheduling/binding step and the other steps will be carried out based on supply voltages determined in Step (i).

Then our scheduling/binding problem is, for given a CDFG $G(N, E)$, a clock period constraint T_{clk}, a control step constraint $S_{\max}$, a set of functional units, huddle configuration, an initial supply voltage assigned to each huddle, and huddle placement, to find scheduling and functional unit binding of every node in a given CDFG and to determine supply voltages assigned to given huddles so as to minimize the total energy consumption meeting clock period constraint and control step constraint. Note that, interconnection delays are ignored in the first iteration since floorplanning is not carried out and we assume that all the huddles are overlapped with each other.

Our scheduling/binding is composed of the three phases: initial phase, voltage-increasing phase, and voltage-decreasing phase. In the initial phase, scheduling and binding are executed according to the previous huddle placement and voltages. Since operation binding may be changed, its voltages may be changed in this phase but huddle voltages are not changed. Voltage-increasing phase is executed when the initial phase result does not satisfy the control step constraint and huddle voltages are increased so as to satisfy the control step constraint. Voltage-decreasing phase decreases huddle voltages so as to minimize total energy consumption while meeting the control step constraint.

In order to minimize the energy consumption so as to satisfy control step constraint through the voltage-increasing phase and the voltage-decreasing phase, we design a priority $P_s(h_j)$ for a huddle h_i. We will change the supply voltage of h_i based on its priority. The priority $P_s(h_j)$ is calculated by:

$$P_s(h_j) = \sum_{f_i \in F(h_j)} E(f_i)/D(f_i). \tag{5}$$

According to Ref. [10], $P_s(h_j)$ expresses the energy-efficient effect that are caused by assigning the voltage to h_j. If low voltage can assigned to h_j that has high $P_s(h_j)$, we can gain farther reduction of overall energy consumption.

Initial phase is executed as a first step of scheduling/binding according to huddle placement and voltages obtained by the previous iteration. **Figure 7** shows the initial phase. Basically, we use data-transfer-table based scheduling [12]. If the initial phase result here does not satisfy the control step constraint, we will execute the voltage-increasing phase. Otherwise, we will execute the voltage-decreasing phase.

Voltage-increasing phase is executed when the initial phase result does not satisfy the control step constraint. Th voltage-increasing phase will increase huddle voltages and satisfy the control step constraint. **Figure 8** shows the voltage-increasing phase. We first try to increase the huddle voltage from v_l to v_m to satisfy the control step constraint. At that time we pick up the huddle h_j with the voltage v_l whose priority $P_s(h_j)$ is the smallest first. This is because, even if the supply voltage of h_j is increased to from v_l to v_m, we expect that overall energy consumption is as small as possible satisfying the control step constraint. If the control step constraint is not satisfied when we increase all the huddle voltages from v_l to v_m, we try to increase the huddle voltage from v_m to v_h in a similar way.

Voltage-decreasing phase decreases huddle voltages so as to minimize total energy consumption while meeting the control step constraint. **Figure 9** shows the voltage-decreasing phase. We first try to decrease the huddle voltage from v_h to v_m while meeting the control step constraint. At that time we pick up the huddle h_j with the voltage v_h whose priority $P_s(h_j)$ is the largest first. If we succeed to decrease the huddle voltage for the largest priority huddle, we can drastically decrease the total energy consumption. After that, we try to

[*1] It is not guaranteed that our algorithm always generates converged results even when there exists a feasible solution, but the algorithm has converged in 2–8 iterations in our experimental results in Section 5. Note that other distributed-register architecture synthesis algorithms also have the similar convergence problem above.

© 2012 Information Processing Society of Japan

Initial Phase.

(1) Calculate data transfer table $DT(f_i, h_k)$ for each functional unit f_i and each huddle h_k.
(2) Perform scheduling/binding based on the data transfer table $DT(f_i, h_k)$ [12].
(3) If the result satisfies the control step constraint, perform (a)–(f) so as to minimize energy consumption:
  (a) If there are multipliers with the voltage v_l and v_m,
    (i) Select a multiplication node n which are executed with the voltage v_m.
    (ii) Assume that n is executed with the voltage v_l (not v_m), and perform scheduling/binding based on $DT(f_i, h_k)$ [12] without changing any other operation voltages.
    (iii) If the result satisfies the control step constraint, we accept the result. Otherwise, we execute n with the original voltage v_m.
    (iv) Repeat the above steps until we try all the multiplication nodes executed with v_m.
  (b) If there are multipliers with the voltage v_l and v_h, perform the same steps above for multiplication node executed with the voltage v_h.
  (c) If there are multipliers with the voltage v_m and v_h, perform the same steps above for multiplication node executed with the voltage v_h.
  (d) If there are adders with the voltage v_l and v_m, perform the same steps above for addition node executed with the voltage v_m.
  (e) If there are adders with the voltage v_l and v_h, perform the same steps above for addition node executed with the voltage v_h.
  (f) If there are adders with the voltage v_m and v_h, perform the same steps above for addition node executed with the voltage v_h.

