Genetic Approach to Minimizing Energy Consumption of VLSI Processors Using Multiple Supply Voltages

Masanori Hariyama, *Member*, *IEEE*, Tetsuya Aoyama, and Michitaka Kameyama, *Fellow*, *IEEE*

Abstract—This paper presents an efficient search method for a scheduling and module selection problem using multiple supply voltages so as to minimize dynamic energy consumption under time and area constraints. The proposed algorithm is based on a genetic algorithm so that it can find near-optimal solutions in a short time for large-size problems. n efficient search can be achieved by crossover that prevents generating nonvalid individuals and a local search is also utilized in the algorithm. Experimental results for large-size problems with 1,000 operations demonstrate that the proposed method can achieve significant energy reduction up to 50 percent and can find a near-optimal solution (within 2.8 percent from the lower bound of optimal solutions) in 10 minutes. On the other hand, the ILP-based method cannot find any feasible solution in one hour for the large-size problem, even if a state-of-art mathematical programming solver is used.

Index Terms—Automatic synthesis, scheduling, module selection, data-path design.

1 Introduction

In recent years, low power has become a primary design concern [1]. An effective way to reduce dynamic power consumption is to lower the supply voltage of a circuit. However, reducing the supply voltage increases the circuit delay. The use of multiple supply voltages is a well-known technique which reduces dynamic power consumption without increasing the circuit delay [2]. Fig. 1 shows an example when a time constraint and a data-flow graph (DFG) to specify an input behavioral description are given. A lower supply voltage V'_{dd} can be applied to operations o_2 and o_3 because they have flexibility about the supply voltages to which they can be assigned. A higher supply voltage V_{dd} can be applied to all the other operations because they have no flexibility about the supply voltages. The dynamic power consumption reduces to 70 percent if $V'_{dd} = V_{dd}/2$ because power consumption is proportional to V^2_{dd} [3]. The major concern of this technique is that the number of functional units, that is, the chip area, increases due to the delay of operations to which lower supply voltages are applied. For example, three functional units are required in the case of Fig. 1b, while only two functional units are required in the case of Fig. 1a. Therefore, an area constraint, as well as a time constraint, is important for low power design using multiple supply voltages.

- *M. Hariyama and M. Kameyama are with the Graduate School of Information Sciences, Tohoku University, Aoba6-6-05, Aramaki, Aoba, Sendai, Miyagi, 980-8579, Japan. E-mail: {hariyama, kame}@kameyama.ecei.tohoku.ac.jp.*
- *T. Aoyama is with the System Devices Research Laboratories, NEC Corp., Japan. E-mail: t-aoyama@cd.jp.nec.com.*

Manuscript received 3 Feb. 2004; revised 29 Sept. 2004; accepted 7 Oct. 2004; published online 15 Apr. 2005.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-0036-0204.

Several researches of high-level synthesis using multiple supply voltages have been reported [4], [5], [6], [7], [8]. The algorithms for time-constrained problems are presented [4], [5]. A time and area constrained problem is also discussed [7], [8] because a multiple supply voltages approach tends to result in area overheads as described above. For this problem, the integer linear programming (ILP) method is usually used. However, the ILP method is practical only for small-size DFGs.

This paper presents an efficient search method for the dynamic energy consumption minimization problem under time and area constraints which can be applicable to the large-size DFGs. The proposed algorithm is based on a genetic algorithm (GA). The critical problem for a GA is to generate nonvalid individuals which can slow down or even prevent convergence of algorithms.

In our problem, typical crossover methods such as the one-point crossover generate a large number of nonvalid individuals that don't satisfy precedence constraint since they don't consider dependencies between nodes in DFGs. To solve the problem, we propose a crossover based on data-flow graph representation. Moreover, we combine a GA and local search heuristic which can get local optima in a limited search space to make the search more efficient. Experimental results for large-size problems with 1,000 operations demonstrate that the proposed method can achieve significant energy reduction up to 50 percent and can find near-optimal solution (within 2.8 percent from the lower bound of optimal solutions) in 10 minutes. On the other hand, the ILP-based method cannot find any feasible solution in one hour for the large-size problem even if a state-of-the-art mathematical programming optimizer, which includes a lot of efficient algorithms to reduce computational

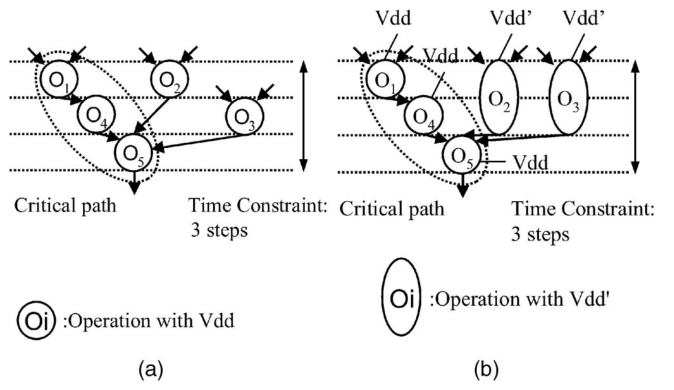

Fig. 1. Power consumption reduction using multiple supply voltages. (a) Single supply voltage. (b) Dual supply voltages.

complexity, is used to accelerate the exploration of search space.

2 Energy Minimization Problem

2.1 Data Flow Graph

An input behavioral description is given by a DFG, as shown in Fig. 2. A DFG is a directed acyclic graph $G(V, E)$, where V is a set of nodes and E a set of edges. Each $v_i \in V$ represents an operation (o_i) in the behavioral description. A directed edge e_{ij} from $v_i \in V$ to $v_j \in V$ exists in E if the result of an operation o_i is used as an input of an operation o_j. In this case, v_i is called an immediate predecessor of v_j and the set of all immediate predecessors of v_i is denoted by $Pred_{v_i}$. Each operation o_i can be executed in D_{o_i} control steps. The value of D_{o_i} depends on functional units performing operation o_i.

