Datapath Scheduling using Dynamic Frequency Clocking

Saraju P. Mohanty
Department of Comp Sc and Engg
University of South Florida
Tampa, FL 33620
smohanty@csee.usf.edu

N. Ranganathan
Center for Microelectronics Research
University of South Florida
Tampa, FL 33620
ranganat@csee.usf.edu

V. Krishna
Agilent Technology
Palo Alto, CA 94303
vamsi@labs.agilent.com

Abstract

In this paper, we describe a new datapath scheduling algorithm called DFCS based on the concept of dynamic frequency clocking. In dynamic frequency clocking scheme, all functional units in the datapath are driven by a single clock line that switches frequency dynamically at run time. The algorithm schedules lower frequency operators at earlier steps and delays higher frequency operators to later steps. Next, it regroups some of the higher frequency operators with low frequency operators so as to meet the time constraint. During this phase, DFCS assignes the frequency for each cycle and the functional unit with the corresponding voltage. The algorithm has been applied to various high level synthesis benchmark circuits under different time constraints. The experimental results show that using three supply voltage levels $(5.0V, 3.3V, 2.4V)$ *and time constraints* $(\{1.5, 1.75\ and\ 2.0\} *$ *the critical path delay), average energy savings in the range of* 46% *to* 68% *is obtained with respect to using a single-frequency and single-voltage scheme.*

1 Introduction

High-level synthesis is the transformation from a behavioral specification of a system to its RTL structure specification [1]. The essential tasks involved in synthesis are scheduling, allocation, binding and clock selection. The need for low power synthesis is driven by several factors such as [6]: (1) demand of portable systems (battery life), (2) thermal considerations (cooling and packaging), (3) environmental concerns (natural resources), and (4) reliability issues. The average power dissipation is a concern for the first three factors, whereas, peak power is the critical design concern for reliability. During synthesis, to increase battery life, power-delay-product has to be minimized, whereas, to increase battery life alongwith delay reduction, energy-delay-product should be minimized.

The three equations for a CMOS circuit are [6, 8] as follows. Energy dissipation per operation,

$$E = C_{eff} V_{dd}^2 \tag{1}$$

where, C_{eff} is the effective switched capacitance and V_{dd} is the supply voltage. C_{eff} depends not only on the circuit structure but also on the input pattern applied to the system. With frequency f, the power dissipation for the operation is

$$P = C_{eff} V_{dd}^2 f \tag{2}$$

The delay in a device (t_d) that determines the maximum frequency (f_{max}) or the clock cycle time (T) is

$$t_d = k \frac{V_{dd}}{(V_{dd} - V_T)^\alpha} \tag{3}$$

where, V_T is the threshold voltage, α is a technology dependent factor and k is a constant. From the above three equations, the followings can be deduced : (1) by reducing V_{dd} both power and energy can be saved while compromising with performance (delay), (2) slowing down the CPU by reducing f will save power but not energy, and (3) varying frequency as well as voltage will save energy and power while maintaining performance. This forms the basis for our approach. We generate a datapath schedule that will operate at different frequencies at different cycles based on dynamic frequency clocking mechanism developed in [5, 12].

Synthesis techniques incorporating various low power features have been developed by numerous researchers. The HYPER-LP synthesis system [4], uses parallelization and pipelining to reduce power consumption. The SCALP algorithm uses both supply voltage scaling and capacitance reduction [11]. In [8], ILP formulations (called MOVER) of multiple supply voltage scheduling problem are given which handle timing and resource constraints. The authors in [15] present a resource constrained scheduling algorithm that helps in reducing power using multiple supply voltages. A scheduling algorithm using "shut-down" technique, multiplexor reordering and pipelining has been developed in [7]. A dynamic programming technique for

scheduling is presented in [10]. The authors in [14] combine variable voltage scheme and clustured voltage scaling to reduce power. A heuristic scheduling algorithm based on dynamic frequency clocking and multiple voltages is introduced in [16]. Two time and resource constrained multiple voltage scheduling technique are proposed in [17]. In [18], multiple voltage resource and latency constrained scheduling is discussed where power reduction is achieved by reducing switching activity as well as by reducing power consumption of level converters.

