Algorithmic Transformations and Peak Power Constraint Applied to Multiple-Voltage Low-Power VLSI Signal Processing

Hsueh-Chih Yang and Lan-Rong Dung
Department of Electrical and Control Engineering
National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.
1001 Ta Hsueh Road, Hsinchu, 30010
TAIWAN, R.O.C.
yang.ece91g@nctu.edu.tw and lennon@cn.nctu.edu.tw

Abstract: - We present a multiple-voltage high-level synthesis methodology that minimizes power dissipation of VLSI signal processing. By applying algorithmic transformations, the proposed approach optimizes the power saving, in terms of the average power and peak power, for DSP applications when the resources and the latency are constrained. Our approach is motivated by the maximization of task mobilities. The mobility is defined as the distance between its as-late-as-possible (ALAP) schedule time and its as-soon-as-possible (ASAP) schedule time. The increase of mobilities may raise the possibility of assigning tasks to low-voltage components. To earn task mobilities, we use loop shrinking, retiming and unfolding techniques. The loop shrinking can reduce the iteration period bound (IPB), while the others are employed for shortening the minimum achieved sample period (MASP) as much as possible. The minimization of MASP implies high task mobilities. Thereafter, we can assign tasks with high mobilities to low-voltage components and minimize energy dissipation under resource and latency constraints. With considering the overhead of level conversion and the minimization of peak power, the proposed methodology has low complexity and can achieve significant power reduction.

Key-Words: - High-level synthesis, low-power synthesis, peak power, multiple-voltage scheduling, resource and latency constrained scheduling, algorithmic transformation

1 Introduction

With increasing demand of portable devices, the reduction of power consumption has become the essential issue in VLSI design. [1] and [2] described that decisions during high-level synthesis (HLS) have a profound impact on the power consumption of the final design. Hence, [2], [3], [4], and [5] have addressed on power saving techniques, such as voltage scaling, capacitance reduction and switching minimization for HLS. However, these papers are based on a single voltage supply for power minimization and cannot take full advantage of available schedule slacks to reduce the voltage. Therefore, the use of multiple supply voltages becomes very attractive to low power design recently, such as [6], [7], [8], [9], [10], [12], [13], and [14]. The idea is to assign non-critical tasks to low-voltage components and execute time-critical tasks at higher supply voltage. In [2], [8], [12], and [13], the multiple-voltage scheduling method for power optimization of HLS using either integer linear programming (ILP) or dynamic programming were presented. However, both approaches have pseudo-polynomial or even exponential time complexity. In [10], Shiue and Chakrabarti present a list-based multiple-voltage scheduling algorithm with polynomial-time complexity. The algorithm is driven by three parameters: depth, mobility, and switching capacitance. With considering the level converters, [10] provides effective resource-constrained and latency-constrained schemes for multiple-voltage HLS. From Chakrabarti's group, later on, [14] uses the Lagrange multiplier method to find the optimal solution of multiple-voltage scheduling under both resource and latency constraints.

The papers mentioned above have presented efficient scheduling for multiple-voltage HLS. Yet, few papers have considered the effect of algorithmic transformations on multiple-voltage power minimization, which forms the major motivation for this work. [11] exploited on algorithmic transformations for multiple-voltage HLS and present an efficient approach to minimize power consumption under resource and latency constraints. The main concept is to change the computational structures by transformations and make mobility of

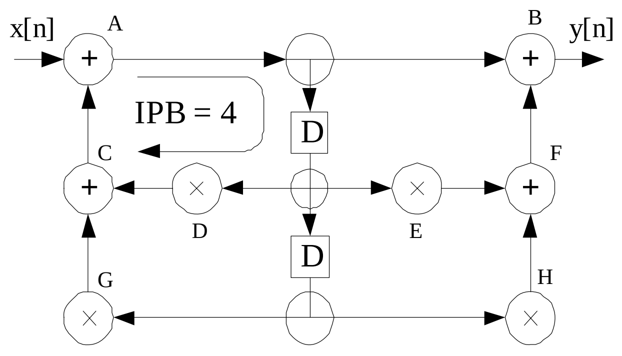

Figure 1: FSFG of second-order IIR filter.

each task in fully-specified flow graph (FSFG) as high as possible. The mobility means the ability to schedule the starting time of a task. It is defined as the distance between its as-late-as-possible (ALAP) schedule time and its as-soon-as-possible (ASAP) schedule time. Obviously, the increase of mobilities may raise the possibility of assigning tasks to low-voltage components. To earn task mobilities, we use loop shrinking, retiming and unfolding techniques. Furthermore, this paper provides thorough analysis on different combinations of algorithmic transformations.

