IPSJ Transactions on System LSI Design Methodology Vol.7 74–80 (Aug. 2014)

[DOI: 10.2197/ipsjtsldm.7.74]

Short Paper

Energy-efficient High-level Synthesis for HDR Architecture with Multi-stage Clock Gating

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

Received: December 5, 2013, Revised: February 27, 2014,
Accepted: April 28, 2014, Released: August 4, 2014

Abstract: With the miniaturization and high performance of current and future LSIs, demand for portable devices has much more increased. Especially the problems of battery runtime and device overheating have occurred. In addition, with the downsize of the LSI design process, the ratio of an interconnection delay to a gate delay has continued to increase. High-level synthesis to estimate the interconnection delays and reduce energy consumption is essential. In this paper, we propose a high-level synthesis algorithm based on HDR architectures (huddle-based distributed register architectures) utilizing multi-stage clock gating. By increasing the number of clock gating stages in each huddle, we increase the number of the control steps at which we can apply the clock gating to registers. We can determine the configuration of the clock gating with optimized energy consumption. The experimental results demonstrate that our proposed algorithm reduced energy consumption by up to 27.7% compared with conventional algorithms.

Keywords: high-level synthesis, huddle-based distributed register architecture, multi-stage clock gating, clock gating timing, gating step count

1. Introduction

With the miniaturization of LSIs and their increasing performance, demand for high-functional portable devices has grown significantly. At the same time, battery lifetime and device overheating are leading to major design problems hampering further LSI integration. On the other hand, the ratio of an interconnection delay to a gate delay has continued to increase as device feature size decreases. We have to estimate interconnection delays and reduce energy consumption even in a high-level synthesis stage.

In order to tackle this problem, Abe et al. have proposed huddle-based distributed-register (HDR) architectures and its associated high-level synthesis algorithm [1]. Akasaka et al. have proposed high-level synthesis for HDR architectures with clock gating based on concurrency-oriented scheduling/functional unit binding [3]. In HDR architectures, functional units, registers, and a controller located close to each other are grouped as a *huddle* and then we can easily estimate interconnection delays. The algorithm [3] assumes coarse-grained clock gating to huddles and focuses on the number of control steps, or *gating steps*, at which it can apply the clock gating to registers in every huddle. Two methods are proposed to increase gating steps: One is that it schedules operations to be performed at the same timing. By adjusting the clock gating timings in a high-level synthesis stage, it can enhance the effect of clock gating more than applying clock gating after logic synthesis. The other one is that it synthesizes huddles such that each of the synthesized huddles includes registers which have similar or the same clock gating timings. However, the algorithm [3] sets at most one type of clock gating per huddle. It is quite necessary to increase the number of clock gating stages in each huddle to further reduce energy consumption.

Based on the discussion above, we propose in this paper a high-level synthesis algorithm based on HDR architectures utilizing the multi-stage clock gating. We configure three-stage clock gating and set one clock gating circuit per stage during HDR synthesis. By increasing the number of clock gating stages in each huddle, we also increase gating steps. Furthermore, we reduce all energy consumption including register energy as well as clock tree energy. The experimental results demonstrate that our proposed algorithm reduced all energy consumption by up to 27.7% compared with conventional algorithms.

2. Target Architecture and Problem Definition

2.1 Huddle-based Distributed-register Architecture

We use an HDR architecture as our target architecture [1]. HDR is an architecture that introduces huddles into distributed-register architectures [4], [6] and abstracts each module inside an LSI chip. A huddle has a rectangular shape within a range determined by the clock cycle and share functional units, registers, a controller, and level converters in its inside. Since huddles are rectangle but not regular, we can achieve small area by packing them effectively.

Figure 1 shows a huddle consisting of the following components:

- Huddled Local Register (HLR)
  Set of local registers and multiplexers dedicated to each huddle.

[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)] hiroyuki.akasaka@togawa.cs.waseda.ac.jp
[b)] togawa@togawa.cs.waseda.ac.jp

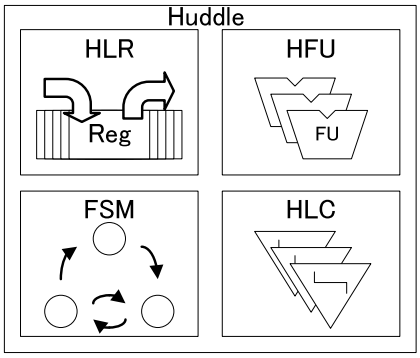


Fig. 1 Huddle.

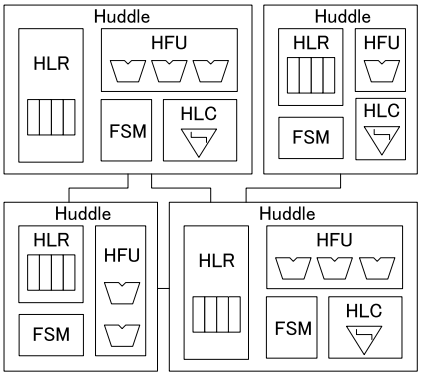


Fig. 2 HDR architecture.

