Compact QUBO Formulation of Resource-Constrained Operation Scheduling in High-Level LSI Design

Haruki Yamagishi Takuto Kishimoto Kazuhito Ito

Graduate School of Science and Engineering
Saitama University
Saitama 338-8570, Japan

Abstract— Resource-constrained operation scheduling in LSI design determines the start time of operation execution so as to satisfy precedence constraints, and is known to be an NP-hard combinatorial optimization problem. By formulating the operation scheduling problem as a QUBO model, it is possible to search for an optimal operation schedule through parallel solution of the QUBO model. This paper proposes a QUBO formulation for operation scheduling that reduces the number of variables. As a result, the number of variables was reduced by up to 65%, and the solving time was reduced by up to 81%.

I. Introduction

Operation scheduling in high-level large-scale integration (LSI) design refers to determining the start time of each operation (addition, multiplication, etc.) under precedence constraints imposed by data dependencies between operations [1]. Generally, there are multiple combinations of operation start times that satisfy all these precedence constraints. Resource-constrained scheduling takes into account the limitation on the number of operations that can be executed simultaneously due to the availability of computational resources. Figure 1(a) shows an example data flow graph (DFG) representing operations and their dependencies, while Fig. 1(b) and (c) present two examples of scheduling results assuming two adders and one multiplier. As illustrated, there can be combinations of operation execution times that require less total time to complete all operations. The goal of resource-constrained scheduling is to minimize the total execution time of all operations.

Minimizing the execution time in resource-constrained operation scheduling is known to be an NP-hard combinatorial optimization problem. On conventional computers, solving such problems optimally requires examining all possible combinations, which is computationally expensive. Therefore, heuristic methods such as list scheduling [1] have traditionally been used to find near-optimal solutions. However, solutions obtained through list scheduling are not always optimal. For example, in the DFG shown in Fig. 2, when the resources of two adders and two multipliers are assumed, list scheduling yields only a solution with an execution time of 19 (from 0 to 18) as shown in Fig. 3(a), whereas the optimal execution time is actually 18 (from 0 to 17) as shown in Fig. 3(b).

In recent years, combinatorial optimization problem formulation using the Ising model, which enables optimization

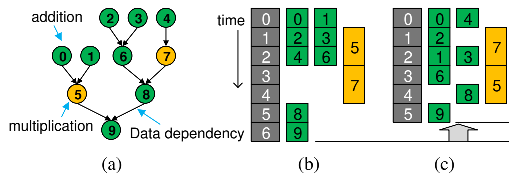


Fig. 1. (a) Data-flow graph, (b),(c) possible operation schedules.

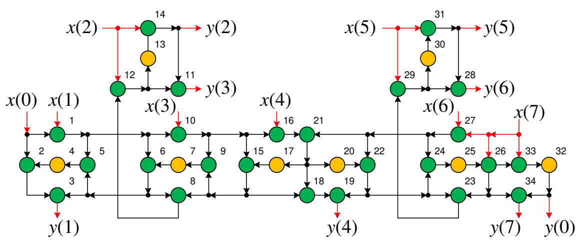
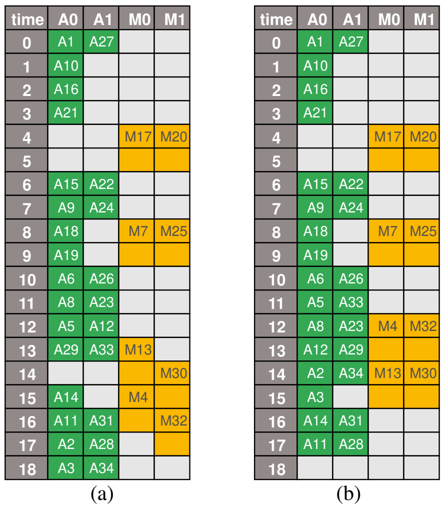


Fig. 2. 5th order wave elliptic filter (WEF).



time	A0	A1	M0	M1
0	A1	A27		
1	A10			
2	A16			
3	A21			
4			M17	M20
5			🟨	🟨
6	A15	A22		
7	A9	A24		
8	A18		M7	M25
9	A19		🟨	🟨
10	A6	A26		
11	A8	A23		
12	A5	A12		
13	A29	A33	M13	
14			🟨	M30
15	A14		M4	🟨
16	A11	A31	🟨	M32
17	A2	A28		🟨
18	A3	A34		