In this paper, we propose a scheduling algorithm called Dynamic Frequency Clocking Scheduling (DFCS) using dynamic frequency concept [5, 12]. Initially, DFCS groups the operations in a control step such that the functional units with same maximum frequency can be operated concurrently. Later, DFCS reduces the energy of the initial schedule by regrouping the operations and using multiple voltages without violating the time constraints. The algorithm has been applied to various high level synthesis benchmarks under different time constraints. The experimental results show that using three supply voltage levels $(5.0V, 3.3V, 2.4V)$, an average energy saving of 46% to 68% (with the time constraint of 1.5 to 2.0 times the critical path) is obtained as compared to using a single-frequency clocking scheme with a single supply voltage. DFCS is suitable for data flow intensive applications such as DSP, image and video processing.

2 Dynamic Frequency Clocking and Energy Saving

In dynamic frequency clocking, the clock frequency is varied on-the-fly based on the functional units active in that cycle. In this clocking scheme, all the units are clocked by a single clock line which switches at run-time. A clocking mechanism that varies the clock frequency dynamically has been shown to improve the execution time as compared to using a uni-frequency global clock [5, 12]. Fig. 1 shows the unifrequency and dynamic frequency diagrams. The Dynamic Clocking Unit (DCU) uses a clock divider strategy to generate frequencies $(f_{base}/2n)$ which are submultiples of the base frequency. The base frequency f_{base} is the maximum frequency of any functional unit (FU) at the maximum supply voltage. A value n is loaded as an input to the DCU which comes from controller. The scheme for dynamic frequency generation is shown in Fig. 2.

As discussed in section 1 with reference to the equations 1- 3, frequency scaling helps in reducing power, but not energy. The frequency reduction creates opppertunity to operate the different fuctional units at different voltages, which in turn helps in energy reduction. DFCS finds the appropriate clock width (frequency) for the cycle and the

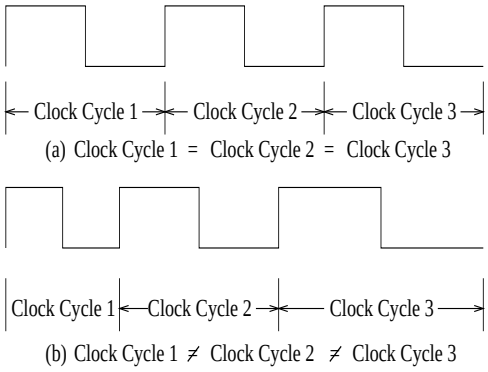

Figure 1: (a) Single frequency (b) Dynamic frequency

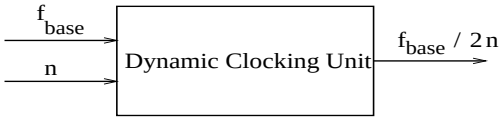

Figure 2: Scheme for dynamic frequency generation

operating voltage of each functional unit operating at that cycle.

Let us consider the following example data flow graph (DFG) shown in Fig. 3. Let t_a and t_m be the delays

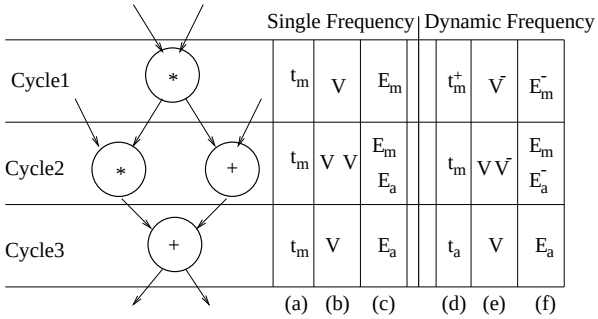

Figure 3: Example DFG

of the adder and the multiplier respectively at the maximum supply voltage V. The DFG is sheduled to three control steps. In the case of single frequency, each cycle has clock width dictated by the slowest operator delay t_m. So the total power consumption is given by $E_{single} = 2E_m + 2E_a$ and the total delay is $t_{single} = 3t_m$. In case of dynamic frequency clocking, we have the flexiblity of changing the clock width of each cycle. Say, we fix the clock cycle width for 3rd cycle at t_a which is smaller than t_m; this allows us to increase the clock width of some other cycles from t_m to some t_m^+ without violating the time constraints. In this case, the total delay time is $t_{dynamic} = t_m^+ + t_m + t_a$ and we have $t_{dynamic} \approx t_{single}$. We exploit these time slacks available at different control steps for different func-

tional units and reduce the voltage appropriately. This helps in reducing the energy consumption of some units and as a whole. So the new energy consumption by the DFG is given by $E_{dynamic} = E_m + E_a + E_m^- + E_a^-$; since E_m^- and E_a^- are some energy values less than E_m and E_a we have $E_{dynamic} < E_{single}$; thus we save energy. In turn, we also reduce average power consumption $E_{dynamic}/t_{dynamic}$ (if, $t_{dynamic} > t_{single}$).