- Huddled Functional Unit (HFU)
  Set of functional units collected in each huddle. HFU mainly accesses the HLR in its huddle.
- Finite State Machine (FSM)
  Controller dedicated to each huddle. It controls the HFU and HLR in its huddle.
- Huddled Level Converter (HLC)
  Set of level converters collected in each huddle. It is used to transfer a data to different-voltage huddles

Figure 2 shows an HDR architecture composed of several huddles. If we communicate data inside each huddle, data transfer time can be ignored, i.e., it can be done in a single clock cycle. If we communicate data between two huddles, multi-cycle data communication between these huddles will be done. HLC is used when the voltages of the two huddles are different. By introducing HDR architectures into LSI chip design, we can very easily estimate interconnection delays and thus we can have high-level synthesis considering them.

2.2 Clock Gating and Problem Definition

There are two types of clock gating: One is fine-grained clock gating which cuts off the clock signal to registers one by one. The other one is coarse-grained clock gating which cuts off the clock signal to register groups [8].

Fine-grained clock gating has an advantage that each register does not consume extra energy, since we determine the clock gating timing considering the active timing of every register. But fine-grained clock gating must be done at the *leaf side* of a clock tree. Coarse-grained clock gating has a disadvantage that each register may consume extra energy, since we determine the clock gating timing considering the active timing of some register groups. But coarse-grained clock gating can be done at the *root side* of a clock tree. When we focus on a clock tree itself, it consumes energy caused by its drivers and buffers inserted for adjusting the skew. If we apply clock gating at its root side, low-energy clock tree can be synthesized [7].

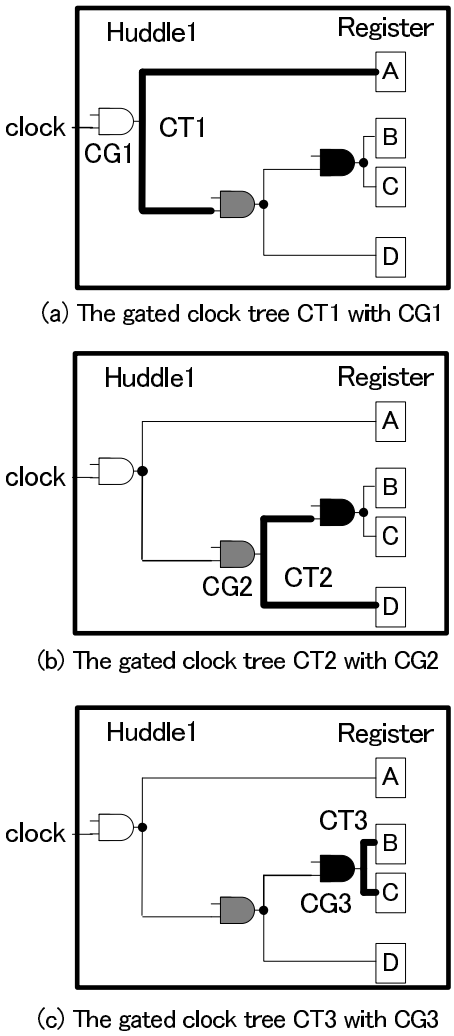


Fig. 3 Three-stage clock tree in a huddle.

Now we consider that we apply the multi-stage clock gating which has both the advantages of fine-grained/course-grained clock gating. Particularly in this paper, we focus on three-stage clock gating, since it gives better trade-off between fine-grained clock gating and course-grained clock gating [5]. Clock tree is composed of the upper clock tree from the clock terminal of a chip to huddles and lower clock trees from a huddle edge to registers [3]. **Figure 3** shows an example of lower clock trees in a huddle for three-stage clock gating. Let CG1, CG2, and CG3 be the three clock gating circuits. Similarly, let CT1, CT2, and CT3 be the gated clock trees associated with CG1, CG2 and CG3, respectively. The sinks in each clock tree are sink registers and the next-stage clock gating circuit. At the control steps when CG1 cuts off the clock signal, we can reduce the dynamic power of CT1, CT2, and CT3. At the control steps when CG2 cuts off the clock signal, we can reduce the dynamic power of CT2 and CT3. At the control steps when CG3 cuts off the clock signal, we can reduce the dynamic power of CT3.

Since our coarse-grained clock gating proposed in Ref. [3] has only one clock gating circuit in each huddle, it cannot cut off the clock signal effectively. On the other hand, three-stage clock gating above can have three clock gating circuits in each huddle. We expect it can cut off the clock signal to more registers and then reduce energy consumption.