(a)

time	A0	A1	M0	M1
0	A1	A27		
1	A10			
2	A16			
3	A21			
4			M17	M20
5			🟨	🟨
6	A15	A22		
7	A9	A24		
8	A18		M7	M25
9	A19		🟨	🟨
10	A6	A26		
11	A5	A33		
12	A8	A23	M4	M32
13	A12	A29	🟨	🟨
14	A2	A34	M13	M30
15	A3		🟨	🟨
16	A14	A31		
17	A11	A28		
18				

(b)

Fig. 3. (a) list scheduling result, (b) an optimal schedule.

through parallel processing [2], and fast solution using dedicated machines [3, 4] or graphical processing unit (GPU) [5] have attracted attention. By formulating operation scheduling into an Ising model, it is expected that a solution can be obtained quickly using a dedicated machine or a parallel computer for solution. While the solution of the list scheduling is unique and suboptimal, it is expected that the solution of the formulated Ising model will allow the search for an optimal operation schedule with shorter operation execution time than list scheduling. It is known that the Ising model and the quadratic unconstrained binary optimization (QUBO) model can be equivalently transformed into each other. In this paper, we propose a QUBO model formulation with the reduced number of variables for resource-constrained operation scheduling.

The remainder of the paper is organized as follows. The proposed QUBO formulation is presented in Sect. 2. The solution of the formulated Ising model using GPU is described in Sect. 3. Experimental results are shown in Sect. 4 and Sect. 5 concludes the work.

II. QUBO Formulation

Energy in the QUBO formulation consists of constraint terms and objective terms. The constraint terms are 0 when the solution satisfies the constraints of the problem, and positive values otherwise. In resource-constrained operation scheduling, the constraints are (1) operation execution constraints that each operation is executed exactly once, (2) operation precedence constraints to satisfy the operation execution order derived from data dependencies, and (3) functional unit (FU) constraints to ensure that the number of operations executed simultaneously does not exceed the specified number of FUs. The objective term is minimum when the solution is the optimal for the problem. Here, the optimal solution is the one that takes the shortest time from the start of execution to the completion of all operations. The solution with the smallest energy satisfies all constraints and is the optimal solution for the problem.

In the following, we propose a QUBO formulation for the resource-constrained operation scheduling problem. First, we explain the scheduling time range, which is the set of possible execution start times for each operation to minimize the number of variables. Then, we propose two methods for formulating the FU constraints.

A. *Scheduling time range*

The operation scheduling is formulated for the time range $T = [t_s, t_e)$. The notation $[a, b)$ represents the set of times $\{t \mid a \le t < b\}$. Here, the set of all operations is N, and operation $i \in N$ can start executing in the time range $T_i = [t_{si}, t_{ei}) \subset T$ due to precedence constraints and resource constraints.

A.1. *Precedence constraints*

T_i satisfies $T_i \subset [t_s + t_{asap,i}, t_e + t_{alap,i} + 1)$ with as soon as possible (ASAP) scheduling result $t_{asap,i}$ and as late as possible (ALAP) scheduling result $t_{alap,i}$ determined by precedence constraints without resource constraints. $t_{asap,i}$ is the earliest execution start time of i when all operations start execution after time 0, and $t_{alap,i}$ is the latest execution start time of i when all operations finish execution before time 0, and defined as

$$t_{asap,i} = \max\left\{ \max_{(k,i)\in E} \{t_{asap,k} + \lambda_{p_k}\}, 0 \right\} \tag{1}$$

$$t_{alap,i} = \min\left\{ \min_{(i,j)\in E} \{t_{alap,j} - \lambda_{p_i}\}, -\lambda_{p_i} \right\} \tag{2}$$

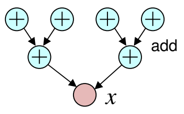


Fig. 4. Six additions preceding operation x.

where p_i is the function type of operation i, λ_p is the execution time of an operation of type p, and E is the set of all data dependencies (i, j) from operation i to operation j. The result of maximum (max) and minimum (min) are $-\infty$ and $+\infty$, respectively, when the argument is empty. Therefore, if there is no preceding operation for operation i, $t_{asap,i} = 0$, and if there is no succeeding operation for i, $t_{alap,i} = -\lambda_{p_i}$.

A.2. *Resource constraints*

The allowed time to start operation execution can be restricted by the relationship between the DFG structure and the computational resources. In the DFG in Fig. 4, $t_{asap,x} = t_s + 2\lambda_{add}$ for operation x. However, x can start only after all the preceding six additions have been executed. If the number of adders is restricted to two, the time required to execute the six additions is $6\lambda_{add}/2 = 3\lambda_{add}$, and the earliest execution start time of x is $t_{si} = t_s + 3\lambda_{add}$, which is later than $t_{asap,x}$.