2.2 Datapath Architecture

Fig. 3 shows a datapath architecture, where functional units and registers are connected by multiple buses to support parallel data transfer. The architecture model is very flexible. The number of FUs, types of FUs, the number of registers, and the number of buses can be changed as long as area and time constraints are satisfied. Connections between FUs are not restricted and arbitrary point-to-point interconnection between FUs can be implemented. Moreover, the datapath architecture allows both a nonpipelined datapath and a pipelined one with an arbitrary degree of spatial parallelism.

We focus on the minimization of dynamic energy consumption that is caused by signal transitions in circuits. The technique of gating a clock is used to prevent registers from loading unnecessary new values so that unnecessary signal transitions in functional units fed by the registers are suppressed. The gated-clock datapath architecture also simplifies the objective function of the energy consumption minimization problem, as described later.

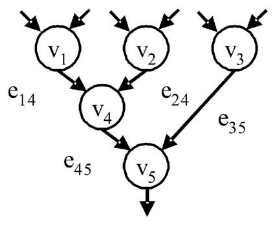

Fig. 2. Data flow graph.

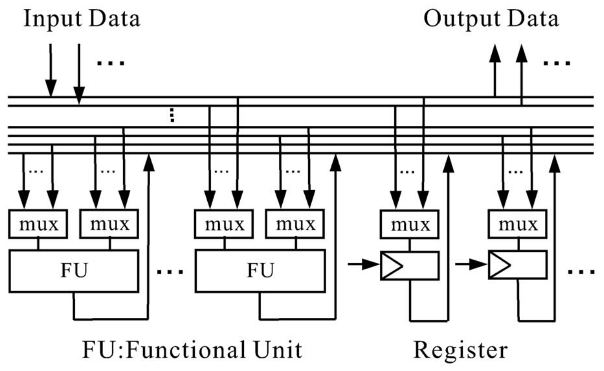

Fig. 3. Architecture model.

The use of multiple supply voltages is a well-known technique to obtain low energy implementation at reduced performance overhead. In the context of high-level synthesis, one way to utilize multiple supply voltages is module selection, that is, the process of mapping operations from the DFG to component templates from the RTL library that contains multiple versions of each component corresponding to different supply voltages. Note that only a functional unit template, not a specific instance, is associated with each operation. Table 1 shows an example of the RTL component library. The OP type denotes an operation type that can be performed by the functional unit templates. For example, functional unit templates of types F_1 and F_3 can perform addition (denoted by "ADD") and multiplication (denoted by "MUL"), respectively. The delay denotes the number of steps for one operation. The energy denotes the average energy consumption for one operation. The functional unit templates have an OP type, a supply voltage, an area, a delay, and an energy. The library can also contain a different circuit implementation for each OP type. For example, an addition can be implemented by using a ripple-carry adder, carry-lookahead adder, and carry-select adder, etc.

2.3 Problem Definition

For the energy consumption minimization problem, we make the following assumptions:

Assumption 1. *All operations synchronize with a clock cycle and let the single clock cycle be "step."*

TABLE 1
RTL Component Library

FU type	OP type	Supply voltage	Area	Delay	Energy
F_1	ADD	5V	1	1 step	2
F_2	ADD	3V	1	2 step	1
F_3	MUL	5V	8	2 step	16
F_4	MUL	3V	8	4 step	8
F_5	SUB	5V	1	1 step	2
F_6	SUB	3V	1	2 step	1

Assumption 2. *The execution time of each operation is given by kT_{clk} for an integer value k, where T_{clk} is the clock period.*

Assumption 3. *The delay involved in a register-to-register transfer is negligible.*

Assumption 4. *The energy consumed by registers and an interconnection network is negligible. The areas are also negligible.*

Assumption 5. *Static power consumption is negligible.*

Scheduling and module selection are discussed in our approach. Basically, scheduling refers to the process of mapping operations to control steps. As can be seen from Table 1, multicycle operations are used for our problem. Thus, the scheduling is extended to determine a start control step of each operation.

The goal is to minimize the total energy consumed when all the operations are performed. The total energy is simply given by the sum of energy consumption for all the operations because the gated-clock datapath architecture is employed as described above. The energy consumption for each operation depends on the functional unit to which the operation is assigned so that the objective function is given by

$$minimize \sum_{0 \leq i \leq N} (E_{F_i} \times N_{F_i}), \tag{1}$$

where E_{F_i} is the energy consumed by a functional unit of type F_i and N_{F_i} is the number of all the functional units of type F_i used in the processor.

We describe the following constraints:

Time Constraint. Any operation must finish by T_{max}, that is, the maximum number of control steps available to execute the operations in the DFG. Therefore, (2) must be satisfied for any operation o_i.

$$Sched_{o_i} + D_{o_i} - 1 \leq T_{max}, \tag{2}$$

where $Sched_{o_i}$ is the start control step of operation o_i.

Area Constraint. The total area of all the functional units must not exceed A_{max}, that is, a maximum chip area available to implement. This condition is given by the following equation:

$$\sum_{1 \leq i \leq M} (A_{F_i} \times N_{F_i}) \leq A_{max}, \tag{3}$$

where A_{F_i} is an area of the functional unit of type F_i.

Precedence Constraint. An operation o_j must not start before an operation o_i has finished if o_i is a predecessor of o_j (i.e., $o_i \in Pred_{o_j}$).