Then our high-level synthesis problem is defined as follows:

Definition 1. *Our high-level synthesis problem is, for given a control-data flow graph (CDFG), a clock period constraint, a latency 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, to assign a supply voltage to each huddle, to apply three-stage clock gating 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.*

3. Energy-efficient High-level Synthesis for HDR Architecture with Multi-stage Clock Gating

Based on our previous algorithm [3], **Fig. 4** shows our high-level synthesis flow composed of Initial huddling, Concurrency-oriented Scheduling/FU binding, Register binding, Controller synthesis, Floorplanning, Multi-stage CG timing calculation, Multi-stage CG huddling, and Unhuddling. By repeatedly performing these steps, we finally have an energy-saving high-level synthesis result with floorplanning. Huddle placement is represented by a sequence-pair and it is fed back to the next iteration as the initial floorplanning. Interconnection delay is represented by a delay table which shows a delay necessary for the communication between modules. By using this delay table obtained by the previous iteration, we have a scheduling result reflecting the previous huddle placement in the scheduling step at the current iteration. Note that, we require approximately 10 iterations in our algorithm as well as Ref. [3]. We consider that our iteration-based algorithm is reasonable enough.

We can use the exactly the same algorithms as Ref. [3] other than "Multi-stage CG timing calculation" and "Multi-stage CG huddling" and then we propose here each of them.

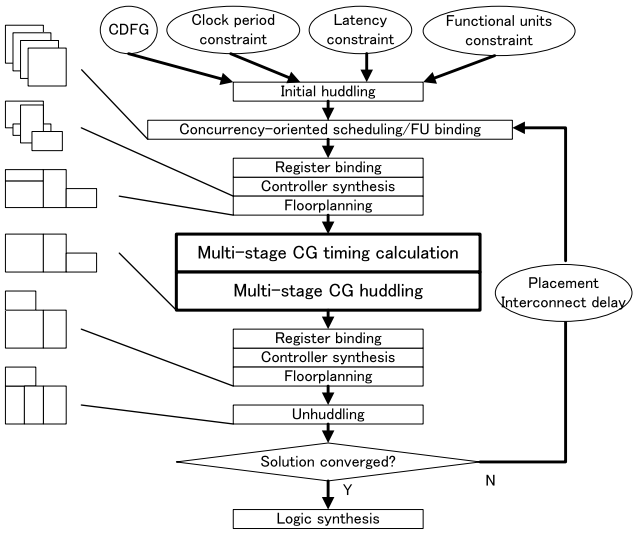


Fig. 4 Proposed high-level synthesis flow.

3.1 Multi-stage CG Timing Calculation

"Multi-stage CG timing calculation" assigns three-stage CG timings to each huddle based on the active timings of registers and the clock tree energy. Since our lower clock trees have three-stage clock gating, its configuration can be too complex to perform exhaustive search. Then, we employ a strategy to determine its configuration in a stage-by-stage manner.

Figure 5 shows an example of three-stage CG timing calculation in Huddle 1. **Figure 6** shows clock tree configurations associated with Fig. 5. "✓" in Fig. 5 shows the control step where the register is used. We can cut off the clock signal at the control step where "✓" is not marked.

3.1.1 First Stage Clock Tree Configuration

First, we determine the first-stage CG timing in Huddle 1 (Fig. 5 (a)). Assume that we determine the first stage CG timing as depicted in Figs. 5 (a) and 6 (a). "CG1" in Fig. 6 (a) shows the clock gating circuit in the first stage. The white circle in Fig. 5 (a) shows the control step when CG1 cuts off the clock signal. For determining the CG timing, we define the cost function $Cost(R)$ of $R = \{r_1, \cdots, r_k\}$ of k registers in Huddle 1. Let $Step(R)$ be the gating step count, i.e., the number of control steps that we can cut off the clock signal (In Fig. 5, $Step(R)$ is the number of white, gray and black circles). Let E_{step} be the energy consumption of a register in one step. Let $E_u(R)$ be the energy consumption of the upper clock tree for Huddle 1 when we apply clock gating to the register subset R in Huddle 1. Let $E_d(R)$ be the energy consumption of the lower clock trees in Huddle 1 when we apply clock gating to the register subset R in Huddle 1. Then the cost function $Cost(R)$ can be defined by:

$$Cost(R) = (E_u(R) + E_d(R)) - E_{step} \times Step(R) \quad (1)$$

where $E_u(R)$ and $E_d(R)$ can be obtained by using the equations in Ref. [9]. The power consumption of the clock tree P in Ref. [9] is defined by:

$$P = 1.5 f V^2 (C_0 W + C_s) + P_0 \sum_{buffers} k_i \quad (2)$$

where f is the frequency, V is the voltage, C_0 is the capacitance per unit length, W is the wire length, C_s is the total sink load capacitance, P_0 is the static power dissipated by a minimum size buffer, *buffers* is the number of clock buffers, and k_i is the size of the i-th buffer. Based on each huddle size and huddle voltage, we estimate these parameters and set them. $E_d(R)$ is directly dependent on the lower clock tree configuration in h and calculated mainly based on how many lower clock trees are required inside h and how many sinks each lower clock tree requires. $E_u(R)$ is calculated mainly based on the number of upper clock sinks and their positions, which are dependent on lower clock tree configurations [*1]. Sinks of the upper clock tree are all the lower clock

[*1] We can show how lower clock tree configuration affects the upper clock sinks using Fig. 6. Huddle 1 in Fig. 6 (a) has two sinks for the upper clock tree, one is directly connected to Register A and the other is connected to Registers B, C, and D through CG1. In Fig. 6 (b), Huddle 1 has only one sink for the upper clock tree. Based on the number of upper clock sinks and other parameters, we can calculate $E_u(R)$ using Eq. (2).