Let M_p denote the number of FUs of operation type p. $B_{p,i}$ is the earliest execution start time of operation i determined by the time required by M_p FUs to execute operations of type p preceding operation i. Let Pre_i denote the set of operations which preced operation i on the DFG. That is, there is one or more directed paths from the operation in Pre_i to i on the DFG. The formula for calculating $B_{p,i}$ is

$$B_{p,i} = \max_{\substack{k\in Pre_i \\ p_k = p}} \left(t_{sk} + \left\lfloor \frac{\nu_{p,ki}}{|M_p|} \right\rfloor \lambda_p \right) \tag{3}$$

where $\nu_{p,ki}$ represents the number of operations of type p on all directed paths with operation k as the start point and operation i as the end point (excluding i). The time required to execute $\nu_{p,ki}$ operations on M_p FUs is $\lfloor \nu_{p,ki}/|M_p| \rfloor \lambda_p$. Assuming that operation k starts at its earliest execution start time t_{sk}, operation i cannot start before the time $t_{sk} + \lfloor \nu_{p,ki}/|M_p| \rfloor \lambda_p$. Therefore, the maximum value $B_{p,i}$ calculated for all operations k in Pre_i gives the earliest execution start time of operation i according to the number of FUs of operation type p. In addition, $PB_{p,i}$ is the earliest execution start time of operation i determined by the time required to execute all type p operations preceding i by M_p FUs. $PB_{p,i}$ is calculated as

$$PB_{p,i} = t_s + \left\lfloor \frac{\mu_{p,i}}{|M_p|} \right\rfloor \lambda_p \tag{4}$$

where $\mu_{p,i}$ denote the number of operations of type p in Pre_i. The maximum value of $B_{p,i}$ and $PB_{p,i}$ for all $p \in P$ is the earliest execution start time of operation i with respect to the amount of computing resources.

Similarly, $A_{p,i}$ is the latest execution start time of operation i determined by the time required to execute operations of type p succeeding operation i on M_p FUs. Let Suc_i denote the set of operations which succeed operation i on the DFG. The formula

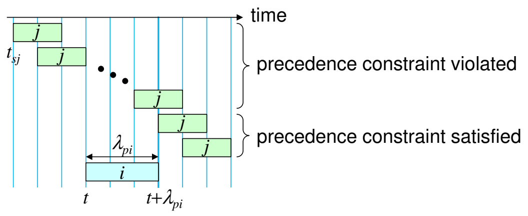

time
t_{sj}
j
precedence constraint violated
precedence constraint satisfied
λ_{pi}
i
t
$t+\lambda_{pi}$

Fig. 5. Candidate start time of operation j and precedence constraint for data dependency (i, j).

for calculating $A_{p,i}$ is

$$A_{p,i} = \min_{\substack{j \in Suc_i \\ p_j = p}} \left(t_{ej} - \left\lfloor \frac{\nu_{p,ij}}{|M_p|} \right\rfloor \lambda_p \right) \tag{5}$$

$PA_{p,i}$ is the latest execution start time of operation i determined by the time required to execute all type p operations suceeding i by M_p FUs. $PA_{p,i}$ is calculated as

$$PA_{p,i} = t_e + \left\lfloor \frac{\rho_{p,i}}{|M_p|} \right\rfloor \lambda_p \tag{6}$$

where $\rho_{p,i}$ denotes the number of operations of type p in Suc_i.

From the above, the earliest execution start time t_{si} and the latest execution start time t_{ei} of operation i are the solutions of the following set of equations.

$$t_{si} = \max \left\{ t_s + t_{asap,i}, \max_{p \in P} \{B_{p,i}, PB_{p,i}\} \right\} \quad \forall i \in N \tag{7}$$

$$t_{ei} = \min \left\{ t_e + t_{alap,i} + 1, \min_{p \in P} \{A_{p,i}, PA_{p,i}\} \right\} \quad \forall i \in N \tag{8}$$

B. FU binding method

Focusing on the fact that each FU executes at most one operation at a time, the operator constraints can be formulated by explicitly considering the binding between each operation and the FU to execute it. This QUBO formulation is called the FU binding method. The QUBO formulation was proposed in [6] and briefly reviewed here. In this method, a binary variable $x_{i,t,m}$ has the following property.