In low-power designs for battery-driven portable applications, the peak power drives the transient characteristic of the CMOS circuit. Therefore, in this work, the minimization of the peak power is another important consideration. Following the optimization of average power dissipation, we suppress the peak power dissipation by the barrier-driven approach. The barrier-driven approach gradually compresses the task schedulability until no further legal scheduling can be found.
As the results, our approach can achieve significant power reduction. In the case of the third-order IIR filter, the proposed methodology can save up to 54.77% of power consumption while the resources running at 5V and 3.3V under the latency constraint of $1.5T_c$ and resource constraints of {1, 1, 1, 1}(one 3.3V multiplier, one 5V multiplier, one 3.3V adder, and one 5V adder).

The rest of the paper is organized as follows. In Section 2, we introduce algorithmic transformations. Section 3 presents the proposed approaches in details. Section 4 shows the experimental results and Section 5 is the conclusion of this work.

2 Overview of Algorithmic Transformations

2.1 Fully-Specified Flow Graph

A deterministic DSP algorithm can be represented by an FSFG. The FSFG describes the relationship between a set of input and output sequences [15]. Fig. 1 shows an FSFG for a second-order IIR filter. In FSFG, the IPB is determined by loops [16] and has been used to measure the performance bound of the implementation of FSFG [16, 17, 18]. The iteration period (IP) for a loop is defined as the total computational latency in the loop divided by the total number of delays. The IPB is the maximum value of IPs and represents the lower bound of MASP. For instance, if a multiplication takes 2 time-units and an addition takes one time-unit, the FSFG shown in Fig. 1 has an IPB of 4 time-units. However, the IPB is not always achieved without using algorithmic transformations. In Fig. 1, for example, the MASP is limited by the critical path G-C-A-B and so equals to 5 time-units. Thus, to obtain the rate-optimal implementation of FSFG, this paper introduces three techniques for the IPB reduction and the minimization of MASP. These algorithmic transformation techniques will be explained in the following subsections.

2.2 Loop Shrinking

Loop shrinking can reconstruct the FSFG to obtain the optimal IPB for loops. Fig. 2 is an example of loop shrinking. Fig. 2(a) has a chain of two additions within the loop. According to the associativity of addition, the function a + (b + c) in Fig. 2(a) is equivalent to the function (a + b) + c in Fig. 2(b). Obviously, the critical loop, L_1, has been shrunk in Fig. 2(b) and IP is reduced as well. Therefore, we can perform loop shrinking on critical loop, which has the maximum IP, to reduce the IPB while the functionality of FSFG keeps the same. In case each task takes one time-unit to execute, the IPB can be reduced from 3 time-units to 2 time-units.

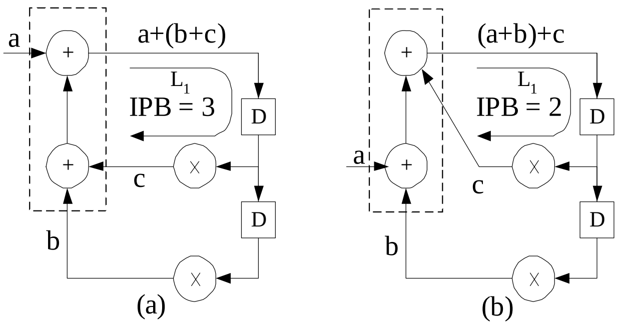

Figure 2: Loop shrinking of second-order IIR. (a) The original FSFG.(b) The equivalent FSFG.

2.3 Retiming and Unfolding

The optimal IPB does not guarantee the optimal rate. Retiming is a process that may help making MASP equal to IPB. With the delay transfer or nodal transfer, it is possible to make MASP optimized. Unfortunately, the retiming technique might not guarantee the optimal MASP. Fig. 3(a), for example, the MASP can not be achieved by retiming since node A requires 20 time-units to execute. To achieve the optimal rate, [18] presents the unfolding technique. Instead of describing one iteration of the computation in the form of a recursive loop, unfolding by a factor P implies P consecutive iterations. If the original FSFG has N tasks, the P-unfolded FSFG has P×N tasks, and the IPB is P times larger than that of the original FSFG. Fig. 3(b) illustrates the result of 2-unfolded FSFG in Fig. 3(a).