Fig. 7 Initial phase in scheduling/binding.

Voltage-increasing phase

(1) Pick up a huddle h_j with the smallest priority $P_s(h_j)$ among the huddles executed with the voltage v_l. Change its voltage from v_l to v_m.
(2) Perform the initial phase again (Fig. 7).
(3) If the result satisfies the control step constraint, finish.
(4) Repeat the above steps 1–3 until all the huddles with the voltage v_l are tried.
(5) Pick up a huddle h_j with the smallest priority $P_s(h_j)$ among the huddles executed with the voltage v_m. Change its voltage from v_m to v_h.
(6) Perform the initial phase again (Fig. 7).
(7) If the result satisfies the control step constraint, finish.
(8) Repeat the above steps 5–7 until all the huddles with the voltage v_m are tried.

Fig. 8 Voltage-increasing phase in scheduling/binding.

decrease the huddle voltage from v_m to v_l in a similar way.

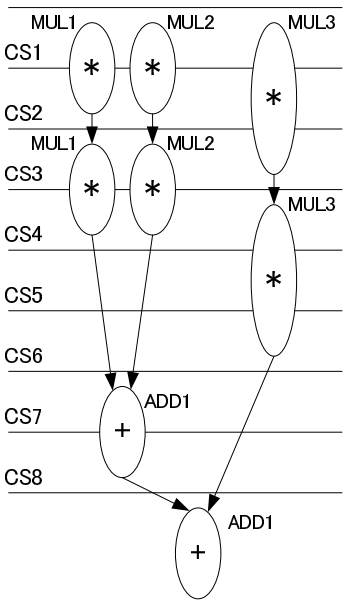
Example 3. *Let us consider a DFG as depicted in* ***Fig. 10*** *(a). Assume that the clock cycle constraint of* $T_{clk} = 3$ ns *and the control step constraint* $S_{\max} = 8$ *are given. Tables 1 and 2 summarize functional unit and level converter specifications. Huddle configurations of Fig. 10 (b) are also given and we assume that the interconnection delays between the three huddles as* $D_w(Dist(h_1, h_3)) = D_w(Dist(h_1, h_2)) = D_w(Dist(h_2, h_3)) = 1$ ns. *Register delays are given by* $D_{reg}(h_1) = D_{reg}(h_2) = D_{reg}(h_3) = 0.5$ ns.

At the initial phase, a data transfer table $DT(f_i, h_k)$ *is constructed first. Figure 10 (c) shows the constructed data transfer table* $DT(f_i, h_k)$ *and the input DFG is scheduled as in Fig. 10 (a). Since Fig. 10 (a) does not satisfy the control step constraint, we execute the voltage-increasing phase next.*

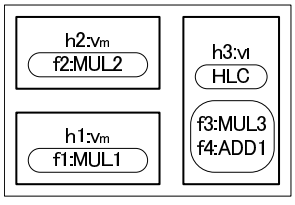
Voltage-decreasing phase

(1) Pick up a huddle h_j with the largest priority $P_s(h_j)$ among the huddles executed with the voltage v_h. Change its voltage from v_h to v_m.
(2) Perform the initial phase again (Fig. 7).
(3) If the result does not satisfy the control step constraint, we assign the original voltage v_h to h_j.
(4) Repeat the above steps 1–3 until all the huddles with the voltage v_h are tried.
(5) Pick up a huddle h_j with the largest priority $P_s(h_j)$ among the huddles executed with the voltage v_m. Change its voltage from v_m to v_l.
(6) Perform the initial phase again (Fig. 7).
(7) If the result does not satisfy the control step constraint, we assign the original voltage v_m to h_j.
(8) Repeat the above steps 5–7 until all the huddles with the voltage v_m are tried.

Fig. 9 Voltage-decreasing phase algorithm.



(a) DFG.



(b) Placement information.