3 Datapath Model

The architecture model consists of a datapath, a controller and a DCU as shown in Fig. 4. The datapath consists

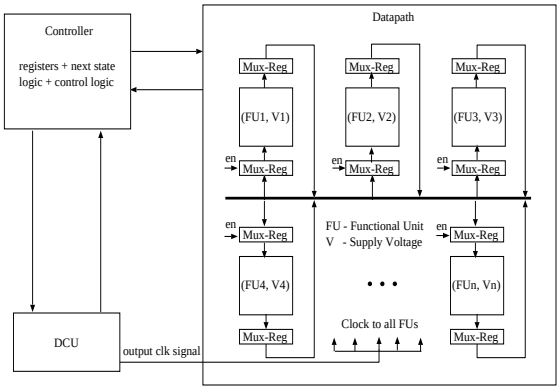

Figure 4: Target architecture

of n functional units (FUs) with registers and multiplexors. The FUs perform single-cycle operations. A controller decides which FUs are active in each control step and those that are not active are disabled. The frequency is changed dynamically and the supply voltage is assigned from one of the available levels ($5.0V, 3.3V, 2.4V$). The reason behind choosing these voltage levels is that these are used in industrial design. Level converters are required since data is transferred between functional units operating at different voltages. The DCU clocks the FUs based on the units operating at each control step.

3.1 Delay Model and Frequency Selection

The dynamic frequency clocking scheme clocks a control step at a frequency determined by the units operating in that step. The delay of a control step is dependent not only of the functional unit delay but also on the multiplexor and register set-up and propagation delay. The worst case delay of a control step can be written as:
$d_i = 2 * registerdelay + 2 * muxdelay + FUdelay$
where, d_i is the delay of control step i, the register delays include the set-up and propagation delays, and FU delay is the delay of the slowest component in control step i. The

Library Component	Delay Type	Delay (5V)	Delay (3.3V)	Delay (2.4V)
Register	set-up	3.3ns	5.9ns	10.1ns
Register	propagation	3.3ns	5.9ns	10.1ns
Multiplexor	propagation	6.0ns	10.8ns	18.4ns
add/sub	propagation	15.4ns	27.8ns	47.3ns
Multiplier	propagation	43.7ns	78.9ns	134.2ns

Table 1: Delay values for 16-bit components

Component	Delay (5V)	Delay (3.3V)	Delay (2.4V)
add/sub	35.0ns	62.2ns	105.3ns
Maximum Freq	28.5MHz	16.07MHz	9.49MHz
Scaled Down Freq	28MHz	14MHz	7MHz
Multiplier	63.3ns	113.3ns	192.2ns
Maximum Freq	15.8MHz	8.82MHz	5.2MHz
Scaled Down Freq	14MHz	7MHz	4.6MHz

Table 2: Delay values and clock frequencies

worst case delay of a level converter is less than $1ns$ as seen from HSPICE simulations. Using the above delay model, the worst case delays for library components are shown in Table 1 (adopted from [16]). The delay costs of the level converters are absorbed in the worst case delay values.

Table 2 shows the operating clock frequencies of each functional unit at different voltages. We choose a base frequency and scale the other frequencies to a number given by $(f_{base}/2n)$, where, $n = 1, 2, 3, ...$ The base frequency to the DCU is $28MHz$ and it generates the frequencies $14MHz$, $7MHz$ and $4.6MHz$.

3.2 Energy Model

Assuming an activity of 0.5 at the inputs, the average energy dissipated for the components, Mux, 16-bit adder/subtractor and the 16-bit multiplier at different voltages is computed [16] as shown in Table 3 The energy dissipation (pJ) of the 16-bit level converters (from voltage level $V1$ to level $V2$) is shown in Table 4 (adopted from [10]).

