Multivoltage multifrequency low-energy synthesis for functionally pipelined datapath

Xing, Xianwu; Jong, Ching Chuen

2009

Xing, X., & Jong, C. C. (2009). Multivoltage Multifrequency Low-Energy Synthesis for Functionally Pipelined Datapath. IEEE Transactions On Very Large Scale Integration (VLSI) Systems. 17(9), 1348-1352.

<https://hdl.handle.net/10356/91872>

<https://doi.org/10.1109/TVLSI.2008.2002684>

© 2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder. <http://www.ieee.org/portal/site> This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

Multivoltage Multifrequency Low-Energy Synthesis for Functionally Pipelined Datapath

Xianwu Xing and Ching Chuen Jong

Abstract—In this paper, an algorithm named MuVoF is proposed to perform multivoltage multifrequency low-energy high-level synthesis for functionally pipelined datapath under resource and throughput constraints. A datapath is partitioned into a number of pipelined stages such that the clock period can be extended maximally. A multivoltage assignment algorithm then utilizes the extended clock period to reduce energy by lowering the supply voltages of the resources. The results are further refined by four local transformations performed in an iterative process. The experiment results show that MuVoF is capable of exploring the design space effectively and achieves efficient energy reduction.

Index Terms—Functional pipelining, low energy, multivoltage multifrequency (MuVoF), high-level synthesis.

I. INTRODUCTION

As the density of VLSI systems is rapidly increasing and portable and wireless devices, in which energy consumption must be reduced to extend battery life, are more and more popular, energy consumption has become a more critical design factor than circuit size. The well-known formula $E_{\text{dynamic}} = 0.5 \cdot V_{dd}^2 \cdot C_{\text{out}} \cdot E_{\text{transition}}$ shows that the dynamic energy consumption depends on three parameters: the supply voltage (V_{dd}), the load capacitance (C_{out}), and the switching activity ($E_{\text{transition}}$). Power/energy optimization based on switching activity reduction has been widely explored [1], [2]. Due to the quadratic dependency on supply voltage, power/energy optimization based on voltage scaling is regarded as one of the most effective methods. Multivoltage techniques have gained popularity among voltage scaling techniques [3]–[6]. In [3], it was demonstrated that multivoltage scheduling problem was NP-hard and a dynamic programming approach was proposed. In [4] and [5], mobility-based heuristic algorithms were proposed for the scheduling problem of multivoltage datapath for simultaneous energy and transient power reduction. There were also works on frequency scaling. A multiple-clock scheme for register transfer level (RTL) datapath synthesis was proposed in [7]. The authors of [4]–[6] used dynamic frequency clocking (DFC) to achieve optimization on variable power and energy parameters. DFC does not directly reduce power, but compensates the extended delay caused by reducing voltage supply of resources.

In this paper, a technique named multivoltage multifrequency (MuVoF) is proposed for low-energy high-level synthesis for functionally pipelined datapath under resource and throughput constraint. Functionally pipelined structure has existed for a long time, but has not been explored for low-energy high-level synthesis. In MuVoF, the datapath is first partitioned into several pipelined stages, each of which operates at its own frequency subject to a global throughput constraint. The proposed stage partitioning algorithm optimally creates large slacks in some stages so that their clock periods can be extended under the throughput constraint. Then, a multivoltage assignment process utilizes the extended clock period for supply voltage reduction. To our best knowledge, this is the first technique addressing the high-level

Fig. 1. DFG example (a) with its multivoltage multifrequency functionally pipelined datapath (b). (a) IIR DFG. (b) Pipelined stages.

synthesis of multivoltage and multifrequency functionally pipelined datapath with the low energy as the optimization goal.

The next section introduces the multifrequency and multivoltage synthesis for functionally pipelined datapath. Section III describes the proposed algorithms. Section IV presents the test results and Section V concludes the paper.