Thus, the energy consumption minimization problem is defined as the problem to schedule operations and assign a functional unit to each operation so as to minimize the energy consumption under given time and area constraints. The integer linear programming (ILP) method is usually used for the problem. We can formulate this problem as an integer linear programming problem. However, the ILP method is impractical since its execution time grows rapidly with the size of problems. Instead, we propose an efficient search method which is based on genetic algorithms and can be applicable to large-scale problems, as described next section.

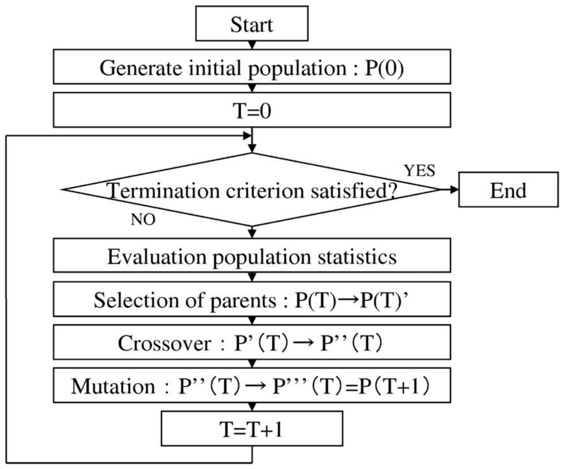
Start
Generate initial population : P(0)
T=0
Termination criterion satisfied?
YES
NO
End
Evaluation population statistics
Selection of parents : P(T)→P(T)'
Crossover : P'(T)→P''(T)
Mutation : P''(T)→P'''(T)=P(T+1)
T=T+1

Fig. 4. Flowchart of the basic genetic algorithm.

3 GA-Based Efficient Search Method

3.1 Basic Genetic Algorithm

A genetic algorithm is a stochastic search technique based on the mechanism of natural selection and natural genetics. A genetic algorithm starts with an initial set of random solutions called population. Each individual in the population is called a chromosome, which represents a solution to the problem at hand. The chromosomes evolve through successive iterations, called generations. During each generation, the chromosomes are evaluated, using some measures of fitness. To create the next generation, new chromosomes, called children, are formed by either 1) merging two chromosomes from the current generation using a crossover operator or 2) modifying a chromosome using a mutation operator. A new generation is formed by 1) selecting, according to the fitness values, some of the parents and children and 2) rejecting others so as to keep the population size constant. Fitter chromosomes have higher probabilities of being selected. After several generations, the algorithms converge to the best chromosome, which hopefully represents the optimal or suboptimal solution to the problem. The flowchart of the basic genetic algorithms is shown in Fig. 4.

3.2 Chromosome Representation

The approach for energy consumption minimization consists of scheduling and module selection as described in the previous section. Because the chromosome representation for the problem must contain the information for both scheduling and module selection, we can use the following string for the problem with n nodes:

$$x_1\, y_1\, x_2\, y_2\, x_3\, y_3\, \ldots\, x_n\, y_n,$$

where x_i is the start control step of operation o_i and corresponds to scheduling and y_i is the functional unit template which is assigned to operation o_i and corresponds to module selection.

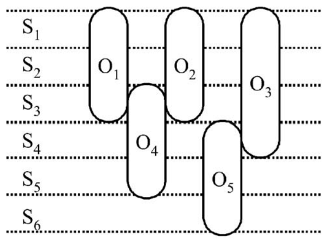

Fig. 5. Mobility range of each operation in the DFG shown in Fig. 2.

The earliest and latest bounds of x_i are determined by the As-Soon-As-Possible (ASAP) and As-Late-As-Possible (ALAP) algorithms, respectively. The ASAP algorithm maps each operation onto the earliest possible step [9], while the ALAP one maps each operation onto the latest possible step [10]. Let E_i and L_i be the start control steps into which operation o_i is scheduled by the ASAP and ALAP algorithms. The number of control steps between E_i and L_i is called the mobility range of operation o_i and is defined as follows:

$$mrange(o_i) = \{S_j | E_i \leq j \leq L_i\}.$$

Fig. 5 shows the mobility range of every operation in the DFG shown in Fig. 2 for a time constraint of six steps, where S_j is the jth control step. For example, the mobility range of o_3, $mrange(o_3)$, is $\{S_1, S_2, S_3, S_4\}$ since its ASAP and ALAP labels are $E_3 = 1$ and $L_3 = 4$, respectively. Fig. 6 shows a scheduling example for the mobility range shown in Fig. 5. The start control steps of operations o_1 and o_2 are S_3 and S_1, respectively.

The possible range of y_i can be defined as the following FU_{t_i}, where the type of an operation o_i is denoted by t_i.

FU_{t_i}: the set of functional unit templates from the given RTL component library that can perform operations of the type t_i.

Fig. 7 shows a module selection example for the RTL component library shown in Table 1, where F_j is the type of functional unit template. In the figure, the set of functional unit templates that can perform operation o_1, FU_{t_1}, is $\{F_1, F_2\}$. The functional units to which operation o_1 and o_2 are assigned are the type F_1 and F_2, respectively.

Fig. 8 shows a scheduling and module selection example. The start control step of operation o_3 is $S_2(= x_3)$ and the functional unit to which operation o_3 is assigned is $F_2(= y_3)$. The chromosome representation of this example is shown in Table 2.

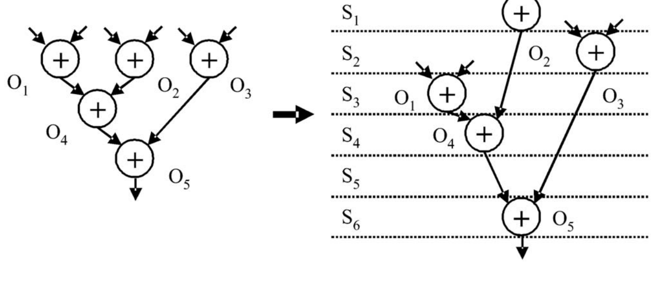

Fig. 6. Scheduling example.