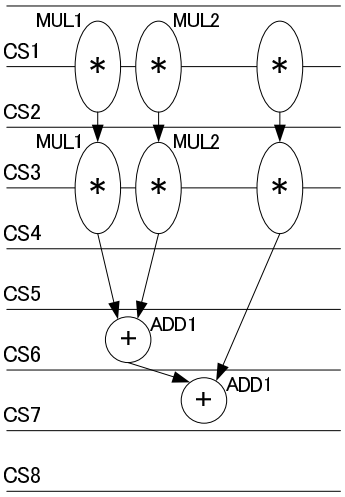
	h1	h2	h3
f1	0	1	2
f2	1	0	2
f3	2	2	0
f4	2	2	0

(c) Data transfer table.

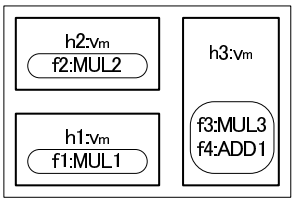
Fig. 10 Inputs of our scheduling/binding.

In the voltage-increasing phase, we pick up the huddle h_3 *in Fig. 10 (b) with the voltage* v_l *having the smallest priority.* ***Figure 11*** *(b) shows the result that the huddle voltage* $V(h_3)$ *is changed from* v_l *to* v_m. *After that,* $DT(f_i, h_k)$ *is re-constructed and the initial phase is executed again. In this case, we have* $DT(f_i, h_k)$ *as in Fig. 11 (c) and we have a scheduling result of Fig. 11 (a). Since Fig. 11 (a) satisfies the control step constraint, we execute the voltage-decreasing phase next.*

© 2012 Information Processing Society of Japan



(a) DFG.



(b) Placement information.

	h1	h2	h3
f1	0	1	1
f2	1	0	1
f3	1	1	0
f4	1	1	0

(c) Data transfer table.

Fig. 11 Outputs of our scheduling/binding.

In the voltage-decreasing phase, we first pick up a huddle with the voltage v_h with the largest priority but there exists no huddles with the voltage v_h in Fig. 11 (b). Then we pick up the huddle h_3 with the voltage v_m having the largest priority and its huddle voltage $V(h_3)$ is changed from v_m to v_l. In this case, we obtain a scheduling result of Fig. 10 (a) when we execute the initial phase again. However, since Fig. 10 (a) does not satisfy the control step constraint, $V(h_3)$ is returned to the original voltage v_m.

In a similar way, the voltage-decreasing phase continues. However, the result satisfying the control step constraint can not be obtained. So we can finally have a result of Fig. 11 (a) satisfying the control step constraint with the smallest energy consumption. □

4.3 Register/Controller Synthesis and Floorplanning

In the register/controller synthesis and floorplanning step, register and controller configuration in each huddle is determined according to the result of a scheduling/binding step and then huddle placement is optimized. The same algorithm with GDR architectures [13] is applied to the register/controller synthesis for HDR architectures. Since we can determine which components are assigned to each huddle, we can also determine the total area of each huddle.

Huddle placement as well as its height and width is optimized by using a simulated annealing (SA) strategy based on a sequence-pair representation [11]. In our floorplanning, power network resources are considered as in Ref. [8]. In SA optimization, its cost function *cost* is expressed by

$$cost = \frac{A_{BB}}{A_{total}} + \alpha\frac{V}{T_{clock}} + \beta\frac{W}{W_{MAX}} + \gamma\frac{A_{PNR}}{A_{total}} \quad (6)$$

where A_{BB} is the rectangle area which includes all the huddles (dead space may be included), A_{total} is the sum of huddles' area (dead space is not included), T_{clock} is the clock period constraint, V is the sum of violations of clock period constraint, W is the wire length, W_{MAX} is the max wire length calculated by (rectangle area's height + width) × the number of wires, and A_{PNR} is the sum of power network resource. α, β and γ are parameters.

The initial solution of floorplan at each iteration is the floorplan solution represented by its sequence-pair of the previous result so that the entire iteration in Fig. 6 can converge gradually. Initial temperature T_i in floorplan at the i-th iteration of the synthesis flow is computed by

$$T_{i+1} = KT_i \quad (7)$$

where K is also a parameter and set to be $K < 1$ [*2].

Note that Step (ii) of register/controller synthesis and floorplanning and Step (iv) of register/controller synthesis and floorplanning in Fig. 6 are completely the same steps.

4.4 Huddling

In huddling, we merge adjacent huddles into a single huddle based on the floorplan result. Since the floorplanning cost is calculated by Eq. (6), huddles that should be merged into a single huddle must be placed close to each other.