In Fig. 3(b), the total number of delays, however, remains unchanged and precedence constraints are also not violated. The unfolding technique can obtain the rate-optimal static schedules.

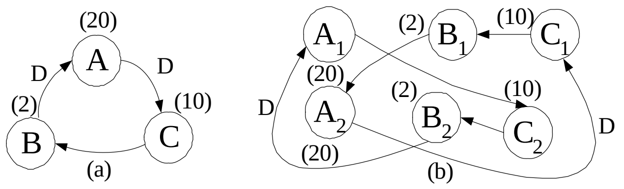

Figure 3: Unfolding result. (a) An example of FSFG that cannot achieve IPB. (b) A rate-optimal FSFG using unfolding.

3 Proposed Approach

We proposed a multiple-voltage HLS for the low power DSP realization. The HLS algorithm first applies the loop shrinking technique for IPB reduction, and then minimizes the MASP using the retiming and unfolding. Once the MASP is optimized, the mobilities can be enlarged and the scheduler will have more room to schedule low-voltage components.

3.1 Multi-Voltage HLS Algorithm

```
SCHEDULE(FSFG, $R_u$, $T_u$, L, EnergyTb, LCTb)
{
g = Read(FSFG, $R_u$, $T_u$, L, EnergyTb, LCTb);
g1 = Shrink(graph);
if (MASP = IPB)
S = MVS(g1, $R_u$, $T_u$, L);
else{
        g2 = Minimize_MASP(g1);
        S = MVS(g2, $R_u$, $T_u$, L);
    }
S = LC_refine(S);
Report(S);
}
```

The inputs to our methodology are an FSFG, a resource constraint R_u, a latency constraint T_u, a number of voltage levels L, an energy table of multiple voltages *EnergyTb*, an energy table of level converters *LCTb*, and the outputs are the voltage assignment, start time, and end time of each node and the total power consumption of the scheduling if the legal scheduling exists. In a nutshell, the proposed resource and latency constrained algorithm operates in four passes. In the first pass, the input file specifies R_u, T_u, and the operations within the FSFG. Once having the input information, we use loop shrinking technique to reduce the IPB. In the second pass, we compute the MASP to check whether it matches the IPB or not. If the MASP matches the IPB, the graph will be sent to the third pass. If the MASP is not equal to the IPB yet, the minimization of MASP can be achieved by *Minimize_MASP(graph)* to obtain optimal mobilities under the given resource constraint. In the third pass, *MVS(graph, R_u, T_u, L)*, is used to schedule and assign tasks to the proper scheduling time and components such that the total power/energy consumption is minimum. In the last pass, *LC_refine(S)* refines the schedule of the third pass by considering level converters.

3.2 *Shrink(graph)*

The follows list the loop shrinking steps.

```
Step1 : Calculate initial IPB;
Step2 : Search for the critical loop;
Step3 : Rearrange edges having relation to the
        adjacent addition nodes in the critical loop;
Step4 : Calculate new IPB; If new IPB < initial IPB;
        save the rearranged FSFG and let initial
      IPB equals to new IPB; Otherwise go to Step6;
Step5 : Go to Step2;
Step6 : Loop shrinking ends;
```

Shrink(graph) searches two adjacent addition operations in the critical loop first and then rearrange the associated edges to reduce the number of nodes in the critical loop. The procedure *Shrink(graph)* will repeat Step2 to Step4 until the IPB cannot be improved.

3.3 *Minimize_MASP(graph)*

This subroutine uses retiming or unfolding techniques to obtain MASP and hence optimize mobilities under given timing constraints. The retiming transformation has been implemented by the integer linear programming (ILP) formulation [19]. If the MASP of the retimed FSFG can not

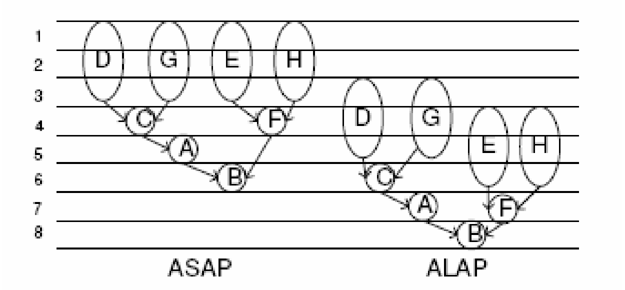

Figure 4: Schedules of second-order IIR before retiming.