$$x_{i,t,m} = \begin{cases} 1 & \text{if operation } i \text{ starts at time } t \text{ on } m\text{th FU} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Operation execution constraint Each operation is executed exactly once at any time in any FU. This operation execution constraint is formulated by the following equation.

$$H_E = \sum_{i \in N} \left(\sum_{t \in T_i} \sum_{m=1}^{M_{p_i}} x_{i,t,m} - 1 \right)^2 \tag{10}$$

For operation i, the sum of variables $x_{i,t,m}$ for all possible execution start times $t \in T_i$ and all M_{p_i} FUs $(1 \leq m \leq M_{p_i})$ which can execute i is 1 when i is executed exactly once. Therefore, subtracting 1 and squaring the sum results in 0 when the operation execution constraint is satisfied and positive when it is violated. By taking the sum for all operations, $H_E = 0$ when the operation execution constraint is satisfied for all operations and $H_E > 0$ when any operation violates the constraint.

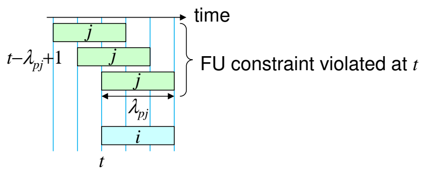


Fig. 6. Candidate start time of operation j to violate the FU constraint with operation i starting at time t.

Operation precedence constraint When there is a data dependency from operation i to operation j, j must start executing after i has finished executing. This is the operation precedence constraint, which is formulated as follows.

$$H_P = \sum_{(i,j) \in E} \sum_{t \in T_i} \left(\sum_{m_1=1}^{M_{p_i}} x_{i,t,m_1} \times \sum_{u \in R_{i,j,t}} \sum_{m_2=1}^{M_{p_j}} x_{j,u,m_2} \right) \tag{11}$$

When the operation execution constraint is satisfied for operation i, $\sum_{m_1=1}^{M_{p_i}} x_{i,t,m_1} = 1$ if i starts execution at time t, and $\sum_{m_1=1}^{M_{p_i}} x_{i,t,m_1} = 0$ otherwise. When i starts execution at time t, i ends at time $t + \lambda_{p_i}$, and j must start execution at or after $t + \lambda_{p_i}$ to satisfy the operation precedence constraint as shown in Fig. 5. Considering the scheduling time range of j, the time set $R_{i,j,t}$ is defined as

$$R_{i,j,t} = \{ u \,|\, t_{sj} \leq u \leq \min(t + \lambda_{p_i} - 1, t_{ej} - 1) \} \tag{12}$$

When j starts execution at any time u in $R_{i,j,t}$, $\sum_{m_2=1}^{M_{p_j}} x_{j,u,m_2} = 1$ and it means that j starts execution before $t + \lambda_{p_i}$. Therefore, for time t, if $\sum_{m_1=1}^{M_{p_i}} x_{i,t,m_1} = 1$ and $\sum_{u \in R_{i,j,t}} \sum_{m_2=1}^{M_{p_j}} x_{j,u,m_2} = 1$, that is, if the product of the two terms is 1, the operation precedence constraint is violated. Otherwise, the product is 0, and $H_P = 0$ when operation precedence constraints are satisfied for all data dependencies. If any operation precedence constraint is violated, $H_P > 0$.

FU constraint Each FU cannot execute more than one operation at the same time. This FU constraint is formulated by the following equation.

$$H_R = \sum_{p \in P} \left(\sum_{\substack{i,j \in N_p \\ i \neq j}} \sum_{t \in T_i} \sum_{m=1}^{M_{p_i}} \left(x_{i,t,m} \times \sum_{u \in Q_{j,t}} x_{j,u,m} \right) \right) \tag{13}$$

The execution duration of operation j is λ_{p_j}. When j starts execution at time w, an FU is used from time w to time $w + \lambda_{p_j} - 1$. The set of times $Q_{j,t}$ is defined as

$$Q_{j,t} = \{ u \,|\, \max(t - \lambda_{p_j} + 1, t_{sj}) \leq u \leq \min(t, t_{ej} - 1) \} \tag{14}$$

If j starts execution at any time $u \in Q_{j,t}$, an FU is used to execute j at time t as shown in Fig. 6. When operation i of the same type p_j $(p_i = p_j)$ starts execution in the mth FU at t, $x_{i,t,m} = 1$. At the same time, if $\sum_{u \in Q_{j,t}} x_{j,u,m} = 1$, i and j are being executed in the same mth FU at t, and it means the FU constraint is violated.

When the number of operations executed simultaneously is at most 1 in all the FUs, $H_R = 0$, and when the FU constraint is violated, $H_R > 0$.