II. MULTIVOLTAGE AND MULTIFREQUENCY SYNTHESIS FOR FUNCTIONALLY PIPELINED DATAPATH

MuVoF synthesizes energy-efficient multivoltage functionally pipelined datapath from a scheduled data flow graph (DFG). We use the DFG of an infinite impulse response (IIR) filter shown in Fig. 1(a) to illustrate the concept. The schedule is represented as S_INIT1, with the operations in the same control step (cstep) enclosed by a pair of brackets “(),” i.e.,

$$\text{S_INIT1} = \{(17, 16), (15, 19), (21, 20, 14), (23), (13, 18), (22), (24), (26), (25), (27), (28), (29)\}.$$

A. Functional Pipelining for Power/Energy Optimization

A functionally pipelined datapath consists of several partitioned stages. Each stage has its own resources and operates at its own clock period. Let us consider S_INIT1 with a single supply of 5.0 V and a single clock period of 30 ns. The total delay would be $16 \times 30 \text{ ns} = 480 \text{ ns}$. If the input data arrives faster than this, additional resources are required to construct a pipelined datapath in order to satisfy the throughput. Suppose the data arrives every 300 ns and four multipliers and two arithmetic logic units (ALUs) are now available, the schedule S1 below is a feasible datapath with two pipelined stages,

Manuscript received October 24, 2007; revised March 02, 2008. First published February 27, 2009; current version published August 19, 2009.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: xing0003@ntu.edu.sg; eccjong@ntu.edu.sg).

Digital Object Identifier 10.1109/TVLSI.2008.2002684

each enclosed by a pair of “{ }.” The datapath now has a total delay of $10 \times 30 \text{ ns} = 300 \text{ ns}$

$$S1 = \{(17, 16), (), (19, 15), (), (14, 20, 21), (23)\} \\ + \{(13, 18), (), (22), (24), (26), (25), (27), (28), (), (29)\}.$$

The functionally pipelined datapath provides space for power/energy optimization. As different stages have different latencies, the clock periods in the stages with shorter latencies can be extended, resulting in peak power reduction. The extended clock period also facilitates supply voltage reduction, which reduces both power and energy. In addition, the operations in a stage have less dependency constraints than those in a nonpipelined datapath, thus creating more spaces for voltage reduction, as further discussed below.

B. Multivoltage and Multifrequency Synthesis

Given ALU and multiplier as the available library components, their delay (ns) and energy (pJ) characterization in Table I is adopted from [4].

In a functionally pipelined datapath, the total energy E can be modeled as (1), where N is the number of stages, M_i is the number of resources in stage i , and $\alpha_{i,j}$ is the switching activity

$$E = \sum_{i=1}^N E_i = \sum_{i=1}^N \sum_{j=1}^{M_i} C_{i,j} V_{i,j}^2 \alpha_{i,j}. \quad (1)$$

For simplicity, the examples here consider energy dissipated by functional units only. For $S1$, if the clock period of stage 1 remains as 30 ns, the energy is $E1 = 2202 \times 9 + 57 \times 8 = 20274 \text{ pJ}$. However, since the clock period can be 50 ns now, there is longer clock cycle slack in this stage, which will enable the voltages of the resources to be lowered. Fig. 1(b) shows a feasible solution, represented by $S2$ below, where the superscript on a node number indicates the supply voltage assigned to the node while the superscript on a cstep or a stage applies to all the nodes of the cstep or the stage, respectively. The default supply voltage is 5.0 V

$$S2 = \{(17, 16), (), (19, 15), (), (14, 20, 21), (23)\}^{3.3 \text{ V}} \\ + \{(13^{3.3 \text{ V}}, 18), (), (22), (24), (26), (25), \\ (27), (28), (), (29)\}.$$

In Fig. 1(b), all operations in stage 1 can be assigned with 3.3 V without violating the timing constraint. The voltage of operation 13 in stage 2 is also reduced to 3.3 V, which is now possible due to looser dependency constraints in stage 2 than in the original DFG. If operation 13 of the original DFG in Fig. 1(a) was assigned with 3.3 V, there would be one 3.3-V multiplier and one 5.0-V multiplier. In this case, either operation 20 or operation 14 has to go to cstep 7, breaking the resource constraints in cstep 7. This situation does not occur after the partition, as shown in Fig. 1(b). The energy of $S2$ is $E2 = (960 \times 6 + 25 \times 2) + (960 \times 1 + 2202 \times 2 + 57 \times 6) = 11516 \text{ pJ}$, giving a 43.2% reduction over $E1$.