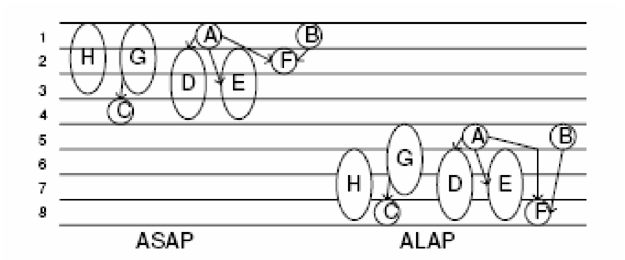

Figure 5: Schedules of second-order IIR after retiming.

achieve the IPB obtained from the *Shrink(graph)* stage, then we apply the unfolding technique to the retimed FSFG to guarantee that MASP matches the IPB. Fig. 4 shows the ASAP and ALAP scheduling result of the original FSFG of the second-order IIR filter. By applying *Minimize_MASP(graph)* subroutine, Fig. 5 and Fig. 6 illustrate the scheduling results obtained by the retiming and the unfolding techniques, respectively.

3.4 *MVS(graph, R_u, T_u, L)*

Fig. 7 shows the flowchart of *MVS(graph, R_u, T_u, L)*, where the index *i* and *j* represent the number of classes among all tasks and voltage levels, respectively. The index *k* represents the number of tasks in the class *c*. In the beginning, all tasks in the FSFG are set to be unmarked to represent the un-scheduled status of each node. Then the program will choose a class of operations, *c*, such as multiplications, according to the effectiveness among all tasks. To obtain maximally power saving, we determine the number *M*, which represents how many number of tasks with the highest effectiveness will be assigned to the lowest voltage resources under the given constraint. The number *M* is defined By $\lfloor T_u / T_c(v(j)) \rfloor$, where $T_c(v(j))$ represents the execution time of the task with the highest effectiveness operating at *v(j)* voltage. Then we can assign tasks by a proposed task-assignment scheme. The scheme is priority-based in that the task with higher priority will have higher opportunity to be assigned to lower voltage resource. So we

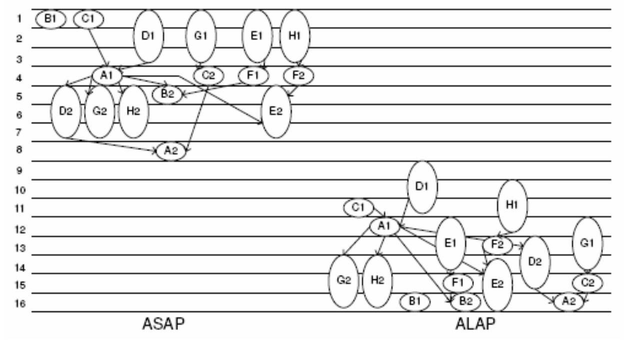

Figure 6: Schedules of second-order IIR after unfolding.

recursively compute the parameter-list including the value of ALAP and ASAP, depth, and the mobility for each task and assign tasks with higher priority to lower voltage resources. Note that the scheduling order of all tasks does not always follow the data precedence, which means we might deal with all the multiplications before all additions, therefore, we use $T_s + T_c(v(j)) \leq T_L$ to check the legal scheduling result of each task, where T_s is the scheduling time and T_L is the end time in the ALAP scheduling result. Once the timing constraint is illegal, the higher voltage resource will be utilized by the increment of the number *j* to make sure the scheduling result is feasible. In addition, the peak power is bounded between the PP_L and the PP_U, where the PP_L and the PP_U present the lower bound and the upper bound of peak power, respectively. These two bounds are defined by the average power consumption of using the lowest and the highest voltages without latency constraints. We explore the bounded space from the middle point to find the minimum peak power solution. Moreover, we can reduce the number *M* if necessary. For instance, according to the energy chart in [20], the multiplications have higher effectiveness than that of additions. If T_u = 10 and R_u = {1, 1, 1, 1}, we will try to assign $(\lfloor 10/4 \rfloor)$ multiplications to 3.3V resources because one 3.3V multiplier takes 4 time-units for the execution. In case this constraint can not be achieved, we must relieve it by resetting the number with *M = M 1.*