Objective The objective H_c is the sum of the execution times of all operations, and when H_c is minimized, it is expected that the execution completion time of the last operation to be executed will be minimized and the derived schedule is optimal.

$$H_c = \sum_{i \in N} \sum_{t \in T_i} t \times \sum_{m=1}^{M_{p_i}} x_{i,t,m} \tag{15}$$

The number of variables Each operation i needs $|M_{p_i}||T_i|$ variables to check the violations of the FU constraints for all time ranges T_i. Hence the total number of variables in this formulation is

$$\sum_{i \in N} |M_{p_i}||T_i| \tag{16}$$

C. Operation counting method

In the above-mentioned FU binding method, the FU constraint is satisfied by ensuring no more than one operations are executed simultaneously in each FU. Since variables $x_{i,t,m}$ are prepared for all combinations of operations, time, and FUs, the number of variables increases. We propose a formulation that constrains the number of operations executed simultaneously to be less than or equal to the number of FUs, thereby reducing the number of variables. This formulation is called the operation counting method.

Binary variables $\overline{x_{i,t}}$ and $y_{p,m,t}$ are employed.

$$\overline{x_{i,t}} = \begin{cases} 1 & \text{if operation } i \text{ starts at time } t \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$y_{p,m,t} = \begin{cases} 1 & \text{if } m\text{th FU of type } p \text{ is used at time } t \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Operation execution constraint Each operation is executed exactly once at any time in any FU. This operation execution constraint is formulated by the following equation.

$$H_E = \sum_{i \in N} \left(\sum_{t \in T_i} \overline{x_{i,t}} - 1 \right)^2 \tag{19}$$

Operation precedence constraint When there is a data dependency from operation i to operation j, j must start executing after i has finished executing. The operation precedence constraint is formulated by the following equation.

$$H_P = \sum_{(i,j) \in E} \sum_{t \in T_i} \left(\overline{x_{i,t}} \times \sum_{u \in R_{i,j,t}} \overline{x_{j,u}} \right) \tag{20}$$

If i starts execution at time t, $\overline{x_{i,t}} = 1$. If j starts execution at any time $u \in R_{i,j,t}$, that is, j starts before the execution of i finished, $\sum_{u \in R_{i,j,t}} \overline{x_{j,u}} = 1$. In that case the precedence constraint is violated and $H_P > 0$.

FU constraint When operation i starts execution at time $u \in Q_{i,t}$, an FU is used at time t to execute i. Hence execution of i uses an FU at time t if $\sum_{u \in Q_{i,t}} \overline{x_{i,u}} = 1$. The number of operations of type p executed at time t is obtained as

$$K_{p,t} = \sum_{i \in N_p} \sum_{u \in Q_{i,t}} \overline{x_{i,u}} \tag{21}$$

```
input: ( $S$, $\mathscr{F}$, $h$, $J$, $T_{hot}$, $T_{cold}$, $Ratio$, $Repeat$)
initialize all $\sigma_i \in S$
for $T$ from $T_{hot}$ to $T_{cold}$ multiplied by $Ratio$
  for $repeat$ from 1 to $Repeat$
    for $S' \in \mathscr{F}$
      for $\sigma_i \in S'$ in parallel
        $\delta_i \leftarrow h_i$
        for $\sigma_j \in S$, $j \neq i$ in partially parallel
          $\delta_i \leftarrow \delta_i + J_{ij}\sigma_j$
        $\Delta H \leftarrow -2\delta_i\sigma_i$
        random_value $\leftarrow$ random value of [0,1)
        if $\exp(-\Delta H/T) >$ random_value
          $\sigma_i \leftarrow -\sigma_i$
```

Fig. 7. Pseudo code of Ising annealing.

$K_{p,t}$ must be less than or equal to M_p and the FU constraint is formulated by the following equation

$$H_R = \sum_{p \in P} \sum_{t \in T(p)} \left(K_{p,t} - \sum_{m=1}^{M_p} y_{p,m,t} \right)^2 \tag{22}$$

where $T(p)$ is the set of times at which the maximum number of FUs used by operations of type p may exceed M_p. $Y_{p,t} = \sum_{m=1}^{M_p} y_{p,m,t}$ takes a value from 0 to M_p according to the combination of $y_{p,m,t}$ values. When $K_{p,t} \leq M_p$, $y_{p,m,t}$ are set to 0 or 1 accordingly so that $Y_{p,t} = K_{p,t}$ and hence $K_{p,t} - Y_{p,t} = 0$. When $K_{p,t} > M_p$, that is, FU constraint is violated, $K_{p,t} - Y_{p,t} > 0$. $H_R = 0$ if all the FU constraints are satisfied.