C. Influence of Stage Partitioning and Scheduling

Different stage partitions result in different frequency scaling in the stages, which affects the multivoltage assignment. Consider a different partition scheme $S3$

$$S3 = \{(17, 16^{3.3 \text{ V}}), (), (15, 19), (), \\ (21, 20, 14^{3.3 \text{ V}}), (23), (13, 18), ()\} \\ + \{(22), (24), (26), (25), (27), (28), (), (29)\}.$$

With this partition, the clock period can be extended to $300 \text{ ns}/8 = 37.5 \text{ ns}$ for both stages. After the voltage scaling is performed, stage

1 requires one 3.3-V multiplier, two 5.0-V multipliers, and one 5.0-V ALU, while stage 2 needs one 5.0-V multiplier and one 5.0-V ALU. The total resources are still four multipliers and two ALUs. The energy of $S3$ is $E3 = (960 \times 2 + 2202 \times 6 + 57 \times 2) + (2202 \times 1 + 57 \times 6) = 17790 \text{ pJ}$. This achieves only 12.3% reduction when compared to $E1$, which is not as significant as in $E2$.

Another factor that influences the energy reduction is the initial schedule. Consider an alternative initial schedule

$$S_INIT2 = \{(15, 17, 14), (), (19, 20, 16), (), (13, 18), (), (21), \\ (23), (25), (27), (28, 22), (24), (26), (29)\}.$$

One possible partition scheme with three pipelined stages is

$$S4 = \{(17, 14, 15), (20, 16, 19), (18, 13)\}^{3.3 \text{ V}} \\ + \{(21), (23), (25)\}^{2.4 \text{ V}} \\ + \{(27^{3.3 \text{ V}}), (22^{3.3 \text{ V}}, 28^{2.4 \text{ V}}), \\ (24^{3.3 \text{ V}}), (26^{3.3 \text{ V}}), (29^{3.3 \text{ V}})\}.$$

Stage 1 has three 3.3-V multipliers and a clock period of $300 \text{ ns}/3 = 100 \text{ ns}$. Stage 2 has one 2.4-V ALU and a clock period of 100 ns. Stage 3 has one 2.4-V multiplier and one 3.3-V ALU and a clock period of 60 ns. The energy of $S4$ is $E4 = (960 \times 8) + (13 \times 3) + (507 \times 1 + 25 \times 5) = 8351 \text{ pJ}$, which is 58.8% reduction over $E1$, even better than the 43.2% obtained in $E2$.

Therefore, the three major factors influence the energy consumption in a functionally pipelined datapath are the initial schedule, the stage partition, and the multivoltage assignment.

III. ALGORITHM FORMULATION

MuVoF is proposed to optimize the above three factors in an integrated manner. It consists of two main processes: a core process and an iterative process as described below.

A. The Core Process

Given a scheduled DFG, a resource constraint, and a throughput constraint, the core process partitions the DFG into several pipelined stages to optimize the clock periods and performs multivoltage assignment to reduce energy.

1) *Stage Partition*: The objective of the stage partitioning is to divide the operations into a number of pipelined stages such that each stage has the maximum clock period, while still satisfying the resource and throughput constraints. An optimal partition is the one that allows the clock period to be extended maximally such that the energy has the best opportunity to be reduced via voltage reduction. The stage partition is given in Algorithm 1.

// Algorithm 1: Stage Partition

// Input: a scheduled DFG (G), resource constraint (R),
// cycle-time (T), throughput constraint (TP) and (N).
// number of csteps (N).
// Output: a functionally pipelined scheduled DFG, with
// different clock frequencies in different stages.

```
1:  current_cost = ∞;
2:  s0 = initial_partition (); // initial empty partition
3:  s_first = append_to_partition (s0, 1);
4:  S = {s_first};
5:  while (S ≠ Φ){
```

```

6:   select an element  $s \in S$ ;
7:    $S = S \setminus \{s\}$ ;
   // first branch
8:    $s1 = \text{append\_to\_partition}(s, s.\text{step} + 1)$ ;
   // if satisfy throughput constraints
9:   if ( $s1.\text{current\_stage\_len} * T \leq 1/TP$ ) {
10:    if ( $s1.\text{low\_bound} < \text{current\_cost}$ ) {
11:       $S = S \cup \{s1\}$ ;
12:      if ( $s1.\text{step} == N$ ) {
13:         $\text{current\_best} = s1$ ;
14:         $\text{current\_cost} = s1.\text{low\_bound}$ ;
15:      } }
   // second branch
16:    $s2 = \text{start\_new\_partition}(s, s.\text{step} + 1)$ ;
   // if satisfy resource constraints
17:   if ( $s.\text{resources} + \text{resource\_usage}(s.\text{step} + 1) \leq R$ ) {
18:     if ( $s2.\text{low\_bound} < \text{current\_cost}$ ) {
19:        $S = S \cup \{s2\}$ ;
20:       if ( $s2.\text{step} == N$ ) {
21:          $\text{current\_best} = s2$ ;
22:          $\text{current\_cost} = s2.\text{low\_bound}$ ;
23:       } } }
   // perform clock period extension
24:    $\text{update\_stage\_clock}(\text{current\_best})$ .