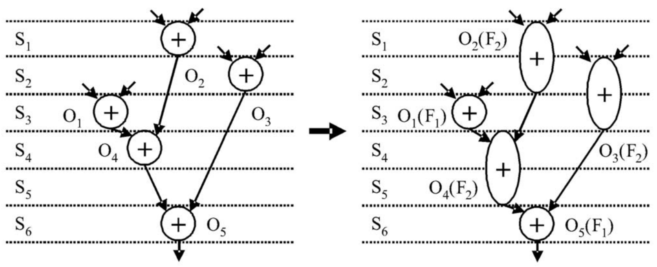

Fig. 7. Module selection example.

3.3 Crossover Based on DFG Representation

For our problem, typical crossover methods such as the one-point crossover generate a large number of nonvalid individuals which slow down or even prevent convergence of algorithms, where the nonvalid individuals are defined as individuals which do not satisfy the precedence constraint. For example, let us explain the one-point crossover, where one cut-point is randomly selected on the chromosomes and the left parts of two parents are exchanged to generate children. Let the DFGs $Parent1$ and $Parent2$ be parents (Fig. 9). Their chromosomes are shown in Table 3. Suppose that a cut-point is selected between O_3 and O_4. Then, the nodes are classified into two groups: Group1 and Group2 corresponding to the left part and right part on the chromosomes, respectively (Fig. 10). Note that O_1, O_2, and O_3 are put into Group1 although they have dependencies with O_4 and O_5 in Group2. The chromosomes of resulting children are shown in Table 4. Both children $Child1$ and $Child2$ are nonvalid individuals because they don't satisfy the precedence constraint, as shown in Fig. 11. The grouping strategy ignoring the dependencies results in a high probability of generating the nonvalid individuals since the different schedules are applied to different parents.

To solve this problem, we propose a crossover method that groups as many nodes with dependencies as possible. It is based on the idea that nodes in the same group should satisfy the precedence constraint. Given DFGs of parents, Parent1 and Parent2, the algorithm is described as follows:

Step 1: Randomly select a cut-point node on the DFGs of the parents. That is, randomly select the number CP of the cut-point node from 1 to n, where n is the total number of nodes in the DFG.

Step 2: Classify the node of Parent1 and Parent2 into two groups. Let G_{11} and G_{21} be, respectively, a set of the predecessors of the cut-point node O_{CP} and a set of nodes excepting G_{11} in Parent1. Let G_{12} and G_{22} be,

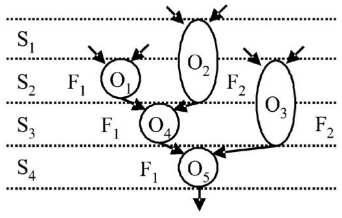

Fig. 8. Scheduling and module selection example.

TABLE 2
Chromosomes of the DFG Shown in Fig. 8

o_1		o_2		o_3		o_4		o_5	
S_2	F_1	S_1	F_2	S_2	F_2	S_3	F_1	S_4	F1

respectively, a set of the predecessors of the cut-point node O_{CP} and a set of nodes excepting G_{11} in Parent2.

Step 3: Exchange Group1 of Parent1 with that of Parent2. That is, Child1 is generated by merging G_{11} and G_{22} and Child2 is generated by merging G_{12} and G_{21}.

For example, assume that the node O_4 is selected as a cut-point node for parents shown in Fig. 9. Then, Group1 and Group2 are determined as shown in Fig. 12. Note that only O_4 in Group1 has dependencies with O_6 in Group2 and the precedence constraint is satisfied. The DFG representation of chromosomes generated by the graph-based crossover is shown in Fig. 13.

3.4 Combination of Local Search and GA

In order to achieve a more efficient search, a genetic algorithm is combined with a local search. A local search technique is used to find local optima in a given problem search space and a genetic algorithm is used to search the space of local optima in order to find the global optimum. Fig. 14 shows the structure of the combination of a local search and genetic algorithm. The local search is applied to new children generated by a crossover and mutation operators. All the individuals in the population obtained by the local search represent local optima. They are evaluated based on their energy consumption values. Promising individuals are selected from the set of local optimal solutions to form the next generation.

We describe a local search for our problem. The local search is applied to all individuals in every generation. The algorithm is shown as follows:

Step1: Select one individual (I_i) from the population (P), where P is a set of individuals generated by crossover and mutation operators. $P = P - \{I_i\}$.

Step2: Select one operation (o_i) from O_{I_i}, where O_{I_i} is a set of nodes in the individual (I_i). $O_{I_i} = O_{I_i} - \{o_i\}$.

Step3: Search a feasible scheduling and module selection for operation o_i to improve the solution, while the scheduling and module selection for all the operations except operation o_i are fixed.

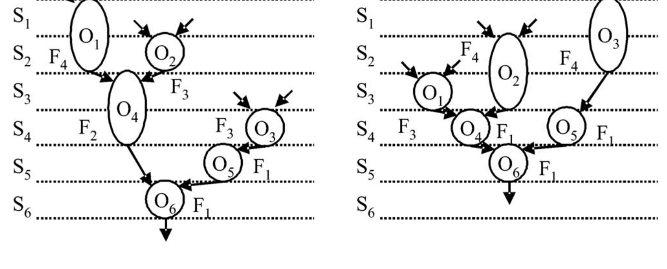

Fig. 9. Scheduling and module selection example (left: Parent1, right: Parent2).

TABLE 3
Chromosome Representation of the DFGs Shown in Fig. 9

	o_1		o_2		o_3		o_4		o_5		o_6	
Parent1	S_1	F_4	S_2	F_3	S_4	F_3	S_3	F_2	S_5	F_1	S_6	F_1
Parent2	S_3	F_3	S_2	F_4	S_1	F_4	S_4	F_1	S_4	F_1	S_5	F_1

Step4: If $O_{I_i} \neq \phi$, then go to Step2.