4 DFCS Algorithm

The datapath is represented in the form of a data flow graph (DFG) constructed as a sequencing graph. Fig. 5 shows such a graph for the HAL benchmark. The inputs to

Component	Energy (5V)	Energy (3.3V)	Energy (2.4V)
Mux	9pJ	4pJ	2pJ
add/sub	57pJ	25pJ	13pJ
Multiplier	2202pJ	960pJ	507pJ

Table 3: Energy dissipation of functional units

V1-V2	2.4V	3.3V	5.0V
2.4V	-	53.04	139.4
3.3V	21.53	-	178.1
5.0V	22.5	61.4	-

Table 4: Energy dissipation in level converters

the algorithm are an unscheduled data flow graph (UDFG), the scaled down operating frequencies, and the execution time constraint T_c for the whole schedule. From the energy dissipation tables, Tables 3 and 4, it is observed that to get more energy savings the multipliers should be operated at as low frequencies as possible and the adders at as high frequencies as possible. If the adder is operated at low frequencies (hence at low voltage), there may not be energy saving as the energy dissipation in level converter will be more than the saving in adder/subtractor. At the same time, operating adder/subtractor at higher frequency can help in meeting the time constraints. To get maximum energy savings adder/subtractor should not be operated alongwith multipliers in same duty cycle. In cases, when they to operated same cycle to meet the time constraint, energy saving will come from the multipliers not from the adder/subsubtractor.

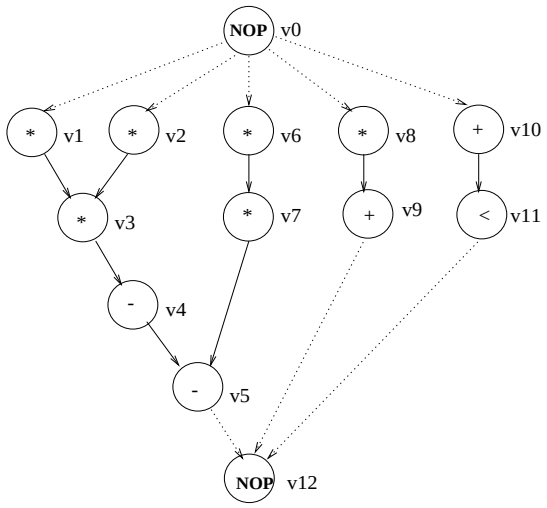

Figure 5: HAL benchmark data flow graph

The algorithm is designed based on these observations. Initially, DFCS uses the concept of dynamic frequency clocking and generates a schedule such that the operators operating at lower frequencies are scheduled at earlier steps/cycles and the operators operating at higher frequencies are scheduled at later steps/cycles. Later on, the DFCS modifies the schedule by moving operations from one step to another with the objective of meeting the time constraint. It then finds appropriate clock cycle width and assignes appropriate voltage.

DFCS (UDFG, FU, Tc)
{
(01) Create PriorityLIST (for all $v_i \in$ UDFG);
(02) DFCSSchedClock$_{v_0}$ = 0; //for source vertex
(03) SchedLIST = v_0; //source vertex is scheduled in 0^{th} cycle
(04) cycle = 1;
(05) while (PriorityLIST $\neq$ NULL) do
(06) {
(07) v_i = TOP of PriorityLIST;
(08) if ($v_i \notin$ SchedLIST and AllPredecessor$_{v_i} \in$ SchedLIST) then
(09) {
(10) if (FrequencyConstraint (cycle)) then
(11) cycle = Max (DFCSSchedClock) + 1;
(12) else schdule in current cycle
(13) DFCSSchedClock$_{v_i}$ = cycle;
(14) PriorityLIST = PriorityLIST - v_i;
(15) SchedLIST = SchedLIST + v_i;
(16) } // end if
(17) } // end do
(18) DFCSClockBound = Max (DFCSSchedClock);
(19) Create CyclePriorityLIST (for all cycle c_i bounded by DFCSClockBound)
(20) CycleFreqLIST = Min (Scaled Down Freq of all $v_i \in c_i$);
(21) T_s = CriticalPathDelay (CycleFreqLIST);
(22) ControlStepIndicator = 1;
(23) while (ControlStepIndicator) do
(24) {
(25) if ($T_s > T_c$) then
(26) {
(27) c_i = FindCycleWithMinALU (for all cycle c_i);
(28) for each $v_i \in c_i$ do
(29) {
(30) DFCSSchedClock$_{Pred(v_i)}$ = DFCSSchedClock$_{Pred(v_i)}$ - 1;
(31) DFCSSchedClock$_{v_i}$ = DFCSSchedClock$_{v_i}$ - 1;
(32) DFCSSchedClock$_{Succ(v_i)}$ = DFCSSchedClock$_{Succ(v_i)}$ - 1;
(33) } // end for
(34) CycleFreqLIST = Min(Scaled Down Freq of all $v_i \in c_i$);
(35) T_s = CriticalPathDelay (CycleFreqLIST);
(36) } // end if
(37) while (more than 50% of cycles having
Multipliers operating at lower freq.) do
(38) {
(39) c_i = TOP of CyclePriorityLIST;
(40) CycleFreqLIST$_{c_i}$ = NextHigher (Scaled Down Freq of all $v_i \in c_i$);
(41) T_s = CriticalPathDelay (CycleFreqLIST);
(42) if ($T_s <= T_c$) then
(43) ControlStepIndicator = 0;
(44) } // end do
(45) } // end do
} // End Algorithm DFCS