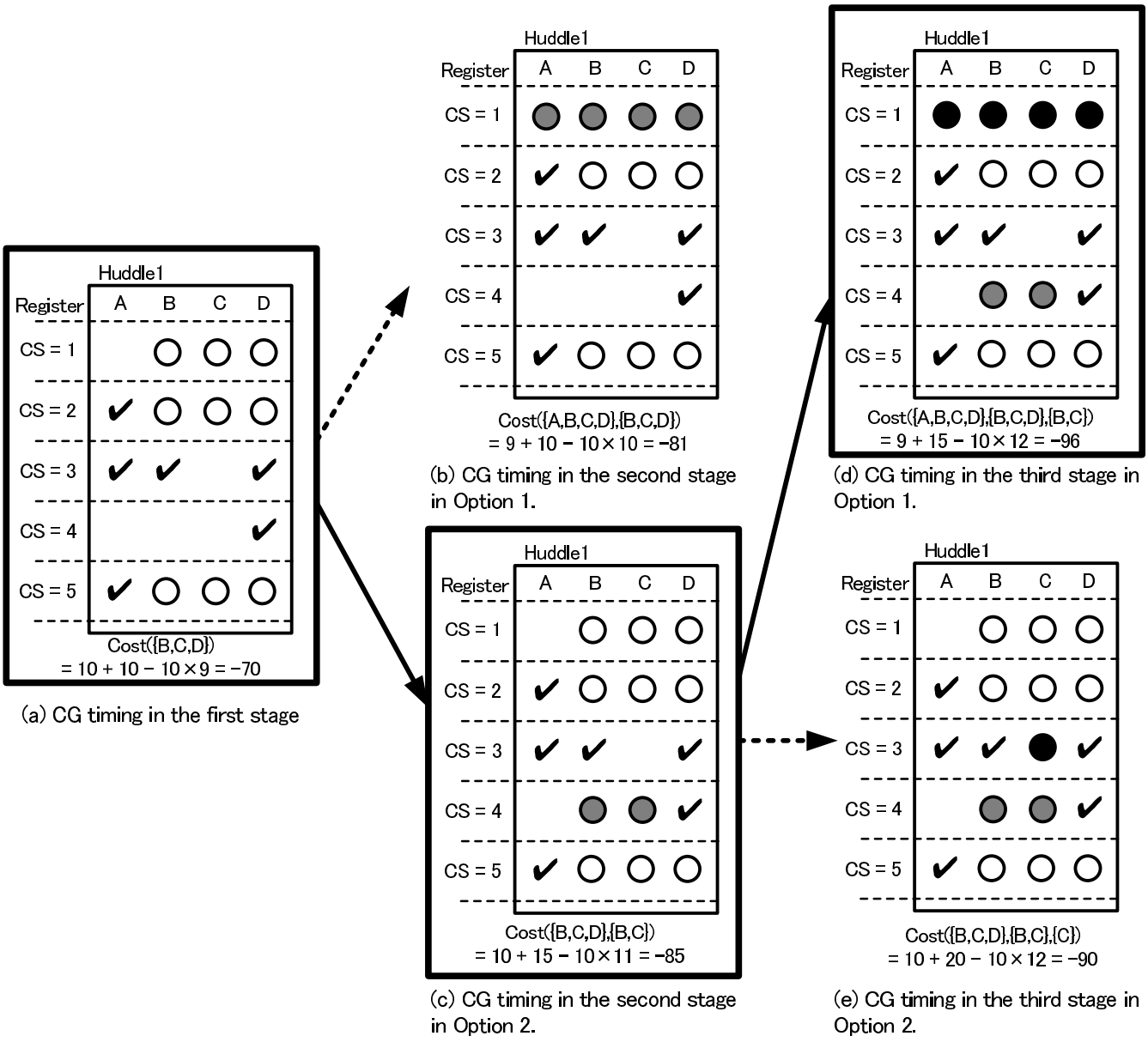


Fig. 5 Three-stage clock gating timing calculation in each huddle.

trees over all the huddles in a chip. In each huddle other than Huddle 1, we use a lower clock tree configuration determined in the previous iteration or just before at this iteration to calculate $E_u(R)$. By considering all possible register subset R in Huddle 1 and calculating $Cost(R)$, we pick up the one which gives the smallest $Cost(R)$ value and determine the first stage CG timing accordingly.

In Fig. 5, we assume that $E_{step} = 10$ and $E_u(R) = 10$ for any subset R in Huddle 1. $E_d(R) = 10$ when we use both a gated lower clock tree and a non-gated lower clock tree in Huddle 1. $E_d(R) = 5$ when we use only a non-gated lower clock tree in Huddle 1. Since $Cost(\{B, C, D\})$ gives the minimum when we consider the single-stage clock tree configuration, we will apply clock gating to Registers B, C and D in the first stage of Huddle 1. In this case, $Step(\{B, C, D\}) = 9$ since the number of white circles in Fig. 5 (a) is nine. We will cut off the clock signal to Registers B, C and D at CS1, CS2, and CS5 in the first stage.