3.5 *LC_refine(S)*

After optimizing power consumption using multiple-voltages on data path scheduling, the proposed approach then takes the power consumption of level converters into account and refines the use of level converter. The reason why the optimization of level converters is considered after resource assignment is because the multiple voltage assignment can gain more amount of power

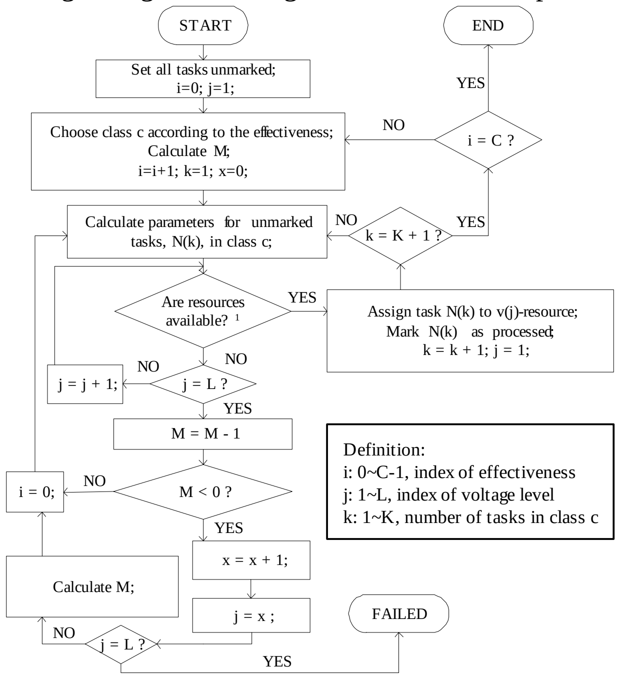
START
END
Set all tasks unmarked; i=0; j=1;
YES
NO
i = C ?
Choose class c according to the effectiveness; Calculate M; i=i+1; k=1; x=0;
Calculate parameters for unmarked tasks, N(k), in class c;
NO
k = K + 1 ?
YES
Are resources available? 1
YES
Assign task N(k) to v(j)-resource; Mark N(k) as processed; k = k + 1; j = 1;
NO
NO
j = j + 1;
j = L ?
YES
M = M - 1
Definition:
i: 0~C-1, index of effectiveness
j: 1~L, index of voltage level
k: 1~K, number of tasks in class c
i = 0;
NO
M < 0 ?
YES
x = x + 1;
Calculate M;
j = x ;
FAILED
NO
j = L ?
YES

Figure 7: Flowchart of multiple voltage scheduling. ([1]In the class *c*, is the resource with *v*(*j*)-voltage available, is the cycle power consumption under the peak power bound, and $T_S + T_C(v(j)) \leq T_L$?)

saving than does the reduction of level converters. From the power models in [20, 9], obviously, the power difference between high-voltage and low-voltage components is larger than power consumption of level converters. One can optimize the power consumption of multiple voltage scheduling by treating level converters and resource assignment simultaneously. [12], for instance, uses ILP to find the power-optimal solution for multiple voltage levels. Their algorithm exhaustively explores all design space and has *$O(N^3)$* time complexity. The exponential complexity makes the multiple voltage scheduling time-consuming. Instead of considering level converters within resource assignment, we separate the reduction of level converters from multiple voltage scheduling to produce comparable results with only polynomial complexity *$O(N)$*. Paper [10] gives the level converter introduction the lowest priority in the list-based algorithm and has polynomial complexity. However, they may not be able to refine inefficient resource assignment by removing level converters. Differently from list-based algorithm, our approach is to remove level converters when their associated resource assignments are inefficient. A resource assignment is called inefficient when the power consumption difference between its high voltage and low voltage components is smaller than the power consumption of level converter. For example,

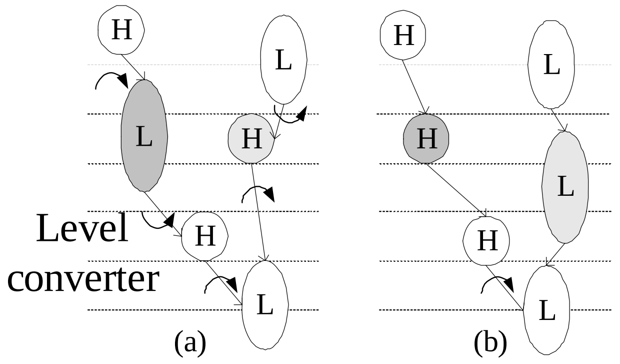

Figure 8: Examples of level converters.

in Fig.8(a), H and L indicate higher-voltage and lower-voltage components, respectively, there are five level converters are required and the power difference between high-voltage and low-voltage components is smaller than low-to-high level converter. We can switch highlighted high-voltage task with highlighted low-voltage task which has been shown in Fig.8(b) to remove requirements of level converters to avoid the power consumption overhead as much as possible.