Step5: If $P \neq \phi$, then go to Step1.

Since the scheduling and module selection for every operation except operation o_i are fixed, the local optima are found in reasonable time. Suppose that an individual shown in Fig. 15a is given. Let us explain the local search for operation o_1. In this case, the scheduling and module selection for all the operations except operation o_1, that is, operations o_2, o_3, o_4, and o_5, are fixed. A feasible scheduling and module selection for only operation o_1 are searched. The resulting individual obtained by the local search for operation o_1 is shown in Fig. 15b, where $V'_{dd} < V_{dd}$. The functional unit which is assigned to operation o_1 changes from a high voltage unit to a low voltage one. The energy consumption for a operation o_1 is reduced, that is, the solution is improved.

4 Experimental Results

We describe some details for our algorithm.

- In order to be sure that only valid individuals are placed into the initial population, we utilize the mobility range of operations.
- Individuals having lower energy consumption are given higher fitness values. The nonvalid individuals are given the lowest fitness value.
- A roulette wheel approach is adopted as the selection procedure. It can select a new population with respect to the probability distribution based on fitness values.
- Individuals which are 10 percent of population-size are placed into the next generation without any genetic operation. They are selected according to

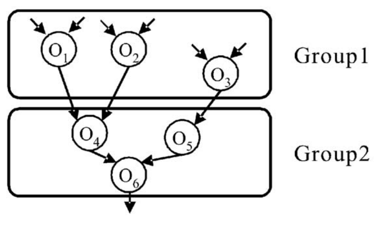

Fig. 10. Two groups divided by a cut-point for Table 3.

TABLE 4
Chromosomes Generated by One-Point Crossover for Chromosomes Shown in Table 3

	o_1		o_2		o_3		o_4		o_5		o_6	
Child1	S_3	F_3	S_2	F_4	S_1	F_4	S_3	F_2	S_5	F_1	S_6	F_1
Child2	S_1	F_4	S_2	F_3	S_4	F_3	S_4	F_1	S_4	F_1	S_5	F_1

The cut-point is between operations o_3 and o_4.

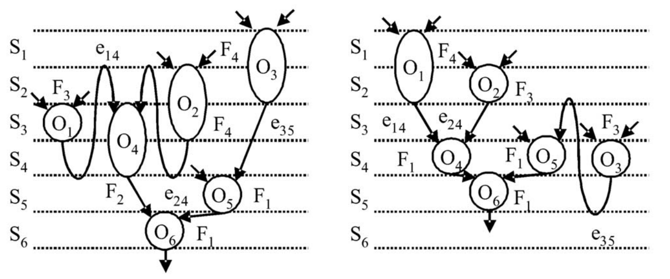

Fig. 11. DFG representation of chromosomes shown in Table 4 (left: Child1, right: Child2).

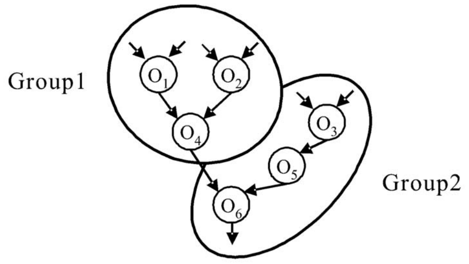

Fig. 12. Groups obtained by the graph-based crossover.

their fitness values. An individual whose fitness value is higher has a higher priority.

Let us evaluate our algorithm (hybrid of the GA using the graph-based crossover and the local search) on some high-level synthesis benchmarks (EW filter, FIR filter, and HAL) and compare it with GA using the one-point crossover and GA using the graph-based crossover.

The algorithm is executed on a PC (Athlon@800MHz, Memory 768M Byte, OS: Windows XP) and compared with two other search methods (GA using the one-point crossover and the graph-based crossover). We assume that the library shown in Table 1 is given which contains two different supply voltages 5V and 3V.

The results for some high-level synthesis benchmarks (EW filter, FIR filter, and HAL) are tabulated in Table 5. Our algorithm is denoted by *hybrid*. T_{max} is the time constraint and A_{max} is the area constraint. T_{max} is given for three different values (T_c, $1.5T_c$, and $2T_c$, where T_c is the critical path delay). E_1 is the energy consumption corresponding to the supply voltage of 5V. E_2 is the energy consumption of the best solution obtained by our algorithm for 50 trials. The reduction ratio is the percentage of E_2 over E_1. We also evaluate our algorithm with the average and variance of

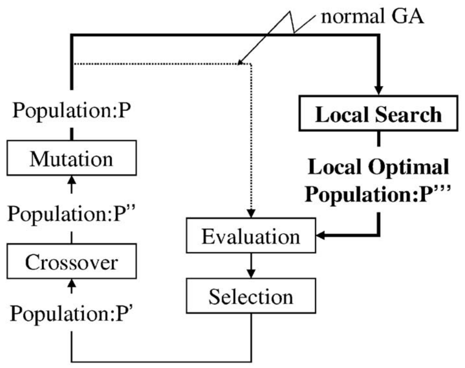

Fig. 14. Flowchart of the genetic algorithm with a local search.

50 trials. When $T_{max} = T_c$, the average energy reduction is 84.8 percent compared to E_1. Similarly, when $T_{max} = 1.5T_c$, the average energy reduction is 58.5 percent and, for $T_{max} = 2T_c$, the average energy reduction is 50.0 percent. To obtain the optimal solution, the energy minimization problem is formulated by using the integer linear programming, as described in Appendix A, and is solved by using the integer program solver package (CPLEX 7.1, ILOG Corp.). GAP is the difference between the reduction ratio and the percentage of the optimal solution over the E_1. The results show that the proposed crossover is useful compared with the one-point crossover and the optimal solutions are obtained for typical high-level synthesis benchmarks.