```

The process adopts a binary branch-and-bound approach. It starts with an empty partition (line 2) that has no cstep processed. The csteps are then processed from the first to the last sequentially. A solution is defined as an intermediate partition, in which the stages of some csteps have been determined while the others are not yet fixed. Given a solution, for each cstep not yet processed, there are two options that form the two branches of the binary branch-and-bound algorithm. The first option (lines 8–15) is to put the cstep into the current stage (line 8). This option increases the latency of the current stage, and thus the throughput constraints might be violated. This is the first bound condition for the first branch (line 9). The second option (lines 16–23) is to put the cstep into a new stage (line 16). This option increases the resources because each pipelined stage consumes its own resources. Thus the resource constraints might be violated and this is the first bound condition for the second branch (line 17).

If no constraint is violated when either the first or the second option is chosen, the lower bound cost of the new solution is calculated. When calculating the cost of a solution s , a dummy solution s' is derived from s , assuming that each unprocessed cstep occupies one separate stage. The cost of s' is the second bound condition for both branches. If it is less than the current cost, the new solution is added to the solution set. We define the cost function as (2), where N is the number of stages, M_i is the number of resources in stage i , $E_{i,j}$ is the energy required for executing the operations, and f_i is the clock frequency of stage i . The cost function enforces that energy-intensive operations (such as multi-operations) are to be assigned to the stages that have lower frequency,

so that they have the best chance to be assigned to the low-voltage resources in the multivoltage assignment process described later

$$\text{cost} = \sum_{i=1}^N \sum_{j=1}^{M_i} E_{i,j} f_i. \quad (2)$$

During the cost calculation, a clock period extension process is performed on each stage. The types of extension include the following: 1) direct extension, which directly extends the clock period to its maximum under the throughput constraint; 2) step merging, which merges consecutive csteps into one cstep; and 3) deserted step removal, which removes csteps with no operations due to direct extension and step merging.

If a solution derived from option 1 or 2 does not break any constraints, its cost will be evaluated. If the cost is acceptable but there are unprocessed operations left, the solution is added as a new branch. If the solution is complete (i.e., all operations have been processed) and its cost is less than the current best cost, the solution is saved and the current best cost is updated (lines 12 and 20). The solution with the best cost is the final solution, of which the clock period for each stage is extended to the maximum under the constraints (line 24). When extending a clock period, the new period is selected from a predefined clock set to limit the number of clocks synthesized.

2) *Multivoltage Assignment*: After the stage partition, the multivoltage assignment is performed on each stage. The technique is formulated as Algorithm 2, which tries to reduce the supply voltages in a step-by-step manner (line 5) for each resource type. With a set of voltages for a particular resource type, the supply voltages of the resources (FUs) are reduced gradually from the highest to the lowest voltage by the function $\text{reassign}(v_H, v_L, R, L, t)$ in line 6. The function tries to reduce the voltage of some resources of type t from v_H to v_L under the resource constraint R and the latency constraint L and is described as Algorithm 3.