4 Experimental Results

The proposed algorithm was implemented in C++ and tested with selected benchmark circuits. We tested the scheduling algorithm using the following sets of resource constraints (RC1, RC2, and RC3):
1) number of multipliers: 1 at 5V and 1 at 3.3V; number of adders: 1 at 5V and 1 at 3.3V;
2) number of multipliers: 1 at 5V and 2 at 3.3V; number of adders: 1 at 5V and 2 at 3.3V;
3) number of multipliers: 1 at 5V, 1 at 3.3V, and 1 at 2.4V; number of adders: 1 at 5V, 1 at 3.3V, and 1 at 2.4V;

Our algorithm has high degree of flexibility and can be applied for other compositions of supply voltages. We also assume that multiple power lines are available, and level converters are needed between resources if they operate at different voltages. The number of level converters is not user defined. Moreover, the proposed algorithm tries to reduce the number of level converters to save the power consumption. The energy consumption and the worst case delays of the different function units have been adopted from [20] and the energy dissipation of level converters adopted from [9]. The delay costs of the level converters are absorbed in the worst case delay values. Because we address the problem under timing constraint, energy

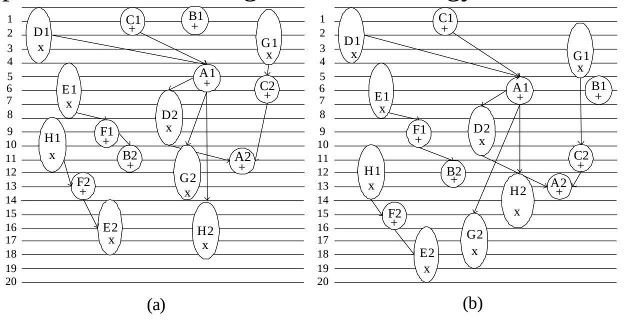

Figure 9: Scheduling results of second-order IIR filter with resource constraint RC2. (a)Without peak power bound. (b)With peak power bound.

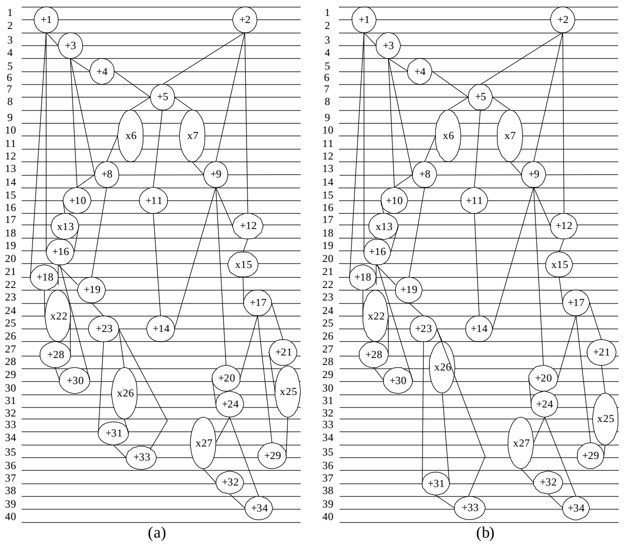

Figure 10: Scheduling results of fifth-order EW filter with resource constraint RC2. (a)Without peak power bound. (b)With peak power bound.

consumption can be referred as power consumption. We assume the clock period is 20ns. So the clock cycle of each different function unit can be computed.