In order to determine huddles that should be merged, adjacency $Adj(h_j, h_k)$ for huddles h_j and h_k ($j \neq k$) is defined by:

$$Adj(h_j, h_k) = \left[\frac{H(h_j)}{2} + \frac{H(h_k)}{2}\right] + \left[\frac{W(h_j)}{2} + \frac{W(h_k)}{2}\right] - Dist(h_j, h_k). \quad (8)$$

$Adj(h_j, h_k)$ will be positive when h_j and h_k are adjacent and $Adj(h_j, h_k)$ will be negative when h_j and h_k are placed far away [*3]. **Figure 12** (a) shows the case that the two huddles are adjacent and $Adj(h_j, h_k)$ will be calculated by $Adj(h_j, h_k) \geq 0$. Figure 12 (b) shows the case that the two huddles are not adjacent and $Adj(h_j, h_k)$ will be calculated by $Adj(h_j, h_k) < 0$. $HC(h_j, h_k)$ shows the number of inter-huddle connections between huddles h_j and h_k where $HC(h_j, h_k) \geq 0$.

Then we can define the priority $P_h(h_j, h_k)$ which shows whether huddle h_j and h_k should be merged or not:

$$P_h(h_j, h_k) = Adj(h_j, h_k) \cdot HC(h_j, h_k). \quad (9)$$

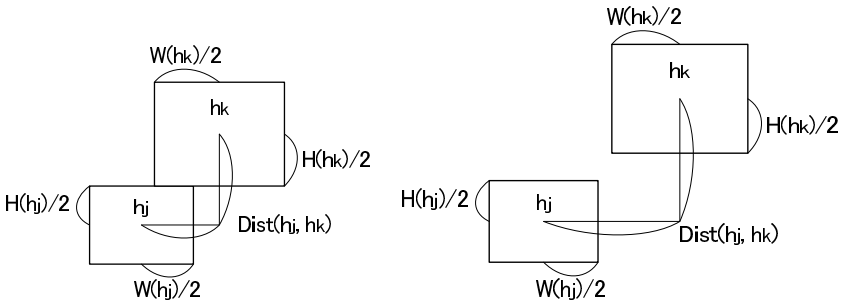
In a similar way, $P_h(h_j, h_k)$ will be positive when h_j and h_k are adjacent and $P_h(h_j, h_k)$ will be negative when h_j and h_k are placed far away. We first pick a pair of huddles whose P_h value is the largest and check whether this pair of huddles satisfy the merging condition. When they satisfy the merging condition, they are

[*2] In our experiments, we set $\alpha = 100, \beta = 1, \gamma = 0.5$ and $K = 0.9$.

[*3] $Adj(h_j, h_k)$ can be positive even if the two huddles h_j and h_k are not adjacent, but we can say that the two huddles are close enough if $Adj(h_j, h_k)$ is positive. This is because of the following reason:

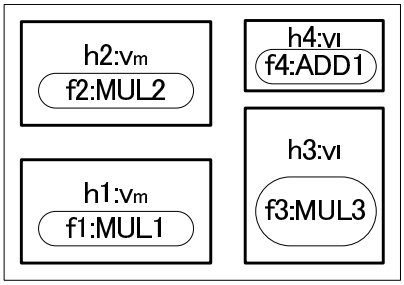
The merging priority $P_h(h_j, h_k)$ in Eq. (9) should represent the amount $HC(h_j, h_k)$ of data transfers in each combination of huddles. However, two huddles which are placed far away should not be merged into a single huddle since a floorplanning result indicates some optimal situation. Therefore we also require the criterion how much close the two huddles are placed. Thus we define $Adj(h_j, h_k)$ as in Eq. (8). In fact, our huddling works well in our experiments in Section 5.

© 2012 Information Processing Society of Japan



(a) Huddles h_j and h_k are adjacent. (b) Huddles h_j and h_k are not adjacent.

Fig. 12 An example of adjacency $Adj(h_j, h_k)$.

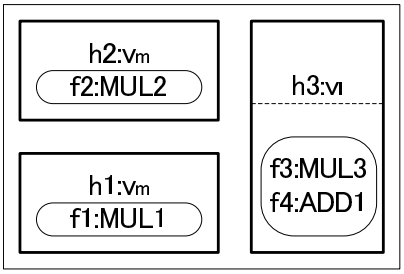


(a) Placement.

	h1	h2	h3	h4
h1		15	10	−10
h2	15		5	10
h3	10	5		15
h4	−10	10	15	

(b) Priority $P_h(h_j, h_k)$ between the two huddles

Fig. 13 Inputs of huddling.



