Energy Efficient Datapath Scheduling using Multiple Voltages and Dynamic Clocking

Saraju P. Mohanty
Dept. of Comp. Science and Eng.
University of North Texas, TX 76203.
E-mail: smohanty@cs.unt.edu

N. Ranganathan
Dept. of Comp. Science and Eng.
University of South Florida, FL 33620.
E-mail: ranganat@csee.usf.edu

Recently, dynamic frequency scaling has been explored at the CPU and system levels for power optimization. Low power datapath scheduling using multiple supply voltages has been well researched. In this work, we develop new datapath scheduling algorithms that use multiple supply voltages and dynamic frequency clocking in a co-ordinated manner in order to reduce energy consumption of datapath circuits. In dynamic frequency clocking, the functional units can be operated at different frequencies depending on the computations occurring within the datapath during a given clock cycle. The strategy is to schedule high energy units, such as multipliers at lower frequencies so that they can be operated at lower voltages to reduce energy consumption and the low energy units, such as adders at higher frequencies, to compensate for speed. The proposed time and resource constrained algorithms have been applied to various high level synthesis benchmark circuits under different time and resource constraints. The experimental results show significant reduction in energy for both the algorithms.

Categories and Subject Descriptors: B.5.1 [**Register-Transfer-Level Implementation**]: Datapath Design; B.5.2 [**Register-Transfer-Level Implementation**]: Automatic Synthesis, Optimization; B.7.1 [**Types and Design Styles**]: VLSI (very large scale integration)

General Terms: Algorithms, Performance, Design, Reliability, Scheduling, Time Complexity

Additional Key Words and Phrases: High-Level synthesis, low-power datapath synthesis, multiple voltage scheduling, time constrained scheduling, resource constrained scheduling and dynamic frequency clocking

1. INTRODUCTION

With the increase in demand for personal computing devices and wireless communications equipment, the demand for synthesizing low power consuming circuits has increased. The need for low power synthesis is driven by several factors, such as [Pedram 1996], demand of portable systems (battery life), thermal considerations (cooling and packaging costs), environmental concerns (use of natural resources), and reliability issues. While energy consumption of a device has to be minimum to increase battery life, the energy-delay-product has to be minimized to increase battery life and to reduce delay, simultaneously.

Let us consider the following equations for a CMOS circuit [Burd and Brodersen 1995;

This research was carried out at the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620.

Pouwelse et al. 2001b] :

—Energy dissipation per operation is

$$E = C_{eff} V_{dd}^2. \quad (1)$$

where, C_{eff} is the effective switched capacitance and V_{dd} is the supply voltage,

—For frequency f, the power dissipation for the operation is

$$P = C_{eff} V_{dd}^2 f. \quad (2)$$

—Further, the critical delay (t_d) in a device that determines the maximum frequency (f_{max}) is below, where, V_T is the threshold voltage, α is a technology dependent factor and k is a constant.

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

From the above three equations, the following can be deduced [Burd and Brodersen 1995; Pouwelse et al. 2001b; Pering et al. 2000; Martin and Siewiorek 2001] :

—Reducing only V_{dd}, both energy and power can be saved at the cost of performance.

—Slowing down the circuit by reducing only f will save power but not energy.

—However, by scaling frequency and voltage in a coordinated manner, both energy and power can be saved while maintaining performance.

The third factor above forms the major motivation for this work. The objective is to generate a datapath schedule that attempts at energy and power reduction without degrading the performance by using multiple voltages and dynamic frequency in a co-ordinated manner.

In this paper, we consider the use of dynamic frequency clocking or frequency scaling alongwith multiple supply voltages for synthesis of low power datapath circuits useful for signal processing applications. In dynamic frequency clocking (hereto, will be referred to as DFC), the functional units could operate at different speeds during each clock cycle depending on the units active in that cycle. We develop two new datapath scheduling algorithms, one referred to as TC-DFC (time constrained) and other referred to as RC-DFC (resource constrained), both of which aim at reducing energy consumption. The resource constraints consist of the number and type of each functional unit, the allowed voltages and frequencies. The time constraint is defined in terms of multiples of the critical path delay of the datapath circuit. RC-DFC minimizes the total energy consumption of the datapath circuit by maximizing the utilization of lower supply voltage resources from the given sets of resources operating at different supply voltages while reducing the time penalty. On the other hand, TC-DFC minimizes the total energy consumption of the datapath circuit without violating the timing constraints assuming that unlimited resources operating at different supply voltages are available. The scheduler will generate a parameter associated with each control step called clock frequency index, denoted as cfi_c for control step c. This parameter is provided to the dynamic clocking unit called DCU which switches the clock based on that parameter.

The paper is organised as follows. Section 2 describes the prior works and Section 3 describes the energy savings and performance improvement possible due to dynamic frequency clocking. Section 4 discusses the target architecture and frequency selection scheme. Sections 5 and 6 present the time constrained scheduling and the resource constrained scheduling algorithms followed by experimental results and conclusions.

2. RELATED WORK

We investigate the use of dynamic frequency clocking alongwith multiple supply voltages, as a means to lower power consumption. We develop high level synthesis scheduling algorithms incorporating both multiple voltage and dynamic clocking. Thus, our discussion of related works will include two broad categories. First, we brief the works involving the use of frequency scaling in the design of general purpose or multipurpose processor architectures, and later, we discuss the low power datapath scheduling works.