The comparison with AR filter (3rd-order IIR filter) has been listed in Table 1. In this example, it has been found that our algorithm yielded a greater reduction in power consumption. For instance, for the 3rd order IIR filter with the resource constraint RC1, and a timing constraint of 16, we achieve a 40.20% reduction with the unfolding factor P = 3 compared to the 26.00% reduction by using the algorithm in [10]. The power reduction from the proposed algorithm compared with E_5 has been tabulated in Table 3, where E_5 is the power dissipation corresponding to the supply voltage of 5V. E_{alg} is the average power dissipation obtained by our algorithm. Table 2 lists the power consumption and reduction of benchmarks by

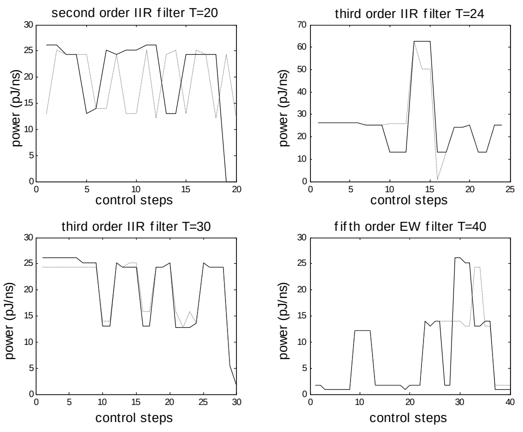

Figure 11: Cycle power consumption of different benchmarks with resource constraint RC2.

applying retiming transformations only. Timing constraints are given for two different values: $1.5T_c$ and $2T_c$, where T_c is the optimal minimum-computation time (critical-path delay) under the given resource constraint. We also plotted the power consumption per cycle, over all the given number of control steps (clock steps) for different benchmarks in Fig. 11. The solid curves correspond to the profile when the scheduling is operated without setting the peak power bound. The profiles with dotted lines correspond to the case when the peak power bound scheme is used. Fig. 9 and Fig. 10 shows the effect of the peak power bound upon on the scheduling results of the second-order IIR and the fifth-order EW filter, respectively.

5 Conclusion

This paper presents a new HLS methodology under resource and latency constraints. The proposed scheme minimizes the power and the peak power consumption by assigning as many nodes to lower voltage components as possible by applying algorithmic transformations. The loop shrinking transformation can reduce the IPB and the unfolding and retiming techniques guarantee the MASP. Doing so, high task mobilities arise the possibility of the assignment of low-voltage resources. As the experimental results, under the timing constraint of $1.5T_c$, the power reduction obtained by the proposed methodology is up to 54.77% with two voltage levels and 59.13% with three voltage levels, respectively. The integration of such a scheduler into a low-power datapath synthesis tool will significantly benefit low-power DSP applications.

Table 1: Comparison results of third-order IIR filter with resource constraint RC1 and a timing constraint of 16 control steps.

Scheduling algorithm	Power (pJ)	% Reduction
E_5	13554	—
[10]	10092	26.00
Retiming applied only	8516	37.16
Proposed	8092	40.20

Table 2: Power consumption and reduction of benchmarks by applying retiming transformations only.

Resource constraint	RC1			RC3	
Benchmark	Latency	E_{alg}	Reduction	E_{alg}	Reduction
second-order IIR filter	$1.5T_c$	6238	30.99%	6096	32.55%
E_5=9039pJ	$2T_c$	5153	44.10%	4900	45.79%
third-order IIR filter	$1.5T_c$	8803	35.05%	8425	37.84%
E_5=13554pJ	$2T_c$	8017	40.85%	8017	40.85%
fifth-order IIR filter	$1.5T_c$	17711	18.36%	17329	20.12%
E_5=21694pJ	$2T_c$	17208	20.68%	16362	24.58%
LMS adaptive filter	$1.5T_c$	12165	22.79%	11918	24.36%
E_5=15756pJ	$2T_c$	10752	31.76%	10240	35.00%
fifth-order EW filter	$1.5T_c$	1330	10.20%	1330	10.20%
E_5=1482pJ	$2T_c$	1188	19.82%	1188	19.82%
2-D fast DCT	$1.5T_c$	23184	26.16%	22543	28.20%
E_5=31398pJ	$2T_c$	21979	30.00%	21693	30.90%

Table 3: Power consumption and reduction of benchmarks by the proposed scheduling algorithm.