3.1.2 Second Stage Clock Tree Configuration

Next, we determine the second-stage CG timing in Huddle 1. When we add one more clock gating circuit to Fig. 6 (a), we can have the two options to increase the gating step count:

Option 1: Try to increase the number of clock-gated registers.

Option 2: Try not to change the number of clock-gated registers but increase the gating steps of current clock-gated registers.

If we apply clock gating to all the registers in Huddle 1 in the first stage, we cannot further apply Option 1 to this clock tree. If we apply clock gating to only one register in Huddle in the first stage, we cannot further apply Option 2 to this clock tree. We must try to apply both Option 1 and Option 2 to the single-stage clock tree obtained by the first stage clock tree configuration above and pick up the better one.

Figures 5 (b) and (c) show the CG timing in the second stage when we apply Option 1 and Option 2 to the single-stage clock tree, respectively. "CG2" in Figs. 6 (b) and (c) show the clock gat-

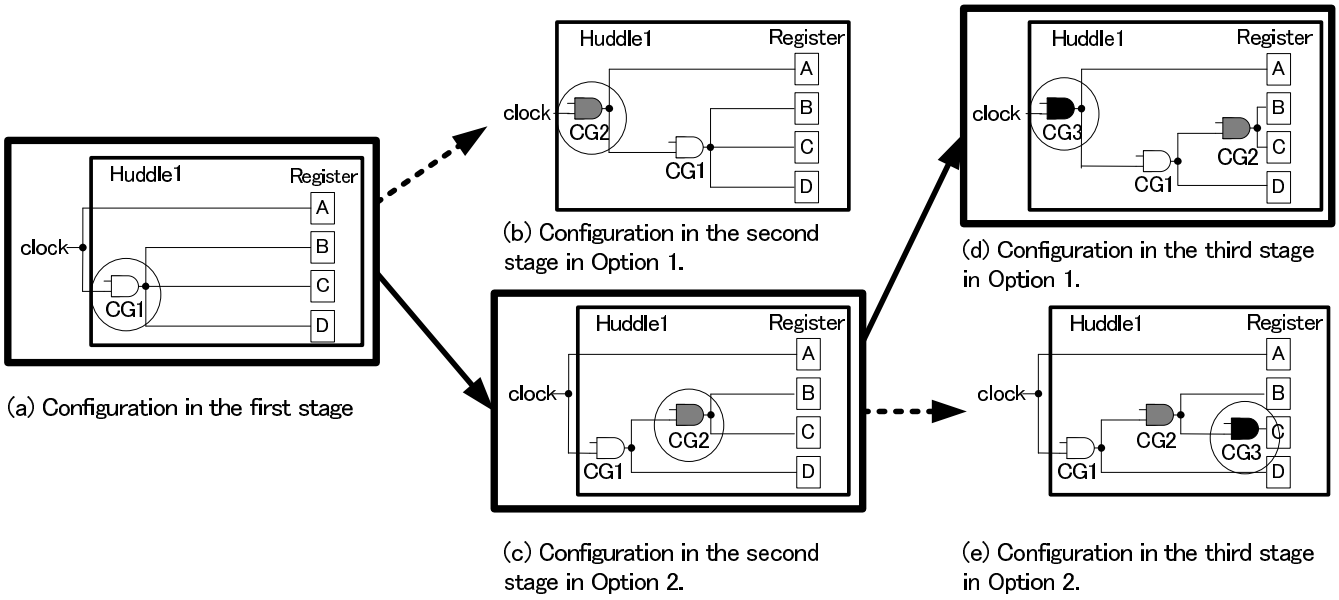


Fig. 6 Clock tree configurations associated with Fig. 5.

ing circuit in the second stage. The gray circle in Figs. 5 (b) and (c) shows the control step when CG2 cuts off the clock signal. In Fig. 5 (b), we apply CG2 to Registers *A*, *B*, *C*, and *D* and CG1 to Registers *B*, *C*, and *D*. In Fig. 5 (c), we apply CG1 to Registers *B*, *C*, and *D* and CG2 to Registers *B* and *C*. Since Option 2 gives a smaller-cost result than Option 1, we pick up Fig. 5 (c) in this case. We cut off the clock signal to registers at CS1, CS2, and CS5 in the first stage and at CS4 in the second stage.

3.1.3 Third Stage Clock Tree Configuration

Finally, we determine the third stage CG timing in Huddle 1. This step is similar to the previous section. Figures 5 (d) and (e) show the CG timing in the third stage when we apply Option 1 and Option 2 to the two-stage clock tree, respectively. "CG3" in Figs. 6 (d) and (e) show the clock gating circuit in the third stage. The black circle in Figs. 5 (d) and (e) shows the control step when CG3 cuts off the clock signal. Since Option 1 gives a smaller-cost result than Option 2, we pick up Fig. 5 (d) in this case and then we can have a three-stage clock gating configuration for Huddle 1.