The $PriorityLIST$ (line 01) (in Fig.6) is created from the list of all vertices such that the vertex with the lower operating frequency gets the higher priority for scheduling; meaning will be scheduled in a control step before the lower priority vertices.

v0	v1	v2	v3	v6	v7	v8	v4	v5	v9	v10	v11	v12	Vertices
0	1	2	3	4	5	6	7	8	9	10	11	12	Priorities

Figure 6: Vertex priority list

$DFCSchedClock_{v_i}$ (line 02) is a data structure that contains the clock cycle step for vertex v_i. It is initialized to zero for the source vertex. $SchedLIST$ (line 03) is a data structure to maintain the list of vertices already schedules which which is initialised to the source vertex. The while loop (line 05) each time takes the highest priority vertex (line 07) and schedules it in an appropriate cycle checking for the frequency constraint violation if all

its predecessors are already scheduled. The data structure $AllPredecessor_{v_i}$ contains the list of all predecessors of any vertices v_i. The function $FrequencyConstraint$ (line 10) helps in checking the frequency constraint. This makes sure that two vertices operating at different frequncies are not scheduled on same cycle. $DFCSClockBound$ is the number of control steps for the schedule already generated (shown in Fig. 7).

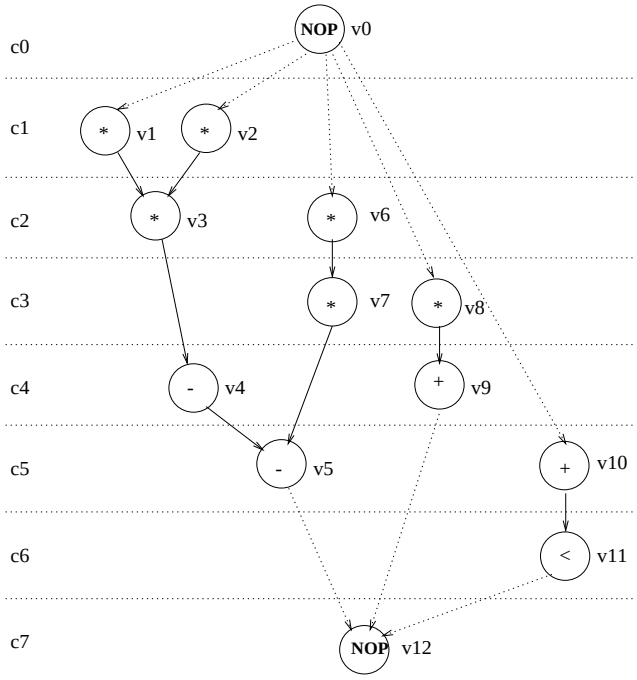

Figure 7: Intermediate scheduled data flow graph

$CyclePriorityLIST$ (line 19) is the priority list (shown in Fig. 8) for the cycles such that the cycle containing more number of multipliers will be operated at lower frequency. The data structure $CycleFreqLIST$ (line 20) is

c0	c5	c4	c1	c3	c2	c6	Cycles
0	1	2	3	4	5	6	Priorities

Figure 8: Cycle priority list

used to store the operating frequency of each cycle. Initially, each cycle is assigned the minimum frequency, and the critical delay of the schedule is found. If the time constraint (line 25) is not satisfied, then we find the cycle is which the number of ALUs used is minimum (line 27). Some of the control steps can be eliminated out of the schedule by moving up the vertices in that cycle (line 28-33). As long as 50% of the cycles are operating at frequencies lower than the maximum scaled down frequency, the clock cycles is assigned to the next higher operating frequency and checked for time constraint satisfied (line 37-44), otherwise, one more control step is eliminated and above steps are repeated. Finally, proper voltage value assigned to the vertices from the Table 2. The final scheduled datapath is shown in Fig. 9. The algorithm also calculates the energy value of the schedule.