Objective The objective H_c is the sum of the execution times of all operations, and when H_c is minimized, it is expected that the execution completion time of the last operation to be executed will be minimized and the derived schedule is optimal.

$$H_c = \sum_{i \in N} \sum_{t \in T_i} t \times \overline{x_{i,t}} \tag{23}$$

The number of variables The required number of variables $\overline{x_{i,t}}$ is $\sum_{i \in N} |T_i|$. The number of variables $y_{p,m,t}$ is $|M_p||T(p)|$ for each operation type p, so the total number of variables is $\sum_{p \in P} |M_p||T(p)|$. Consequently, the required number of variables for this method is

$$\sum_{i \in N} |T_i| + \sum_{p \in P} |M_p||T(p)| \tag{24}$$

A DFG can generally be considered as $|N| \gg |P|$. In the FU binding method, the number of variables is about $|N||M||T|$, whereas in the operation counting method, the number of variables is about $|N||T| + |P||M||T|$. Therefore the operation counting method is expected to require less variables than the FU binding method.

III. Annealing of Ising model on GPU

The QUBO model can be converted to an Ising model by associating each QUBO variable x_i with the Ising model variable σ_i as $x_i = (\sigma_i + 1)/2$ [2]. The correspondence between the variables and energy remains unchanged before and after this conversion, and they are converted into an equivalent energy equation. Therefore, QUBO can be optimized by optimizing the converted Ising model.

The converted Ising model is solved using a GPU. Figure 7 shows the pseudocode for solving the parallelized Ising model by simulated annealing [7]. Here, S is the set of all spins σ_i, $\mathscr{F}$ is the set of all subsets that completely divide S, h is the external magnetic field, J is the mutual coupling coefficient, T_{hot} is the starting temperature, T_{cold} is the ending temperature, *Ratio* is the temperature change ratio, and *Repeat* is the number of repetitions at the same temperature.

Completely dividing S means that $J_{ij} = 0$ for all pairs of spins σ_i, σ_j of each element S' of $\mathscr{F}$. Only the spins included in each element S' of $\mathscr{F}$ are updated in parallel at the same time. This is to prevent the occurrence of loops that do not reach the optimal solution when mutually coupled spins are updated at the same time. An example of a loop that does not reach the optimal solution is as follows. The minimum value of the energy of $H = \sigma_1\sigma_2 + \sigma_2\sigma_3$ is $H = -2$, but when $(\sigma_1, \sigma_2, \sigma_3) = (1, 1, 1)$ and $H = 2$, all spins are updated to -1 at the same time, resulting in $(\sigma_1, \sigma_2, \sigma_3) = (-1, -1, -1)$, which remains at $H = 2$, and then is updated to $(\sigma_1, \sigma_2, \sigma_3) = (1, 1, 1)$, and this process is repeated. In this way, updating all spins at the same time can result in a state where two states oscillate. By introducing $\mathscr{F}$, the group of spins that are updated simultaneously in parallelization is limited to spins that have no mutual coupling.

Consequently, the QUBO model is formulated for a given operation scheduling problem and converted to the Ising model, and $\mathscr{F}$ is obtained on a processor. Then the annealing of the Ising model is performed on a GPU.

Time To Solution (TTS) is the time it takes to find the optimal solution with a probability of p_R [8]. If the probability of finding the optimal solution in one solution is p_s, then the probability of finding the optimal solution at least once in R solution attempts is $1 - (1 - p_s)^R$. Therefore, when this is p_R, R is given as follows.

$$R = \frac{\ln(1 - p_R)}{\ln(1 - p_s)} \tag{25}$$

Thus TTS is $R\tau$ if the time of one solution attempt is τ.

IV. Experimental Results

A. Conditions

The DFGs used in the experiments are shown in Figs. 8 to 10 and Fig. 2. The specifications of the DFGs are shown in Table I. Shown in Table I are from left to right, the name of DFG, the number of additions $|N_A|$, the number of multiplications $|N_M|$, the numbers of adders M_A and multipliers M_M given as resource constraints, the operation execution time LS obtained by list scheduling, and the minimum operation execution time SS obtained by integer programming, i.e., the optimal value of the operation execution time. It is assumed that an addition takes 1 unit of time (u.t.) and a multiplication takes 2 u.t., and the FUs are not pipelined. The goal of the proposed method is to obtain an operation execution time that matches SS for each DFG by simulated annealing of the Ising model converted from the QUBO model.