3.2 Multi-stage CG Huddling

In "Multi-stage CG huddling," we merge several huddles into a single huddle. At the beginning of our iteration algorithm, we prepare huddles, each of which includes only a single functional unit. By merging them and sharing controllers and registers in our iteration flow, we can obtain a result with small area and low energy consumption. In a merging step, huddles with the same supply voltage and satisfying the huddle size constraint described in Section 4 are merged into a single huddle. Furthermore, we calculate the *merge priorities* for huddles and merge them in the descending order of them. In Ref. [1], the merge priority was designed based on huddle adjacency and the number of connections between huddles. Then two huddles which are located close and exchanging data frequently are merged possitively. In Ref. [3], we further introduced the idea of *Similarity* to calculate the merge priority to accommodate clock gating in huddles.

Similarity $Sim(h_j, h_k)$ represents how close the clock gating

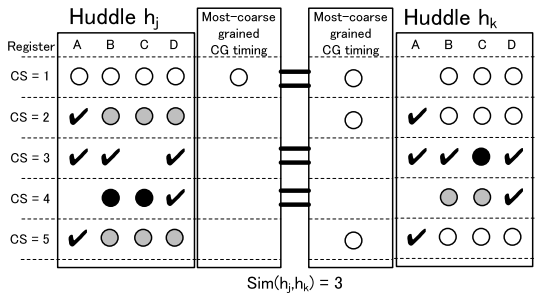


Fig. 7 Similarity.

timings of the huddles h_j and h_k are. By merging the two huddles used at the same or similar timing possitively, more registers at the same or similar timing are assigned to the same huddle. Then, we can cut off the clock signal to the registers in the merged huddle at as many steps as possible [*2].

In our algorithm here, we use the the most-coarse grained CG timing, i.e, gating steps given by the most left-side hand clock gating circuit in Figs. 5 and 6, as Similarity in our proposed algorithm.

Figure 7 shows Similarity between the two huddles h_j and h_k. The most-coarse grained CG timings in h_j and h_k are {CS1} and {CS1,CS2,CS5}, respectively. Then $Sim(h_j, h_k) = 3$ since CG timings at CS1, CS3, and CS4 match.

Based on the merge priority, we will merge two huddles into a single huddle according to the original algorithm of Ref. [1] and apply clock gating to the steps where we can cut off the clock signal to both huddles.

4. Experimental Results

We have implemented our algorithm in C++ and performed experimental evaluations. We used AMD Opteron 2360SE 2.5 GHz × 2 PC with 16 GB memory. We applied our algorithm to hal

[*2] By introducing Similarity $Sim(h_j, h_k)$ to the merge priority when merging huddles, we can reduce energy consumption by up to 10.4% compared with the case where we do not introduce it. See Ref. [2] in detail.

(11 nodes), parker (22 nodes), dct (48 nodes), jacobi (48 nodes including conditional branches), fir filter (75 nodes), ewf3 (102 nodes), and copy (378 nodes including conditional branches).

In the experiments, functional units and registers are assumed to have 16-bit width under the 90 nm technology. The clock period constraint is given to be 2.5 ns. The interconnection delays are assumed to be proportional to square of the wiring length and set interconnection delays to be 1ns when the wiring length is 250 μm [6].

The width and height of each huddle must satisfy the following *huddle size constraint* [1]:

$$2 \cdot D_w(W(h_j) + H(h_j)) + D_{reg}(h_j) \leq \min_{f_i \in F(h_j)} \{Slack(f_i)\} \qquad (3)$$

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 when the wiring length is x, $D_{reg}(h_j)$ is a delay of register in the huddle h_j, f_i is i-th functional unit, $F(h_j)$ is a set of functional units which are bound to the huddle h_j, $Slack(f_i)$ shows the slack time which can be used by data transfer for f_i's succeeding operations.

Energy consumption in each application program is obtained based on the energy consumptions of (a) FUs (including their multiplexers), (b) level converters, (c) registers (including their multiplexers), (d) controllers, and (e) clock trees. Our energy consumption value is then obtained by summing up those of (a)–(e) as seen in Refs. [1], [3], [10], where we assume that the components which are prepared in advance can be used directly in high-level synthesis. We obtained the energy consumptions of (a)–(d) by using Design Compiler. We then use the equations in Ref. [9] to calculate the energy consumption of (e) clock trees in our experiments.

We have compared our algorithm with the following algorithms:

(1) The original algorithm [1],
(2) The original algorithm [1] followed by the coarse-grained clock gating (Ref. [1]+CGCG),
(3) The original algorithm [1] followed by the fine-grained clock gating (Ref. [1]+FGCG), and
(4) Our previous algorithm [3].

In (2) and (4) above, we applied huddle-based coarse-grained clock gating whereas we applied fine-grained clock gating in (3).

Table 1 shows our experimental results. Overall, all energy consumption of our proposed algorithm achieves the minimum in all the cases and is reduced by up to 27.7% compared with the original algorithm [1]. Moreover, it is reduced by up to 10.2%

Table 1 Experimental results.