// Algorithm 2: Multivoltage Assignment

// Input: resource types (RT), resource constraint (RC),

// Latency constraint (L) and number of available voltages (M)

```

1:  for each  $t \in \text{RT}$  {
2:     $V = \{v_i | i \in [1, M], v_1 > v_2 > \dots > v_M\}$ ; // voltage set
3:     $R = \text{RC}(t)$ ;
4:    for each  $r \in R \{r.\text{voltage} = v_1\}$  // for each FU
5:      for ( $i = 1; i < M; i++$ )
6:         $\text{reassign}(v_i, v_{i+1}, R, L, t)$ ; // see Algorithm 3
7:  }

```

// Algorithm 3: Voltage Reassignment

// Function $\text{reassign}(v_H, v_L, R, L, t)$

```

1:   $\text{PR} = \Phi$ ; // resources which have already been processed
2:  Level1: while ( $r \in R \& \& r \notin \text{PR} \& \& r.\text{voltage} == v_H$ ) {
3:     $r.\text{voltage} = v_L$ ; // reduce voltage to  $v_L$ 
4:     $\text{PR} = \text{PR} \cup \{r\}$ ;
5:     $\text{RN} = \Phi$ ; // set of nodes whose voltages are reassigned
6:    for ( $\text{cstep} = 1; \text{cstep} \leq L, \text{cstep}++$ ) {
7:       $N = \text{get\_match\_nodes}(\text{cstep}, r, v_H)$ ;
      // try to scale down the voltage of a node

```

```

8:    success = false;
9:    if ( $n \in N$  &&  $\text{min\_voltage}(n) \leq v_L$ ) {
10:         $n.\text{voltage} = v_L$ ;
11:        if (!check_resource( $R, \text{cstep}, \text{cstep} + \text{latency}(n, v_L) - 1$ ))
12:             $n.\text{voltage} = v_H$ ;
13:        else {
14:             $RN = RN \cup \{n\}$ ;
15:            success = true;
16:        }
17:        if (!success) { // check resource of  $v_H$ 
18:            if (!check_resource( $R, \text{cstep}, \text{cstep}$ )) {
19:                for each ( $rn \in RN$ )  $rn.\text{voltage} = v_H$ ;
20:                 $r.\text{voltage} = v_H$ ;
21:                continue Level1;
22:            }

```

In general, Algorithm 3 tries to reduce the supply voltage from v_H to v_L for as many resources as possible (in the loop *Level1*). In line 3, the voltage of a resource r is reduced to v_L . This may break the latency constraint L because the delay of r is increased due to the lower voltage. In addition, as there is now one more resource with voltage v_L and one less resource with voltage v_H , the resource constraint R may be violated in some csteps. Therefore, the constraints R and L must be checked in every cstep (line 6). The resource constraint might be violated in a situation as illustrated by the example below.

Suppose in Cstep A , all the resources in R are assigned with v_H and all are used. In Cstep B , a resource $r \in R$ is reassigned with v_L . This will then cause the violation as now there is one less resource of v_H to be used in Cstep A . This situation is checked by `check_resource( $R, \text{cstep}, \text{cstep}$ )` in line 18. If the check fails, the voltage of all the nodes whose voltage has been reassigned with v_L is changed back to v_H (line 19). One solution to the resource constraint violation is to reduce the voltage of an operation node from v_H to v_L in the same cstep as well. The function `get_match_nodes( $\text{cstep}, r, v_H$ )` finds all the nodes N that match the resource r in the cstep with voltage v_H . Line 9 checks if there exists a node $n \in N$ whose voltage can be reduced to v_L without breaking the dependency and latency constraints. The function `min_voltage( $n$ )` returns the minimum voltage of a node n such that the dependency and latency constraints are still satisfied. If such a node is found, it is reassigned with the voltage v_L (line 10). Note that resource constraint violation might also be introduced in the new life time $[\text{cstep}, \text{cstep} + \text{latency}(n, v_L) - 1]$ of n . This is because in some csteps in the new life time of n , all resources of v_L may have already been used, leaving no resource of v_L to execute n . This is checked in line 11, where the function `latency( $n, v_L$ )` finds the latency of n under the voltage v_L . If the check fails, the voltage of n is rolled back. Otherwise, line 18 is performed to check the possible resource violation due to the reassignment of r with v_L . The checking is on the current cstep only. The following csteps will be checked in the subsequent iterations. In this manner, the voltage reduction is performed on as many resources and nodes as possible while maintaining the constraints throughout.