(a) Placement.

	h1	h2	h3
h1		15	10
h2	15		10
h3	10	10	

(b) Priority $P_h(h_j, h_k)$ between the two huddles

Fig. 14 Outputs of huddling.

merged into a single huddle. The merging condition here is defined by

$$\begin{cases} V(h_j) = V(h_k) \text{ and} \\ \\ h_j \text{ and } h_k \text{ satisfy the huddle size constraint.} \end{cases} \tag{10}$$

We continue to find a pair of huddles that satisfy the merging condition until no pairs of huddles satisfy the merging condition.

In our huddling, we do not consider overlapping of existing huddles and merged huddles and all pairs of huddles satisfying the merging condition are merged. This overlapping will be resolved at Step (iv) of register/controller synthesis and floorplanning. By introducing this approach, we can have as small number of huddles as possible and will have a floorplanning result consistent with Step (i) of scheduling/binding.

Example 4. ***Figure 13*** *shows an example of huddling. The huddle pair of h_3 and h_4 are picked up since they have the maximum priority $P_h(h_3, h_4) = 15$. Since $V(h_3) = V(h_4) = v_l$ and they also satisfy the huddle size constraint, they satisfy the merging condition. Then h_3 and h_4 are merged into a single huddle h_3.*

We check other pairs of huddles, but h_1 and h_2 do not satisfy the huddle size constraint and other pairs of huddles have different supply voltages.

Overall, we can finally have a new huddle configuration as in ***Fig. 14*** *(a).* □

4.5 Unhuddling

In Section 4.4, we have proposed a huddling step which merges several huddles into a single huddle, but a synthesis solution may fall into a local minimum if we only deal with huddling. We need an unhuddling step which partitions a single huddle into several huddles.

In unhuddling, we also utilize huddle placement information. Let $DT_s(f_i, h_k)$ and $DT_f(f_i, h_k)$ be data transfer tables for a functional unit f_i and a huddle h_k, just after (i) scheduling/binding step in the current iteration and just after (iv) register/controller synthesis and floorplanning step in the current iteration, respectively. Then we check whether the following equation holds true or not:

© 2012 Information Processing Society of Japan

$$DT_s(f_i, h_k) < DT_f(f_i, h_k). \quad (11)$$

If Eq. (11) holds, a data transfer delay from f_i to h_k may violate the given clock period constraint. In this case, we can consider that assigning f_i to huddle $h_j(= Hud(f_i))$ may cause this violation. We eliminate f_i from h_j, construct a new huddle h_l, and assign f_i to the new huddle h_l. h_l is placed so as to overlap h_k. In (ii) register/controller synthesis and floorplanning step of the next iteration, registers and controller for the new huddle h_l will be constructed and the overlap of huddles will be resolved.

If no pairs of huddles satisfy Eq. (11), this means that the current floorplanning satisfies the scheduling/binding result. Then we will finish the the iterative improvement loop.

5. Experimental Results

We have implemented the proposed algorithm in C++. The algorithm has been applied to DCT (48 nodes), Jacobi (48 nodes), EWF3 (102 nodes), and FIR filter (75 nodes). **Table 3** shows the functional unit specification and **Table 4** shows the level converter specification [18]. All the functional units were assumed to have a bit width of 16, and their specifications were obtained by synthesizing them beforehand based on the CMOS 90 nm technology. Controllers were synthesized by Synopsys Design Compiler in each iteration. The interconnection delays were assumed to be a proportion to square of the wiring length and an interconnection delay is set to be 1 ns when wiring length is 250 μm [13] [*4]. Energy consumption is obtained using Synopsys Design Compiler.

We compare our proposed algorithm targeting HDR architectures with multiple supply voltages ("MHDR" in **Table 5**) to a GDR architecture synthesis algorithm [13] ("GDR" in Table 5), MCAS for RDR architectures [2] ("RDR" in Table 5), and our proposed algorithm targeting HDR architecture with a single supply voltage ("HDR" in Table 5). We further compare our algorithm with the following strategy: we first perform the existing multiple supply voltage aware scheduling [20]; Based on this voltage assignment to each operation, we perform MCAS for RDR architecture ("Ref. [20] + RDR" in Table 5), and perform our algorithm targeting HDR architectures ("Ref. [20] + HDR" in Table 5). The clock period constraint was given to be 2.5 ns in all the experiments.

Table 5 summarizes the experimental results. In Table 5 "CS constraints" shows the control step constraint. "Control steps" shows the number of required control steps after synthesizing each circuit. "Area" and "Rectangular area" in Table 5 represent the sum of module/huddle area and the minimum rectangle area including all of them. "Dynamic energy" and "Leak energy" represent dynamic energy consumption and leakage energy consumption. "All Energy" shows the sum of "Dynamic energy" and "Leak energy." "Iterations" shows the number of iterations requred by each algorithm. "CPU Time" shows CPU time to synthesize each circuit.