Resource constraint	RC1			RC3	
Benchmark	Latency	E_{alg}	Reduction	E_{alg}	Reduction
second-order IIR filter E_5=9039pJ	$1.5T_c$	5035	44.28%	5035	44.28%
	$2T_c$	3484	61.44%	2767	69.39%
third-order IIR filter E_5=13554pJ	$1.5T_c$	6130	54.77%	5540	59.13%
	$2T_c$	5164	61.90%	3226	76.20%
fifth-order IIR filter E_5=21694pJ	$1.5T_c$	14583	32.78%	14285	34.15%
	$2T_c$	13537	37.60%	9311	57.08%
LMS adaptive filter E_5=15756pJ	$1.5T_c$	10698	32.10%	10197	35.28%
	$2T_c$	8902	43.50%	7651	51.44%
fifth-order EW filter E_5=1482pJ	$1.5T_c$	1330	10.20%	1330	10.20%
	$2T_c$	1188	19.82%	1188	19.82%

2-D fast DCT E_5=31398pJ	$1.5T_c$	18955	39.63%	17978	42.74%
	$2T_c$	17649	43.79%	16917	46.12%

References:

[1] O.S. Unsal and I. Koren, "System-level power-aware design techniques in real-time systems," in Proceedings of the IEEE, vol.91, no.7, pp.1055-1069, July 2003.

[2] A.P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R.W. Brodersen, "Optimizing power using transformations," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.14, pp.12-31, Jan. 1995.

[3] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low power cmos digital design," *IEEE J. Solid-State Circuits*, vol.27, pp.473-484, April 1992.

[4] A. Raghunathan and N.K. Jha, "Behavioral synthesis for low power," *IEEE Int. Conf. Computer Design: VLSI in Computer and Processors*, pp.318-322, Oct. 1994.

[5] A. Raghunathan and N.K. Jha, "An iterative improvement algorithm for low power data path synthesis," *IEEE/ACM Int. Conf. Computer-Aided Design*, pp.597-602, Nov. 1995.

[6] S. Raje and M. Sarrafzadeh, "Scheduling with two voltages under resource constraints," *tech. rep., Dept. Elect. Eng. Comput. Sci.*, Northwestern Univ. Evanston, IL, 1995.

[7] M. Takahashi, M. Hamada, T. Nishikawa, H. Arakida, T. Fujita, F. Hatori, S. Mita, K. Suzuki, A. Chiba, T. Terazawa, T. Kuroda, and T. Furuyama, "A 60-mw mpeg4 video codec using clustered voltage scaling with variable supply-voltage scheme," *IEEE J. Solid-State Circuits*, vol.33, pp.1772-1780, Nov. 1998.

[8] M. Sarrafzadeh and S. Raje, "Scheduling with multiple voltages under resource constraints," *IEEE Int. Sym. on Circuits and Systems*, pp.350-353, May 1999.

[9] J.M. Change and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Trans.VLSI Syst.*, vol.5, pp.436-443, Dec. 1997.

[10] W.T. Shiue and C. Chakrabarti, "Low power scheduling with resources operating at multiple voltages," *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, vol.47, pp.536-543, June 2000.

[11] Hsueh-Chih Yang and Lan-Rong Dung, "On multiple-voltage high-level synthesis using algorithmic transformations," in *Proceedings of the ASP-DAC*, pp.872-876, Jan. 2005.

[12] M.C. Johnson and K. Roy, "Datapath scheduling with multiple supply voltages and level converters," *ACM Trans. Design Automation Electronic Syst.*, pp.227-248, July 1997.

[13] Y.R. Lin, C.T. Hwang, and A.C.H. Wu, "Scheduling techniques for variable voltage low power design," *ACM Trans. Design Automation Electronic Syst.*, pp.81-97, April 1997.

[14] A. Manzak and C. Chakrabarti, "A low power scheduling scheme with resources operating at multiple voltages," *IEEE Trans. VLSI Syst.*, vol.10, pp.6-14, Feb. 2002.

[15] V.K. Madisetti and B.A. Curtis, "A quantitative methodology for rapid prototyping and high-level synthesis of signal processing algorithms," *IEEE Trans. Signal Processing*, vol.42, no.11, pp.3188-3208, November 1994.

[16] V.K. Madisetti, *VLSI Digital Signal Processors*, Ch. 6, Butterworth-Heinemann, 1995.

[17] T. Barnwell and C. Hodges, "Optimal implementation of signal flow graphs on synchronous multiprocessors," *IEEE Int. Conf. Parallel Processing*, pp.90-95, August 1982.

[18] K. Parhi and D. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," *IEEE Trans. Computers*, vol.40, pp.178-195, Feb. 1991.

[19] 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 of Integrated Circuits and Systems*, vol.10, pp.464-475, April 1991.

[20] S.P. Mohanty, N. Ranganathan, and V. Krishna, "Datapath scheduling using dynamic frequency clocking," *IEEE Computer Society Sym. on VLSI*, pp.58-63, April 2002.