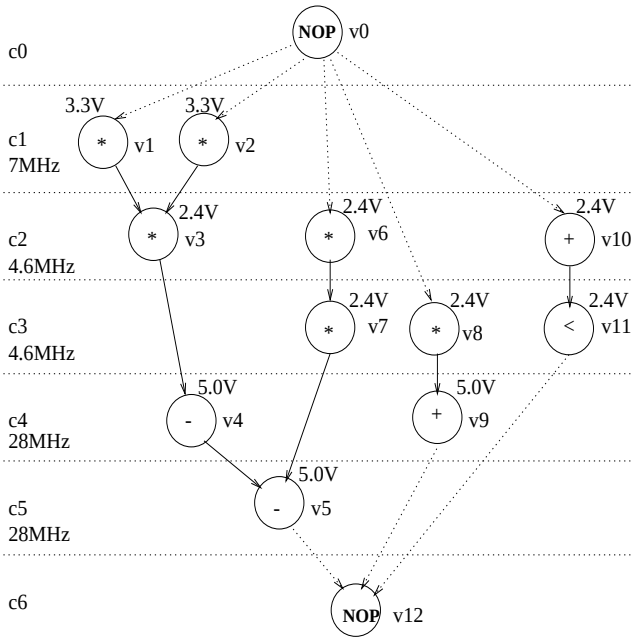

Figure 9: Final scheduled data flow graph

5 Experimental Results

The DFCS algorithm has been implemented in C and applied to selected benchmark circuits. The processor model simulated contains two ALUs (adders/subtractors/comparators) and two MULTs of each operating voltages. The benchmarks used are : Auto-Regressive (AR) filter [13], Band-Pass filter (BPF) [2], Elliptic-Wave filter (EWF) [9], DCT [3], FIR filter [15], HAL differential equation solver. The percentage energy savings is calculated as $(E_{single} - E_{dynamic}) * 100 / E_{single}$. The results for various benchmark circuits are reported in Table 5. The energy savings obtained using different existing multiple voltage based scheduling algorithm is shown in Table 6.

6 Conclusions

This paper introduced a datapath scheduling algorithm based on dynamic frequency clocking. The dynamic frequency clocking scheme could generate enough slack to

Bench. Circuits	Time Cons.	Energy Details for DFCS		
		E_{single}	$E_{dynamic}$	% Savings
AR	$1.5T_{cp}$	36186	21491	40.61
	$1.75T_{cp}$	36186	18139	46.61
	$2.0T_{cp}$	36186	15274	57.79
BPF	$1.5T_{cp}$	27672	15187	45.12
	$1.75T_{cp}$	27672	9350	66.12
	$2.0T_{cp}$	27672	8249	70.19
EWF	$1.5T_{cp}$	19422	12335	36.49
	$1.75T_{cp}$	19422	8814	54.62
	$2.0T_{cp}$	19422	5341	72.50
FDCT	$1.5T_{cp}$	30675	14611	52.37
	$1.75T_{cp}$	30675	14489	52.77
	$2.0T_{cp}$	30675	7714	74.85
FIR	$1.5T_{cp}$	18696	4910	73.74
	$1.75T_{cp}$	18696	4877	73.91
	$2.0T_{cp}$	18696	4820	74.21
HAL	$1.50T_{cp}$	13614	7808	42.64
	$1.75T_{cp}$	13614	6821	49.90
	$2.0T_{cp}$	13614	4449	67.31

Table 5: Eneregy savings using DFCS

Bench.	% Energy Savings					
	DFCS	[10]	[17]	[8]	[18]	[16]
AR	41-58	40-63	16-20	16-59	38-76	3-53
BPF	45-70	-	-	-	-	-
EWF	36-73	44-69	13-32	11-50	13-76	53-54
FDCT	52-75	43-69	-	-	-	-
FIR	74-74	-	16-29	28-73	-	53-53
HAL	43-67	41-61	-	-	22-77	-

Table 6: Eneregy saving using schedulings

apply reduced voltages which in turn saves energy. The functional units are allowed to operate at a frequency governed by its critical path delay. It is observed that using three supply voltage levels $(5.0V, 3.3V, 2.4V)$, an average energy savings of 46% to 68% (with the time constraint of 1.5 to 2.0 times the critical path) is obtained as compared to using a single-frequency clocking scheme with a single supply voltage.