Table 3 Functional unit specification.

Functional Unit	Area [μm^2]	Delay [ns]	Dynamic energy [fJ]	Leak power [μW]
Adder (1.2 V)	386	0.75	92	3.9
Adder (1.0 V)	386	1.22	64	3.2
Adder (0.8 V)	386	2.71	41	2.6
Subtractor (1.2 V)	417	0.78	97	4.2
Subtractor (1.0 V)	417	1.27	67	3.5
Subtractor (0.8 V)	417	2.82	43	2.8
Multiplier (1.2 V)	2,161	1.65	1,135	19.8
Multiplier (1.0 V)	2,161	2.70	788	16.5
Multiplier (0.8 V)	2,161	6.00	504	13.2
Divider (1.2 V)	6,066	6.25	2,306	837.6
Divider (1.0 V)	6,066	10.21	1,601	698.0
Divider (0.8 V)	6,066	22.69	1,234	524.0

Table 4 Level converters specification [18].

V_{in} - V_{out}	Area [μm^2]	Delay [ns]	Dynamic energy [fJ]	Leak power [μW]
1.2 V - 1.0 V	113	0.17	83	49.1
1.2 V - 0.8 V	113	0.22	71	32.3
1.0 V - 1.2 V	113	0.17	76	45.0
1.0 V - 0.8 V	113	0.30	55	18.3
0.8 V - 1.2 V	113	0.22	86	39.1
0.8 V - 1.0 V	113	0.30	55	18.3

The experimental results show that the smallest area is "GDR," HDRs ("HDR," "Ref. [20] + HDR," "MHDR"), and RDRs ("RDR," "Ref. [20] + RDR") in that order. However, "GDR" areas can be sometimes become larger than "HDR" areas. This is because of the following reason: "GDR" has shared register groups but, since its synthesis flow is too complicated as pointed out in Section 2, several registers cannot be shared into any shared register group but become local regsters in order to meet the timing constratins. On the other hand, our "HDR" has a strucutre of huddles and all the regsters in each huddle must be shared into shared regsters.

Between the areas considering single supply voltage and those considering multiple supply voltages, the latter will be larger in most cases. This is because the level converter area must be added and the results considering multiple supply voltages decrease register sharing. However, "MHDR" areas can be sometimes become smaller than "HDR" areas. This is just because of our proposed algorithm cannot always have an optimal (or semi-optimal) result. Since our proposed algorithm is based on an iterative improvement flow, it sometimes fall into an local optima. In the case of EWF3 applied to HDR, it is just the case.

The experimental results show that the dynamic energy consumption of our proposed algorithm "MHDR" is reduced by a maximum of 48.2% and an average of 25.2% compared with the other algorithms. All energy consumption of "MHDR" is also reduced by a maximum of 48.1% and an average of 22.4% compared with the other algorithms. The leakage energy consumption of "MHDR"is reduced by a maximum of 60.3% , but is increased

[*4] In Ref. [13], 1 ns per 250 μm is used but it may be too large for CMOS 90 nm technology. In Ref. [13], they use the values in ITRS '05 [4] and obtain the ratio between gate delay and interconnection delay in CMOS 45 nm technology. Then they apply this ratio to CMOS 90 nm technology and obtain converted interconnection delays as 1 ns per 250 μm. This is because multi-cycle interconnect communication used in RDR and GDR architectures is required in technology nodes finer than 65 nm technology. Since we also use CMOS 90 nm technology here, we use this converted interconnection delays.

Note that, when the technology node is changed, the absolute energy itself must be changed but we can have the similar energy decreasing trend and at least we must have almost the same decreasing ratio in dynamic energy in Table 5.

© 2012 Information Processing Society of Japan

Table 5 Experimental results.