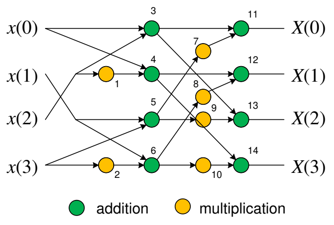


Fig. 8. DFG A.

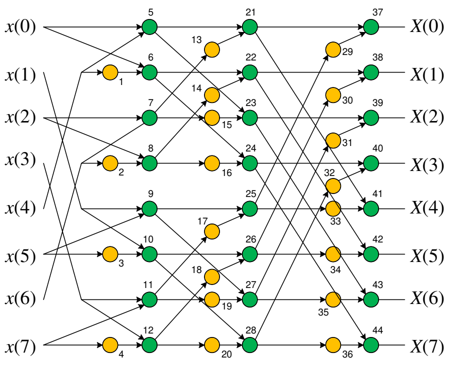


Fig. 9. DFG B.

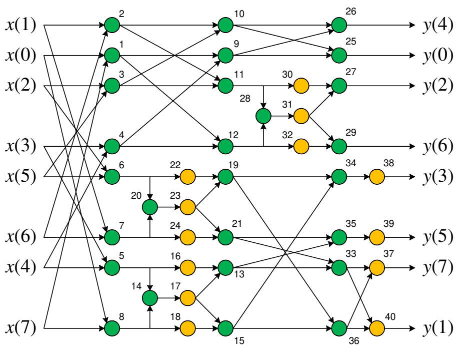


Fig. 10. 8-point DCT (DCT8).

TABLE I
DATA-FLOW GRAPHS

DFG	$\lvert N_A\rvert$	$\lvert N_M\rvert$	M_A	M_M	LS	SS
DFG A	8	8	1	2	8	8
DFG B	24	20	2	4	12	12
DCT8	27	13	4	4	9	9
WEF	26	8	2	2	19	18

The energy used in solving the problem is defined as

$$H = aH_c + \alpha H_E + \beta H_P + \gamma H_R \tag{26}$$

where a, α, β, γ are positive coefficients. These coefficients represent the strength of the influence of the terms in solving the problem.

The experiments were done on a PC with an Intel Core i9-12900 processor and an NVIDIA GeForce RTX 3060 GPU. The QUBO models were generated using the CPU, and there was no significant difference in generation time between the FU binding method and the operation counting method. The parameters used in the annealing are shown in Table II.

TABLE II

PARAMETERS FOR ANNEALING

# solutions	10000
T_{hot}	50
T_{cold}	0.5
Ratio	0.9998
Repeat	10

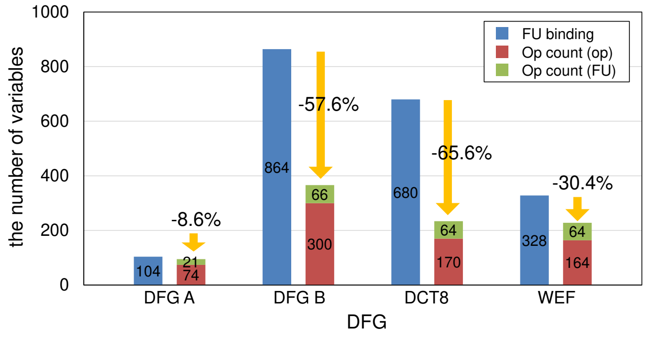


Fig. 11. The number of QUBO variables.

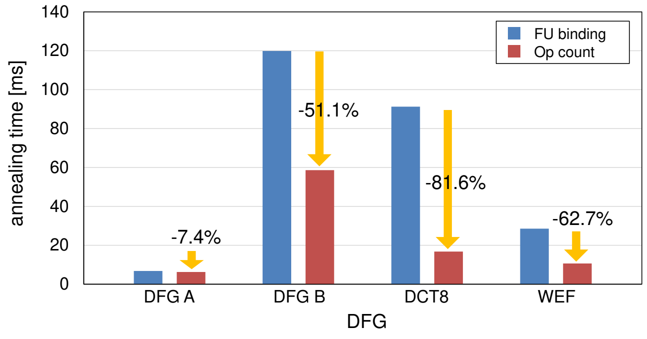


Fig. 12. The average GPU annealing time.