To evaluate our algorithm for large-size problems, the high-level synthesis benchmarks (EW filter, FIR filter, and HAL) are extended to the large-size examples (EWF30, FIR90, and HAL100). The DFGs of the EWF30 consists of 30 DFGs of the EW filter. Similarly, the DFGs of the FIR90 and HAL100 consist of 90 DFGs of FIR filter and 100 DFGs of HAL, respectively. The search time is set to 10 minutes.

The results for each example under the search time of 10 minutes are tabulated in Table 6. The time constraint is $2T_c$. E_2 is the energy consumption of the best solution obtained by our algorithm for 50 trials. The average energy reduction is 58.7 percent compared to E_1. The lower bound is obtained by an optimal linear programming solution. Note that the ILP method cannot obtain optimal solutions for these examples in one hour. GAP is the difference between the reduction ratio and the percentage of the lower

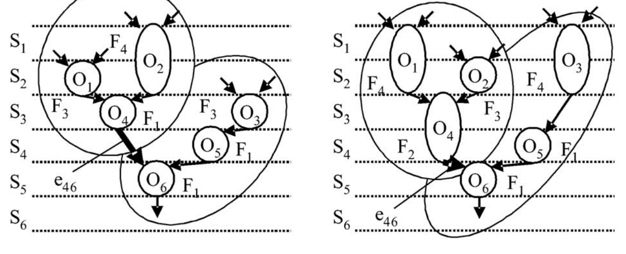

Fig. 13. DFG representation of chromosomes generated by the graph-based crossover.

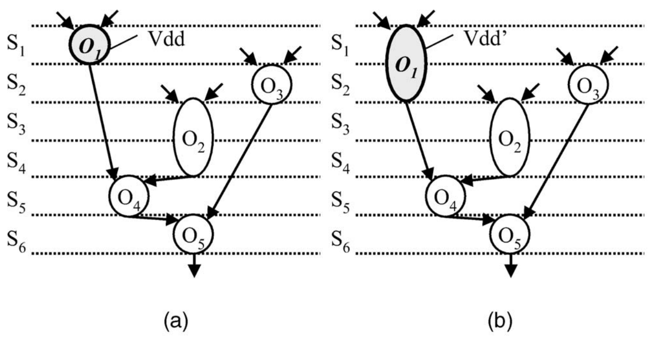

Fig. 15. Example of a local search for an operation o_1.

TABLE 5
Results for the Set of Benchmarks

Example	Search Method	T_{max}	A_{max}	E_1 (5V)	E_2 (5V,3V)	Reduction Ratio(%)	Optimal Solution	GAP (%)	Average	Variance
EWF	GA-OP	T_c	30	180	177	98.3	168	5.0	181.2	9.0
	GA-G				177	98.3	168	5.0	180.4	8.5
	HYBRID				168	93.3	168	0.0	177.9	7.5
	GA-OP	$1.5T_c$			119	66.1	103	8.9	128.6	24.1
	GA-G				113	62.8	103	5.6	126.8	24.1
	HYBRID				103	57.2	103	0.0	117.6	18.4
	GA-OP	$2T_c$			106	58.9	90	8.9	119.3	24.6
	GA-G				95	52.8	90	2.8	116.4	22.3
	HYBRID				90	50.0	90	0.0	91.1	3.7
FIR	GA-OP	T_c	45	158	115	72.8	113	1.3	118.0	16.2
	GA-G				114	72.2	113	1.0	116.8	15.1
	HYBRID				113	71.5	113	0.0	113.8	4.0
	GA-OP	$1.5T_c$			85	53.8	84	0.0	89.0	22.0
	GA-G				84	53.2	84	0.0	87.6	12.2
	HYBRID				84	53.2	84	0.0	84.0	0.1
	GA-OP	$2T_c$			80	50.6	79	0.0	83.4	6.3
	GA-G				80	50.6	79	0.0	82.5	2.6
	HYBRID				79	50.0	79	0.0	79.6	0.2
HAL	GA-OP	T_c	26	106	103	97.2	95	7.5	103.8	0.8
	GA-G				103	97.2	95	7.5	103.6	0.8
	HYBRID				95	89.6	95	0.0	95.2	0.5
	GA-OP	$1.5T_c$			80	75.5	62	17.0	83.3	37.7
	GA-G				78	73.6	62	15.1	82.6	34.1
	HYBRID				69	65.1	62	6.6	70.5	1.4
	GA-OP	$2T_c$			57	53.8	53	3.8	66.5	55.9
	GA-G				56	52.8	53	2.8	64.8	45.3
	HYBRID				53	50.0	53	0.0	58.3	13.6
Average	HYBRID	T_c				84.8		0.0		
	HYBRID	$1.5T_c$				58.5		2.2		
	HYBRID	$2T_c$				50.0		0.0		

GA-OP:GA using the one-point crossover, GA-G:GA using the graph-based crossover, HYBRID: hybrid of GA-G and the local search

TABLE 6
Results for the Large-Size Examples

Example	Search Method	T_{max}	A_{max}	E_1 (5V)	E_2 (5V,3V)	Reduction Ratio(%)	Lower Bound	GAP (%)	Average	Variance
EWF30	GA-OP	$2T_c$	900	5400	3785	70.1	2700	20.1	3872	1812
	GA-G				3713	68.7	2700	18.7	3845	1399
	HYBRID				2737	50.7	2700	0.7	2751	33
FIR90	GA-OP	$2T_c$	4050	14220	9974	70.1	7110	20.1	10159	5973
	GA-G				9748	68.5	7110	18.5	9969	2952
	HYBRID				7192	50.5	7110	0.5	7200	45
HAL100	GA-OP	$2T_c$	7800	31800	7341	94.1	5300	26.2	7461	4306
	GA-G				7231	92.7	5300	24.8	7354	2456
	HYBRID				5850	75.0	5300	7.1	5993	501
Average	HYBRID	$2T_c$				58.7		2.8		

GA-OP:GA using the one-point crossover, GA-G:GA using the graph-based crossover, HYBRID: hybrid of GA-G and the local search

bound over the E_1. The results show that our solution is within 2.8 percent of the lower bound and our algorithm is practical for large-size problems in quality and search time. On the other hand, the ILP method cannot find any feasible solutions in one hour for these large-size problems.