In short, in the core process, a multifrequency multivoltage functionally pipelined datapath is synthesized from the original scheduled

TABLE I
DELAY AND ENERGY CHARACTERIZATION OF LIBRARY COMPONENTS

FU	5.0V		3.3V		2.4V	
	Delay	Energy	Delay	Energy	Delay	Energy
ALU	25.7	57	45.5	25	76.8	13
Multiplier	54	2202	96.6	960	163.7	507

TABLE II
INFLUENCE OF RESOURCE CONSTRAINTS ON ENERGY (pJ)

Benchmark	TP (ns)	A		B		C		D	
		+	* Energy	+	* Energy	+	* Energy	+	* Energy
IIR	350	1	3 18634	3	3 18405	2	4 8407	2	5 5103
FIR	350	5	4 11025	5	6 10218	6	6 8936	7	7 7201
EWf	450	4	2 13856	6	2 8423	6	3 6791	8	4 4831
Lattice	400	2	3 26257	2	5 11458	3	5 10997	4	6 10012
DCT	450	5	4 30667	6	5 17924	7	5 17924	8	6 16074
FDCT	450	5	4 22583	6	5 15836	7	5 15674	8	6 14057
TFIR	350	3	2 22886	4	3 22740	4	4 10998	5	5 8637
DiffEq	300	2	2 6097	3	3 5833	2	4 5121	4	4 4563
Elliptical	500	4	4 32014	5	5 20649	6	5 20598	7	6 20598

DFG. To further reduce the energy, an iterative process is proposed in the next section.

B. The Iterative Process

The iterative process is based on the simulated annealing technique. It aims to further optimize the energy consumption by searching for alternative schedules of the DFG, hence different stage partitions and multivoltage assignments, leading to better energy consumption. In the process, four types of local transformation are used to iteratively refine the solution. One of the transformations is randomly selected in each iteration of the simulating annealing process. The four transformations are as follows.

- i) Push up: It moves the first cstep of stage _{i} to stage _{$i-1$} .
- ii) Push down: It moves the last cstep of stage _{i} to stage _{$i+1$} .
- iii) Split stage: There are some cases where extra resources are available because the total resource usage decreases after a series of transformations. In these cases, the extra resources are allocated to the stage that can provide the maximum cost reduction and the stage is split into two stages, providing further clock period extension and voltage reduction.
- iv) Reschedule a stage: It reschedules all the operations of a pipelined stage. Working together with the other transformations, it can transform the entire schedule and totally alter the influence of the original schedule on the total energy.

IV. EXPERIMENTAL RESULTS

In our experiments, we used the FUs in Table I. The energy of the level converters is from [4], i.e., the low-to-high converters consume the energy of 178.1, 139.4, and 53.04 pJ for 3.3–5, 2.4–5, and 2.4–3.3 V conversion, respectively.

A. Influence of Resource and Throughput Constraints

In this experiment, four different resource constraints (A to D) were used while the throughput (TP) was fixed. The results obtained are shown in Table II. It can be seen that the energy decreases as the resource constraint is relaxed because the loosen constraints promote

TABLE III
INFLUENCE OF THROUGHPUT CONSTRAINTS ON ENERGY (PJ)

Benchmark	R		TP(A) (ns)	A	B	C	D	E
	+	*						
IIR	2	3	250	18639	18457	18405	8503	8503
FIR	5	5	250	21417	11610	10526	9775	8963
EWf	5	2	350	19013	13668	13668	8411	8309
Lattice	2	4	300	23741	23516	13822	12920	12001
DCT	6	4	350	40917	30822	30694	17936	17741
FDCT	6	4	350	28711	28671	16157	15806	15732
TFIR	3	3	250	23017	22993	22993	10896	10735
DiffEq	2	2	200	13623	11052	5833	5833	5098
Elliptical	4	4	400	41400	39429	32014	31558	21681

TABLE IV
ENERGY OPTIMIZATION

Benchmark	R		TP (ns)	SVSF (pJ)	MuVoF		Run Time (s)
	+	*			# Clocks	Energy (pJ)	
IIR	2	4	270	20274	2	10453	48.4
FIR	6	6	300	25362	3	11074	56.3
EWf	6	2	480	19098	2 *	7926	58.5
Lattice	2	5	300	31284	3	18652	40.3
DCT	6	5	420	41118	3	19970	51.4
FDCT	6	5	390	36714	3	18013	50.9
TFIR	4	4	240	24792	3	15026	39.4
DiffEq	2	3	210	13497	2	7963	41.0
Elliptical	5	5	480	49584	3 *	21777	56.1
Average							50.0

* The number of pipelined stages is one more than the number of clocks.

better stage partition, clock cycle extension, and hence voltage supply reduction.