App.	FUs	CS constraints	Architechture	Control steps	Area [μm^2]	Rectangular area [μm^2]	Dynamic energy [pJ]	Leak energy [pJ]	All energy [pJ]	Iterations	CPU time [sec]
ewf3	Add×3	50	GDR	43	47,792	55,250	655.83	57.64	713.47	7	362.48
	Mul×2		RDR	44	69,530	69,530	764.33	85.06	849.39	1	56.21
			HDR	43	53,926	59,706	720.82	79.38	800.21	6	1,169.40
			Ref. [20] + RDR	50	109,350	109,350	570.41	84.57	654.98	1	127.98
			Ref. [20] + HDR	50	57,118	60,882	577.89	108.90	686.80	8	1,815.60
			MHDR	50	44,049	48,208	520.20	91.07	611.27	2	487.20
fir	Add×3	35	GDR	30	36,840	48,278	429.16	38.35	467.51	24	1,212.06
	Mul×3		RDR	29	81,920	81,920	360.01	105.18	465.19	1	75.78
			HDR	30	28,493	32,643	344.88	30.75	375.62	2	353.15
			Ref. [20] + RDR	35	115,200	115,200	507.73	64.94	572.67	1	174.33
			Ref. [20] + HDR	35	51,795	59,220	371.74	75.42	447.17	2	484.67
			MHDR	35	40,231	49,580	284.64	44.63	329.27	2	484.10
fir	Add×4	30	GDR	30	39,407	42,593	473.44	40.34	513.78	24	1,314.37
	Mul×4		RDR	29	82,816	82,816	335.95	123.57	459.52	1	75.59
			HDR	30	34,967	41,087	315.10	38.79	353.89	5	701.88
			Ref. [20] + RDR	30	129,600	129,600	366.16	43.22	409.39	1	190.94
			Ref. [20] + HDR	30	57,672	66,316	475.25	94.20	569.46	2	352.27
			MHDR	30	48,011	59,175	246.41	49.09	295.49	2	576.32
jacobi	Add×2	20	GDR	19	28,026	33,660	273.75	60.70	334.45	8	644.76
	Sub×1		RDR	20	57,600	57,600	224.93	94.94	319.87	1	138.06
	Mul×2		HDR	19	32,031	34,686	201.13	91.78	292.91	2	288.77
	Div×2		Ref. [20] + RDR	20	115,200	115,200	224.32	119.43	343.74	1	144.62
			Ref. [20] + HDR	20	35,124	38,340	163.21	100.32	263.52	2	448.74
			MHDR	20	36,581	42,210	163.56	93.41	256.98	2	447.97
dct	Add×4	10	GDR	8	53,864	58,378	208.48	11.00	219.48	24	1,378.30
	Mul×4		RDR	9	81,476	81,476	220.75	13.69	234.45	1	74.29
			HDR	8	55,709	58,450	196.58	14.66	211.24	2	505.71
			Ref. [20] + RDR	10	115,200	115,200	235.79	52.98	288.76	1	194.40
			Ref. [20] + HDR	10	42,272	44,544	202.21	22.56	224.77	3	746.99
			MHDR	10	50,337	69,030	169.16	25.53	194.69	3	822.64

by an average 9.4% compared with the other algorithms. This is because level converters increase the leakage energy, but the overall energy consumpation is much reduced compared with other algorithms.

Note that the objectives of synthesis algorithms of "GDR," "RDR" and "HDR" are to minimize the required control steps. Actually, their control steps are shorter than the control step constraints. All the energies in Table 5 are evaluated within the required control steps. These results can be fairly compared to results obtained by "Ref. [20] + RDR," "Ref. [20] + HDR" and "MHDR." The number of iterations in "MHDR" is up to three and we can have CPU time comparable to the GDR synthesis algorithm.

6. Conclusions

In this paper, we proposed huddle-based distributed register architectures (HDR architectures) for multi-cycle interconnect communications and a new energy-efficient high-level synthesis algorithm targeting HDR architectures. Our proposed algorithm reduced energy consumption by a maximum of 48.1% and by an average of 22.4% compared with the conventional algorithms.

In the future, we will apply other energy-saving techniques such as power gating and clock gating to our HDR architectures. We also re-consider the convergence problem in our algorithm and will develop an improved version.

Acknowledgments This research was supported in part by NEDO and KDDI Foundation.