5 CONCLUSION

We present an efficient search algorithm based on a genetic algorithm for the energy consumption minimization problem under time and area constraints. We have also demonstrated that our algorithm can be applicable to large-size problems. Our algorithm consists of two schemes. One is a DFG-representation-based crossover that seldom generates non-valid individuals. The other is a combination of a local search and GA. For large-size examples, high-quality solutions are obtained by our algorithm in a short time.

The architecture model in this paper is simple but effective for cases where the power dissipation caused by functional units occupies most of the total power dissipation. To handle the case where power dissipations caused by interconnections and registers are dominant parts, we need additional tasks, interconnection allocation, and register allocation, because interconnections are determined after interconnection allocation and because the number of registers is also determined after register allocation. Integrating these tasks and scheduling requires an enormously large search space and is one of the next challenging issues.

The problem formulation of considering static power due to leakage current is also of importance and we are now extending our method to the problem considering static power. We will be able to achieve it by changing library representation and the objective function such that:

- the FU library includes both of power dissipations in an active and an inactive steps and
- the objective function includes the power dissipations in inactive steps for each functional unit.

APPENDIX

PROBLEM FORMULATION BASED ON INTEGER LINEAR PROGRAMMING

Let us consider energy minimization under a time constraint and an area one. Our approach consists of module selection and scheduling. As described above, module selection refers to the process of mapping operations from the DFG to component templates from the given RTL component library that contains multiple versions of each component corresponding to different supply voltages. Basically, scheduling refers to the process of mapping operations to control steps. As can be seen from Table 1, multicycle operations are used for our formulation. Thus, scheduling is extended to determining the initial control step of each operation [13].

We use the following notations for the ILP-based formulation:

A_{max}: The maximum chip area, that is, an area constraint.

T_{max}: The maximum number of control steps, that is, a time constraint.

S_j: The jth control step. $1 \leq j \leq T_{max}$.

O_{max}: The total number of operations in the given DFG.

o_i: An operation in DFG. $1 \leq i \leq O_{max}$.

F_{max}: The maximum number of possible types of functional unit.

F_k: Possible functional unit types. $1 \leq k \leq F_{max}$.

d_k: Delay time of the functional unit of the type F_k.

E_i: The earliest control step of o_i that is obtained using as-soon-as-possible (ASAP) scheduling, assuming that each operation is mapped to the fastest functional unit template available in the library.

L_i: the latest control step of o_i that is obtained using as-late-as-possible (ALSP) scheduling, assuming that each operation is mapped to the fastest functional unit template available in the library.

x_{ij}: A 0-1 integer variable for scheduling. $1 \leq i \leq O_{max}$, $E_i \leq j \leq L_i$. If the initial control step of o_i is S_j, $x_{ij} = 1$; otherwise, $x_{ij} = 0$.

$y_{i,k}$: A 0-1 integer variable for module selection. $1 \leq i \leq O_{max}$, $1 \leq k \leq F_{max}$. If o_i is mapped to the functional unit of the type F_k, $y_{i,k} = 1$; otherwise, $y_{i,k} = 0$.

E_{F_i}: The energy consumed by a functional unit of the type F_i for an operation.

N_{F_i}: The number of functional units of the type F_i.

A_{F_i}: The area of a functional unit of the type F_i.

t_i: The type of an operation o_i.

$\mathbf{FU}_{t_k}$: The set of functional unit templates from the given RTL component library that can perform operations of the type t_k.

$\mathbf{FUINDEX}_{t_k}$: The set of all the functional unit indices in $\mathbf{FU}_{t_k}$ (i.e., $\mathbf{FUINDEX}_{t_k} = \{l | F_l \in FU_{t_k}\}$).

$\mathbf{optype}_k$: The operation type of the functional unit template of the type F_k. For example, in Table 1, optype_1 and optype_3 are "ADD" and "MUL," respectively.

$\mathbf{OP}_{F_k}$: The set of operations $\{o_i | \mathrm{optype}_k = t_i\}$, that is, the set of operations that can be performed by the F_k-type functional unit.

$\mathbf{OPINDEX}_{F_k}$: The set of all the operation indices in $\mathbf{OP}_{F_k}$ (i.e., $\mathbf{OPINDEX}_{F_k} = \{i | o_i \in OP_{F_k}\}$).

To simplify the formulation, we assumed that energy consumed by registers and the interconnection network is negligible. Then, the objective function of the energy minimization problem is defined as the energy consumed by functional units to perform all the operations in the DFG since the gated-clock datapath architecture is employed as described above. The energy consumption minimization problem under time and area constraints can be formulated as minimizing

$$\sum_{0 \leq i \leq m} (E_{F_i} \times N_{F_i}), \tag{4}$$

subject to the following six constraints:

Constraint 1. The number of the initial steps of o_i must be one and the initial step must be in the range from E_i to L_i.

$$\sum_{E_i \leq j \leq L_i} x_{ij} = 1, \quad for\ 1 \leq i \leq O_{max}. \tag{5}$$

Constraint 2. Each operation must be allocated to only one functional unit template.

$$\sum_{k \in FUINDEX_{t_i}} y_{i,k} = 1, \quad for\ 1 \leq i \leq O_{max}. \tag{6}$$

Constraint 3. The operation without successors must not finish after the maximum control step specified by the time constraint T_{max}.

$$\sum_{E_i \leq j \leq L_i} (j \times x_{i,j}) + \sum_{k \in FUINDEX_{t_i}} (d_k \times y_{i,k}) - 1 \leq T_{max}, \quad for\ all\ o_i\ without\ successors. \tag{7}$$

Note that Constraints 2 and 3 ensure that the total number of control steps do not exceed T_{max}.