References

[1] M. C. McFarland, A. C. Parker, and R. Camposano, "The High-Level Synthesis of Digital Systems", *Proc. of the IEEE*, Vol.78, No.2, Feb 1990, pp.301-318.

[2] C. A. Papachristou and H. Konuk, "A Linear Program Driven Scheduling and Allocation Method", *Proc. of 27th ACM/IEEE DAC'90*, 1990, pp.77-83.

[3] G. Fetweis, J. Chiu and B. Fraenkel, "A Low-Complexity Bit-Serial DCT/IDCT Architecture", *Conference Record, IEEE Intl. Conf. on Comm.*, 1993, Vol.1, pp.217-221.

[4] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing Power Using Transformations", *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol.14, No.1, Jan 1995, pp.12-31.

[5] N. Ranganathan, N. Vijaykrishnan, and N. Bhavanishankar, "A VLSI Array Architecture with dynamic frequency clocking", *Proc. of ICCD'96*, 1996, pp.137-140.

[6] M. Pedram, "Power Minimization in IC Design : Principle and Applications", *ACM Trans. on Design Automation of Electronic Systems*, Vol.1, No.1, Jan 1996, pp. 3-56.

[7] J. Monteiro, S. Davadas, P. Ashar and A. Mauskar, "Scheduling Techniques to Enable Power Management", *Proc. of 33rd DAC'96*, 1996, pp.349-352.

[8] M. Johnson and K. Roy, "Datapath Scheduling with Multiple Supply Voltages and Level Converters", *ACM Trans. on Design Automation of Electronic Systems*, Vol.2, No.3, July 1997, pp.227-248.

[9] P. Kollig and B.M. Al-Hashimi, "Simultaneous scheduling, allocation and binding in high level synthesis", *Electronics Letters*, 28th August 1997, Vol.33, No.18, pp. 1516-1518.

[10] J. M. Chang and M. Pedram, "Energy minimization using multiple supply voltages", *IEEE Trans. on VLSI Systems*, Vol.5, No.4, Dec 1997, pp.436-443.

[11] A. Raghunathan, N. K. Jha, "SCALP: An Iterative-Improvement-Based Low-Power Data Path Synthesis System", *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol.16, No.11, Nov 1997, pp. 1260-1277.

[12] N. Ranganathan, N. Vijaykrishnan and N. Bhavanishankar, "A linear array processor with dynamic frequency clocking for image processing applications", *IEEE Trans. on CSVT*, Vol.8, No.4, August 1998, pp.435-445.

[13] A. Antola, V. Piuri and M. Sami, "A Low-Redundancy Approach to Semi-Concurrent Error Detection in Data Paths", *Proc. of DATE'98*, 1998, pp.266-272.

[14] M. Hamada, et al., "A Top-Down Low Power Deisgn Technique Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme", *Proc. of IEEE Custom Integrated Cicuits Conf.*, 1998, pp.495-498.

[15] A. Kumar and M. Bayoumi, "Multiple Voltage-Based Scheduling Methodology for Low Power in the High Level Synthesis", *Proc. of ISCAS'99*, 1999, pp.371-379.

[16] V. Krishna, N. Ranganathan and N. Vijaykrishnan, "An Energy Efficient Scheduling Scheme for Signal Processing Applications", *Proc. of the 12th Intl. Conf. on VLSI Design*, 1999, pp.440-445.

[17] M. Sarrafzadeh and S. Raje, "Scheduling with Multiple Voltages under Resource Constraints", *Proc. of ISCAS'99*, 1999, pp.350-353.

[18] W-T. Shiue and C. Chakrabarti, "Low-Power Scheduling with Resources Operating at Multiple Voltages", *IEEE Trans. on CAS-II : Analog and Digital Signal Processing*, Vol.47, No.6, June 2000, pp.536-543.