References

[1] Chang, J. and Pedram, M.: Energy minimization using multiple supply voltages, *IEEE Trans. VLSI Systems*, Vol.5, No.4, pp.436–443 (1997).
[2] Cong, J., Fan, Y., Han, G., Yang, X. and Zhang, Z.: Architecture and synthesis for on-chip multicycle communication, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.23, No.4, pp.550–564 (2004).
[3] Cong, J., Fan, Y. and Xu, J.: Simultaneous resource binding and interconnection optimization based on a distributed register-file microarchitecture, *ACM Trans. Design Automation of Electronic Systems*, Vol.14, No.3, pp.1–31 (2009).
[4] International technology roadmap for semiconductors (ITRS) (2005), available from ⟨http://www/itrs/net/⟩.
[5] Jeon, J., Kim, D., Shin, D. and Choi, K.: High-level synthesis under multi-cycle interconnect delay, *Proc. ASP-DAC '01*, pp.662–667 (2001).
[6] Johnson, M. and Roy, K.: Datapath scheduling with multiple supply voltages and level converters, *ACM Trans. Design Automation of Electronic Systems*, Vol.2, No.3, pp.227–248 (1997).
[7] Kim, D., Jung, J., Lee, S., Jeon, J. and Choi, K.: Behavior-to-placed RTL synthesis with performance-driven placement, *Proc. ICCAD '01*, pp.320–325 (2001).
[8] Lee, W., Liu, H. and Chang, Y.: Voltage island aware floorplanning for power and timing optimization, *Proc. ICCAD '06*, pp.389–394 (2006).
[9] Lin, Y., Hwang, C. and Wu, A.: Scheduling techniques for variable voltage low power designs, *ACM Trans. Design Automation of Electronic Systems*, Vol.2, No.2, pp.81–97 (1997).
[10] Manzak, A. and Chakrabarti, C.: A low power scheduling scheme with resources operating at multiple voltages, *IEEE Trans. VLSI Systems*, Vol.10, No.1, pp.6–14 (2002).
[11] Murata, H., Fujiyoshi, K., Nakatake, S. and Kajitani, Y.: VLSI module placement based on rectangle-packing by the sequence-pair, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.15, No.12, pp.1518–1524 (1996).
[12] Ohchi, A., Togawa, N., Yanagisawa, M. and Ohtsuki, T.: High-level synthesis algorithms with floorplaning for distributed/shared-register architectures, *Proc. VLSI-DAT 2008*, pp.164–167 (2008).
[13] Ohchi, A., Togawa, N., Yanagisawa, M. and Ohtsuki, T.: Floorplan-aware high-level synthesis for generalized distributed-register architectures, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.92, No.12, pp.3169–3179 (2009).

© 2012 Information Processing Society of Japan

[14] Raje, S. and Sarrafzadeh, M.: Variable voltage scheduling, *Proc. ISLPED '95*, pp.9–14 (1995).
[15] Sarrafzadeh, M. and Raje, S.: Scheduling with multiple voltages under resource constraints, *Proc. ISCAS 1999*, Vol.1, pp.350–353 (1999).
[16] Shiue, W. and Chakrabarti, C.: Low-power scheduling with resources operating at multiple voltages, *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, Vol.47, No.6, pp.536–543 (2000).
[17] Tanaka, S., Yanagisawa, M., Ohtsuki, T. and Togawa, N.: A Fault-Secure High-Level Synthesis Algorithm for RDR Architectures, *IPSJ Trans. System LSI Design Methodology*, Vol.4, pp.150–165 (2011).
[18] Tran, C., Kawaguchi, H. and Sakurai, T.: Low-power high-speed level shifter design for block-level dynamic voltage scaling environment, *Proc. ICICDT 2005*, pp.229–232 (2005).
[19] Yang, H. and Dung, L.: On multiple-voltage high-level synthesis using algorithmic transformations, *Proc. ASP-DAC '05*, pp.872–876 (2005).
[20] Yang, H. and Dung, L.: Algorithmic transformations and peak power constraint applied to multiple-voltage low-power VLSI signal processing, *WSEAS Trans. Signal Processing*, Vol.3, No.12, pp.479–486 (2007).

Shin-ya Abe received his B.Eng. and M.Eng. from Waseda University in 2011 and 2012. He is presently working toward Dr.Eng. degree there. His research interests include design and verification of VLSI, especially high-level synthesis and energy efficiency design. He is a student member of IEICE.

Masao Yanagisawa received his B.Eng., M.Eng., and Dr.Eng. degree from Waseda University in 1981, 1983 and 1986, respectively, all in electrical engineering. He was with University of California, Berkeley from 1986 through 1987. In 1987, he joined Takushoku University. In 1991, he left Takushoku University and joined Waseda University, where he is presently a Professor in the Department of Electronic and Photonic Systems. His research interests are combinatorics and graph theory, computational geometry, VLSI design and verification, and network analysis and design. He is a fellow of IEICE and a member of IEEE and ACM.

Nozomu Togawa received his B.Eng., M.Eng., and Dr.Eng. degree from Waseda University in 1992, 1994 and 1997, respectively, all in electrical engineering. He is presently a Professor in the Department of Computer Science and Engineering, Waseda University. His research interests are VLSI design, graph theory, and computational geometry. He is a member of IEEE and IEICE.

(Recommended by Associate Editor: *Takashi Takenaka*)

© 2012 Information Processing Society of Japan