Constraint 4. An operation o_j must be performed after completion of an operation o_i if o_i is a predecessor of o_j (i.e., $o_i \in Pred_{o_j}$).

$$\sum_{E_i \leq l \leq L_i} (l \times x_{il}) + \sum_{k \in FUINDEX_{t_i}} (d_k \times y_{i,k}) \leq \sum_{B_j \leq m \leq L_j} (m \times x_{jm}), \quad for\ i\ and\ j\ that\ satisfy\ o_i \in Pred_{o_j}, \tag{8}$$

where $Pred_{v_i}$ denotes the set of all immediate predecessors of v_i.

Constraint 5. The number of operations that can be performed by the F_k-type functional unit in each step must not exceed the maximum number N_{F_k} of functional units of the type F_k.

$$\sum_{p=0}^{d_k-1} \sum_{i \in OPINDEX_{F_k}} x_{i,j-p} \leq N_{F_k}, \quad (9)$$
$$for\ 1 \leq j \leq T_{max}, 1 \leq k \leq F_{max}.$$

Constraint 6. The total area of the functional units must not exceed the maximum chip area A_{max}.

$$\sum_{1 \leq i \leq m} (A_{F_i} \times N_{F_i}) \leq A_{max}. \quad (10)$$

ACKNOWLEDGMENTS

This work was supported in part by the Industrial Technology Research Grant Program from the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

REFERENCES

[1] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-Power Digital CMOS Design," *IEEE J. Solid State Circuits,* pp. 473-484, Apr. 1992.

[2] K. Usami and M. Horowitz, "Clustered Voltage Scaling Technique for Low-Power Design," *Proc. Int'l Workshop Low Power Design,* 1995.

[3] A. Raghunathan, N.K. Jha, and S. Dey, *High-Level Power Analysis and Optimization.* Kluwer Academic, 1997.

[4] S. Raje and M. Sarrafzadeh, "Variable Voltage Scheduling," *Proc. 1995 Int'l Workshop Low Power Design,* 1995.

[5] J.-M. Chang and M. Pedram, "Energy Minimization Using Multiple Supply Voltages," *IEEE Trans. VLSI Systems,* pp. 436-443, Dec. 1997.

[6] W.-T. Shiue and C. Chakrabarti, "Low Power Scheduling with Resources Operating at Multiple Voltages," *IEEE Trans. Circuits and Systems II,* vol. 47, pp. 536-543, June 2000.

[7] M. Johnson and K. Roy, "Low-Power Data-Path Scheduling under Resource," *Proc. IEEE Int'l Conf. Computer Design,* 1996.

[8] Y.-R. Lin, C.-T. Hwang, and A.C.-H. Wu, "Scheduling Techniques for Variable Voltage Low Power Design," *ACM Trans. Design of Automation Electronic Systems,* pp. 227-248, July 1997.

[9] C. Tseng and D.P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems," *IEEE Trans. Computer-Aided Design,* vol. 5, pp. 379-395, 1986.

[10] S.Y. Kung, H.J. Whitehouse, and T. Kailath, *VLSI and Modern Signal Processing,* pp. 258-264. Englewood Cliffs, N.J.: Prentice Hall, 1985.

[11] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[12] D. Gajski, *High-Level Synthesis.* Kluwer Academic, 1992.

[13] C.T. Hwang, J.H. Lee, and Y.C. Hsu, "A Formal Approach to the Scheduling Problem in High-Level Synthesis," *IEEE Trans. Computer-Aided Design,* vol. 10, no. 4, pp. 464-475, 1991.

Masanori Hariyama received the BE degree in electronic engineering, the MS degree in Information Sciences, and the PhD degree in information sciences from Tohoku University, Sendai, Japan, in 1992, 1994, and 1997, respectively. He is currently an associate professor in the Graduate School of Information Sciences, Tohoku University. His research interests include VLSI computing for real-world application, such as robots, high-level design methodology for VLSIs, and reconfigurable computing. He is a member of the IEEE and the IEEE Computer Society.

Tetsuya Aoyama received the BE degree in electronic engineering, and the MS degree in information sciences from Tohoku University, Sendai, Japan, in 2001 and 2003, respectively. He joined NEC Corporation, System Devices Research Laboratories in 2003. His research interests include low-power hardware design and high-level design methodology for VLSIs.

Michitaka Kameyama received the BE, ME, and DE degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1973, 1975, and 1978, respectively. He is currently a professor in the Graduate School of Information Sciences, Tohoku University. His general research interests include intelligent integrated systems for real-world application and robotics, VLSI processor architecture and high-level synthesis, and multiple-valued VLSI computing. He received the Outstanding Paper Awards at the 1984, 1985, 1987, and 1989 IEEE International Symposiums on Multiple-Valued Logic, the Technically Excellent Award from the Society of Instrument and Control Engineers of Japan in 1986, the Outstanding Transactions Paper Award from the IEICE in 1989, the Technically Excellent Award from the Robotics Society of Japan in 1990, and the Special Award at the Ninth LSI Design of the Year in 2002. He is a fellow of the IEEE and a member of the IEEE Computer Society.