App.	FUs	S_{max}	Algorithm	All dynamic energy [pJ]	Dynamic energy in registers [pJ]	Leak energy [pJ]	CT energy [pJ]	All energy [pJ]	Iterations	CPU time (ratio) [sec]
hal	Add × 1	5	Ref. [1]	119.049	108.218	28.3314	2.31617	147.380	2	382.32 (1.000)
	Sub × 1		Ref. [1] + CGCG	102.458	92.1862	28.3354	1.75765	130.793	2	394.70 (1.032)
	Mul × 2		Ref. [1] + FGCG	101.264	88.7507	28.3354	3.99872	129.599	2	394.70 (1.032)
	Com × 1		Ref. [3]	86.3667	77.1209	29.6253	1.48581	115.992	2	381.64 (0.998)
			Ours	82.5502	73.2731	28.3354	1.50841	110.886	2	405.60 (1.061)
parker	Add × 2	10	Ref. [1]	28.3833	23.7773	68.0913	1.75087	96.4746	2	259.48 (1.000)
	Sub × 2		Ref. [1] + CGCG	26.2352	21.8553	68.0945	1.52478	94.3297	2	260.21 (1.003)
	Com × 1		Ref. [1] + FGCG	26.7604	18.0112	68.0945	5.89398	94.8549	2	260.21 (1.003)
			Ref. [3]	21.7289	17.3088	67.9547	1.32697	89.6836	2	197.45 (0.761)
			Ours	21.5102	16.9593	67.9547	1.45251	89.4649	2	262.58 (1.012)
dct	Add × 4	10	Ref. [1]	199.459	132.922	24.4439	8.76961	223.903	3	868.33 (1.000)
	Mul × 4		Ref. [1] + CGCG	187.691	122.028	24.4554	7.89428	212.146	3	913.90 (1.052)
			Ref. [1] + FGCG	203.816	106.734	24.4554	39.3136	228.271	3	913.90 (1.052)
			Ref. [3]	174.917	113.327	26.2647	8.19640	201.182	4	1,167.48 (1.345)
			Ours	167.883	103.424	26.8452	8.75431	194.728	10	3,241.60 (3.733)
dct	Add × 3	15	Ref. [1]	217.622	153.727	21.1666	10.0075	238.789	4	997.75 (1.000)
	Mul × 3		Ref. [1] + CGCG	207.997	146.126	21.1816	7.98265	229.179	4	999.68 (1.002)
			Ref. [1] + FGCG	219.559	121.599	21.1816	44.0723	240.741	4	999.68 (1.002)
			Ref. [3]	199.222	136.085	22.0748	9.25228	221.297	2	488.06 (0.489)
			Ours	192.035	128.268	22.4233	9.84309	214.458	4	955.51 (0.958)
jacobi	Add × 2	15	Ref. [1]	202.975	112.942	77.3490	12.5441	280.324	2	293.82 (1.000)
	Sub × 1		Ref. [1] + CGCG	195.489	105.669	77.3591	12.3310	272.848	2	295.23 (1.005)
	Mul × 2		Ref. [1] + FGCG	209.365	91.7126	77.3591	40.1635	286.724	2	295.23 (1.005)
	Div × 2		Ref. [3]	169.174	89.9171	79.0343	11.1586	248.208	2	273.61 (0.931)
			Ours	163.910	84.3441	79.1876	11.4317	243.098	2	331.31 (1.128)
fir	Add × 3	35	Ref. [1]	371.618	276.493	55.4682	19.6739	427.086	2	510.75 (1.000)
	Mul × 3		Ref. [1] + CGCG	344.971	249.569	55.5030	19.9510	400.474	2	554.51 (1.086)
			Ref. [1] + FGCG	364.275	214.678	55.5030	74.1458	419.778	2	554.51 (1.086)
			Ref. [3]	315.835	225.548	52.8838	13.7516	368.719	2	437.90 (0.857)
			Ours	278.336	187.085	52.9463	14.7490	331.282	2	483.20 (0.946)
ewf3	Add × 4	45	Ref. [1]	590.185	396.598	111.197	34.3950	701.382	10	2,213.18 (1.000)
	Mul × 2		Ref. [1] + CGCG	549.058	358.927	111.257	30.9392	660.315	10	2,931.81 (1.325)
			Ref. [1] + FGCG	593.152	314.492	111.257	119.469	704.409	10	2,931.81 (1.325)
			Ref. [3]	483.411	303.290	71.9863	25.6333	555.397	10	1,968.53 (0.889)
			Ours	462.794	282.510	72.0906	25.7436	534.885	10	2,012.33 (0.909)
copy	Add × 3	165	Ref. [1]	69,469.4	67,816.4	3,107.74	1,106.34	72,577.1	10	4,980.24 (1.000)
	Sub × 1		Ref. [1] + CGCG	63,840.1	62,206.7	3,108.09	1,086.75	66,948.2	10	5,011.44 (1.006)
	Mul × 5		Ref. [1] + FGCG	77,119.8	51,606.0	3,108.09	24,967.1	80,227.9	10	5,011.44 (1.006)
	Com × 1		Ref. [3]	53,316.1	52,389.6	1,989.39	498.391	55,305.5	10	5,829.51 (1.171)
	Shi × 2		Ours	49,979.2	48,409.1	2,508.96	871.604	52,488.2	10	5,172.84 (1.039)

compared with our previous version [3].