B. *Evaluation of the proposed models*

For each DFG, list scheduling was performed to obtain *LS*, and then the QUBO model was formulated with $t_s = 0$ and $t_e = LS + 1$. This is because *SS* is always less than or equal to *LS*. The number of variables used in the formulation using the FU binding method and the proposed operation counting method is shown in Fig. 11. In Fig. 11, 'FU binding' represents the number of variables $x_{i,t,m}$ used in the FU binding method, 'Op count (op)' represents the number of variables $\overline{x_{i,t}}$ used in the operation counting method, and 'Op count (FU)' represents the number of variables $y_{p,m,t}$ used in the same method. The operation counting method reduces the number of variables by up to 65% compared to the FU binding method. The reduction rate of the number of variables is higher when the number of FUs specified as constraints is large.

The average time for 10000 solution runs on the GPU is shown in Fig. 12. The operation counting method reduces the time up to 81% compared to the FU binding method.

The experiment confirmed that the operation counting method reduces the number of variables in the QUBO formulation of operation scheduling compared to the FU binding method, thereby reducing the time required for one solution attempt.

TABLE III

ENERGY WEIGHTS AND TTS FOR $p_R = 0.99$

DFG	method	a	α	β	γ	τ [ms]	R	$R\tau$ [ms]
DFG A	FUbind	1.1	10	4	6	6.736	2	13.47
DFG A	OpCnt	0.8	10	8	4	7.865	5	39.33
DFG B	FUbind	1.0	14	8	12	120.6	5	603.1
DFG B	OpCnt	1.0	15	10	8	60.25	686	41333
DCT8	FUbind	0.9	10	8	8	92.55	4	370.2
DCT8	OpCnt	0.9	12	12	4	16.69	227	3788
WEF	FUbind	1.0	20	14	16	27.78	5	138.9
WEF	OpCnt	1.0	20	10	4	10.32	21	216.8

C. *Solution results*

Table III shows a comparison of TTS at $p_R = 0.99$ for the solution results of each DFG and each formulation method. The energy coefficients a, α, β, and γ are the combinations found by grid search that obtain the optimal solution most times.

The operation counting method is at a disadvantage in TTS compared to the FU binding method. The biggest difference in energy between the FU binding method and the operation counting method is the FU constraint. This constraint term may make the energy landscape difficult to solve. Therefore, it is thought that TTS can be improved by expressing the FU constraint of the operation counting method differently from the current one.

V. CONCLUSIONS

In this paper, a QUBO model formulation for resource-constrained operation scheduling in high-level design of LSIs was proposed. The number of variables is reduced up to 65% and the solution time is reduced by up to 81% by the operation counting method compared to the FU binding method. TTS of the operation counting method is longer than the FU binding method and it is necessary to reduce the TTS of the operation counting method. The objective of optimization in this work is the sum of the execution start times of all operations. For minimizing the time from the start of operation execution to the completion of all operation execution, the objective should be changed. This remains as future work.

REFERENCES

[1] R. Walker and S. Chaudhuri, "Introduction to the scheduling problem," IEEE Design & Test of Computers, vol.12, no.2, pp.60–69, 1995.

[2] A. Lucas, "Ising formulations of many NP problems," Frontiers in physics, vol.2, no.5, pp.1–15, 2014.

[3] M. Yamaoka, T. Okuyama, M. Hayashi, C. Yoshimura, and T. Takemoto, "CMOS annealing machine: an in-memory computing accelerator to process combinatorial optimization problems," 2019 IEEE Custom Integrated Circuits Conference (CICC), pp.1–8, 2019.

[4] Y.H. Chen, C.A. Chou, C.F. Nien, and S.Y. Lin, "Design and implementation of a VLSI-based annealing accelerator for efficiently solving combinatorial optimization problems," IEEE Transactions on Circuits and Systems II: Express Briefs, vol.71, no.9, pp.4291–4295, 2024.

[5] C. Cook, H. Zhao, T. Sato, M. Hiromoto, and S.X.D. Tan, "GPU based parallel Ising computing for combinatorial optimization problems in VLSI physical design," arXiv preprint arXiv:1807.10750, 2018.

[6] T. Kishimoto and K. Ito, "Ising model formulation of operation scheduling in LSI design," IEICE Conferences Archives, The Institute of Electronics, Information and Communication Engineers, 2023. in Japanese.

[7] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi, "Optimization by simulated annealing," Science, vol.220, no.4598, pp.671–680, 1983.

[8] Jij Inc., "Annealing algorithm evaluation and openjij benchmarking capabilities." https://tutorial.openjij.org/en/tutorial/004-Evaluation.html. Accessed: 2025-05-16.