To study the influence of throughput, the resource constraint (R) was fixed and the throughputs were varied. The results are shown in Table III, where $TP(A)$ is the throughput (data arrival interval) for Experiment A. The throughput was then relaxed in 50-ns steps for each of the subsequent experiments from B to E. It shows that the energy decreases when the throughput is relaxed. The loosen throughput constraints promote the clock period extension and hence voltage supply reduction.

B. Comparisons

To find the improvement of energy achieved by MuVoF and to compare with other published works, we chose some typical resources and find the maximum throughput (TP) in nonpipelining single-voltage synthesis (SVSF) under the resource constraints (R) using list scheduling. We then used the same R and TP for MuVoF. Table IV shows the results obtained. The allowed frequencies are of multiples of ten, starting from 1/30 ns and the maximum number of frequencies allowed is 5. The run time is obtained with MuVoF implemented in Java and running on a 2.4-GHz Pentium PC with 256 MB of RAM. Averagely, the energy is reduced by 50.0%. As the resource constraints implicitly impose a limit on the number of pipelined stages, the number of clocks synthesized is small.

As this is the first attempt on energy optimization for multivoltage multifrequency functionally pipelined datapath, a direct comparison with existing works is not straightforward. MuVoF generally uses more resources but achieves higher throughput. In today's DSM technology, the area occupied by FUs is usually not a main concern while throughput and energy consumption are more critical. Table V lists the average energy reduction comparison with some published

TABLE V
AVERAGE IMPROVEMENT (%) COMPARISONS

	[3]	[5]	[6]	[7]	[8]	[9]	MuVoF
Energy Δ (%)	40.19	43.29	23.3	up to 50	39	up to 46.7	50.0

TABLE VI
BENCHMARK IMPROVEMENT (%) COMPARISONS

Benchmark	[3]	[5]	[6]	[8]	MuVoF
IIR	-	-	24.7	-	48.4
FIR	-	41.55	18.0	33.2	56.3
EWf	44.47	43.01	-	44.2	58.5
DCT	-	43.58	-	30.4	51.4
FDCT	43.44	-	-	-	50.9
DiffEq	40.87	-	-	53	41.0

techniques. Table VI shows the improvement comparison on some benchmarks.

V. CONCLUSION

In this paper, MuVoF, a multivoltage multifrequency behavioral synthesis algorithm for functionally pipelined datapath, is proposed. Given a DFG, the resource, and throughput constraints, it synthesizes a low-energy functionally pipelined datapath. MuVoF consists of a core process and an iterative process. The results obtained from experiments show that MuVoF is effective in reducing energy consumption.

REFERENCES

- [1] C. Lyuh and T. Kim, "High-level synthesis for low power based on network flow method," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 3, pp. 364–375, Jun. 2003.
- [2] X. Xing and C. C. Jong, "A look-ahead synthesis technique with backtracking for switching activity reduction in low power high-level synthesis," *Microelectron. J.*, vol. 38, no. 4–5, pp. 595–605, Apr./May 2007.
- [3] J. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 5, no. 4, pp. 436–443, Dec. 1997.
- [4] S. Mohanty and N. Ranganathan, "Energy efficient scheduling for datapath synthesis," in *Proc. Int. Conf. VLSI Des.*, Jan. 2003, pp. 446–451.
- [5] S. Mohanty and N. Ranganathan, "A framework for energy and transient power reduction during behavioral synthesis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 6, pp. 562–572, Jun. 2004.
- [6] S. Mohanty and N. Ranganathan, "Simultaneous peak and average power minimization during datapath scheduling," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 6, pp. 1157–1165, Jun. 2005.
- [7] C. Papachristou, M. Nourani, and M. Spining, "A multiple clocking scheme for low-power RTL design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 2, pp. 266–276, Jun. 1999.
- [8] A. Manzak and C. Chakrabarti, "A low power scheduling scheme with resources operating at multiple voltages," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 1, pp. 6–14, Feb. 2002.
- [9] D. Chen, J. Cong, and J. Xu, "Optimal simultaneous module and multivoltage assignment for low power," *ACM Trans. Des. Auto. Electr. Syst.*, vol. 11, no. 2, pp. 362–386, Apr. 2006.