Note that fine-grained clock gating (FGCG) requires a clock gating circuit to each register in a huddle, i.e., the number of clock gating circuits in FGCG may be equal to the number of registers in a huddle. While the number of registers per huddle is 2–6 in dct and fir, it is 20–40 in copy. That is why clock tree energy (CT energy) of Ref. [1]+FGCG in copy is approximately 23 times larger than Ref. [1]+CGCG.

The number of iterations of our proposed algorithm in dct ($Add \times 4$, $Mul \times 4$) is 10, though the number of iterations of other algorithms is 3–4. This is because the process itself in our proposed algorithm is slightly complicated and might be difficult to converge. However, there is no significant impact on the convergence in our exprimental results since there is no change in the number of iterations in many caces. In the future, we will improve the convergence of our proposed algorithm.

5. Conclusion

In this paper, we have proposed a high-level synthesis algorithm based on HDR architecture utilizing the multi-stage clock gating. Our proposed algorithm reduces energy consumption by up to 27.7% compared with Ref. [1]. Moreover, our proposed algorithm reduces energy consumption by up to 10.2% compared with Ref. [3].

Setting at most one clock gating circuit per one stage is the first step in our algorithm. In the future, we will develop a high-level synthesis algorithm that optimizes the numbers and the types of clock gating in huddles.

Acknowledgments This work was supported partially by "Grant for Advanced Industrial Technology Development" from the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

[1] Abe, S., Yanagisawa, M. and Togawa, N.: Energy-efficient high-level synthesis for HDR architectures, *IPSJ Trans. System LSI Design Methodologies*, Vol.5 (August Issue), pp.106–117 (2012).

[2] Akasaka, H., Yanagisawa, M. and Togawa, N.: Energy-efficient high-level synthesis for HDR architectures with clock gating, *IEEE Proc. 2012 International SoC Design Conference*, pp.135–138 (2012).

[3] Akasaka, H., Abe, S., Yanagisawa, M. and Togawa, N.: Energy-efficient high-level synthesis for HDR architectures with clock gating based on concurrency-oriented scheduling, *IPSJ Trans. System LSI Design Methodologies*, Vol.6 (August Issue), pp.101–111 (2013).

[4] Cong, J., Fan, Y., Han, G., Yang, X. and Zhang, Z.: Architecture and synthesis for onchip multicycle communication, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.23, No.4, pp.550–564 (2004).

[5] Man, X., Horiyama, T. and Kimura, S.: Automatic multi-stage clock gating optimization using ILP formulation, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.95, No.8, pp.1347–1358, (2012).

[6] Ohchi, A., Togawa, N., Yanagisawa, M. and Ohtsuki, T.: Performance-driven high-level synthesis with floorplan for GDR architectures and its evaluation, *Proc. IEEE International Symposium on Circuits and Systems*, pp.921–924 (2010).

[7] Ozaki, N., Amano, H., Nakamura, H., Usami, K., Namiki, M. and Kondo, M.: SLD-1 (Silent Large Datapath): A ultra low power reconfigurable accelerator, *Proc. IEEE Cool Chips XIV*, pp.9–17 (2011).

[8] Shin, J., Dawei, H., Petrick, B., Changku, H. and Leon, A.: A 40 nm 16-core 128-thread SPARC SoC processor, *Proc. IEEE 2011 Solid State Circuits Conference*, pp.1–4 (2010).

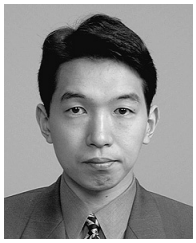
[9] Vittal, A. and Marek-Sadowska, M.: Low-power buffered clock tree design, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.16, No.9, pp.965–975 (1997).

[10] Yang, H. and Dung, L.: On multiple-voltage high-level synthesis using algorithmic transformations, *Proc. IEEE ASP-DAC*, pp.872–876 (Jan. 2005).

Hiroyuki Akasaka was born in 1989. He received his B.E. and M.E. degrees from Waseda University in 2012 and 2014 in Computer Science. His research interest is high-level synthesis of LSIs. In 2014, he joined NTT Communications Corp.

Shin-ya Abe was born in 1988. He received his B.E. and M.E. degrees from Waseda University in 2011 and 2012, respectively, all in Computer Science. He is presently working toward Dr.E. degree there. His research interests are design and verification of VLSI, especially high-level synthesis and energy efficiency design. He is a student member of the Institute of Electronics, Information and Communication Engineers.

Masao Yanagisawa was born in 1959. He received his B.E., M.E., and Dr.E. degrees 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 member of IEEE and ACM and a fellow of the Institute of Electronics, Information and Communication Engineers.



Nozomu Togawa was born in 1970. He received his B.E., M.E., and Dr.E. degrees 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 LSI design, graph theory, and computational geometry. He is a member of IEEE and the Institute of Electronics, Information and Communication Engineers.

(Recommended by Associate Editor: *Mineo Kaneko*)