Several approaches towards reducing power or energy consumption in both general purpose and special purpose processors have appeared in the literature. A dynamic voltage scaled microprocessor system is presented in [Burd et al. 2000; Pering et al. 2000], in which the frequency and the voltage levels for the processor core, are determined by the operating system. A power efficient compiler determines the voltage level and clock frequency at compilation time from high level code in [Hsu et al. 2000]. Similar to the approach in [Burd et al. 2000], the authors in [Pouwelse et al. 2001a], describe a system for low-power microprocessor using dynamic voltage scaling. An energy prioritized scheduler mediates between the application software and the operating system in determining the voltage and frequency levels for the CPU, is described in [Pouwelse et al. 2001b]. In [Grunwald et al. 2000], voltage and clock scheduling algorithms are incorporated within the operating system. In [Martin and Siewiorek 2001], the authors describe the system-level power-performance trade-off for a variable frequency processor system.

In the above works, the suitable frequency and voltage at which the CPU core should be run, is determined either at the operating system level or at the compiler level. Thus, it is quite evident that simultaneous voltage and frequency scaling is becoming important in low power processors. In the above works, frequency scaling is explored at the CPU and system levels, while we explore the use of dynamic frequency clocking within the datapath and datapath scheduling algorithms that can be incorporated into a datapath synthesis tool.

Several low power datapath scheduling techniques have been developed and reported in the literature. A scheduling algorithm using the "shut-down" technique, multiplexor reordering and pipelining is described in [Monteiro et al. 1996]. Scheduling and resource binding algorithms are described in [Musoll and Cortadella 1995], which reduce the power by reducing the activity of functional units by minimizing transition of operands. In [Lin et al. 1997], an ILP formulation and a heuristic for variable voltage scheduling is presented. An ILP-based datapath scheduling scheme using both multiple supply voltages and dynamic frequency clocking is describded in [Mohanty et al. 2003]. A scheduling algorithm called MOVER is presented in [Johnson and Roy 1997a] using ILP formulation. A dynamic programming technique for multiple supply voltage scheduling is discussed in [Chang and Pedram 1997]. A time constrained multiple voltage scheduling technique is proposed in [Sarrafzadeh and Raje 1999]. A resource constrained scheduling algorithm with multiple supply voltages is given in [Kumar and Bayoumi 1999] which helps in reducing power using multiple supply voltages. In [Shiue and Chakrabarti 2000], a resource and a latency constrained list-based scheduling algorithms with multiple supply voltages are discussed. Resource and time constrained scheduling based on the Lagrange multiplier method is addressed in [Manzak and Chakrabarti 2002].

The above scheduling techniques are based on a single clock frequency and consider multiple supply voltages, voltage scaling, capacitance reduction, and switching activity reduction. In this work, we consider the use of dynamic frequency clocking or frequency

scaling alongwith multiple supply voltages in developing resource and time constrained low power datapath synthesis scheduling algorithms.

3. DYNAMIC FREQUENCY CLOCKING AND ENERGY SAVINGS

In this section, we discuss the concept of dynamic clocking frequency in brief. We also analyse the role of dynamic frequency clocking alongwith multiple supply voltages in reducing energy consumption while maintaining performance using a small example.

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. The design and use of such clocking mechanism is explored in several works [Kim and Chae 1996; Ranganathan et al. 1998; Brynjolfson and Zilic 2000b; 2000a; Benini et al. 1998; Benini et al. 1999]. The dynamic frequency clocking mechanism has been shown to improve the execution time as compared to using a uni-frequency global clock in [Ranganathan et al. 1998]. In [Benini et al. 1998; Benini et al. 1999], a concept similar to dynamic clocking, called variable-latency telescopic unit is used to synthesise high-performance systems. Fig. 1 shows the unifrequency and dynamic frequency diagrams. The dynamic clocking unit (DCU) generates the required clock frequency utilizing a clock divider strategy to generate frequencies which are submultiples of the base frequency. The base frequency f_{base} is the maximum frequency (or multiple of maximum) of any functional unit (FU) at the maximum supply voltage. A value cfi_c is loaded as an input to the DCU which comes from the controller. The scheme for dynamic frequency generation is shown in Fig. 2. The clock is determined by dividing the base frequency by cfi_c value, $\frac{f_{base}}{cfi_c}$.

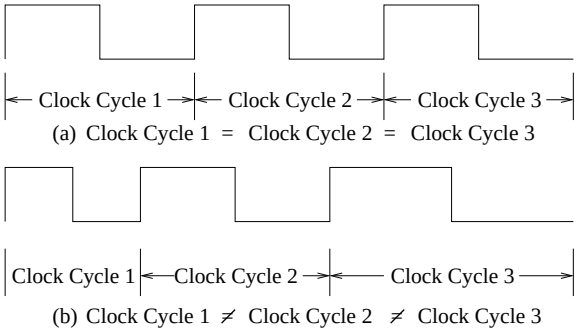


Fig. 1. (a) Single frequency (b) Dynamic frequency

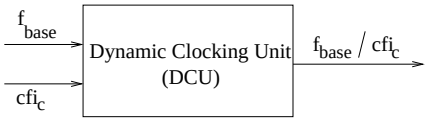


Fig. 2. Scheme for dynamic frequency generation

As discussed in section 1, with reference to the equations 1-3, frequency scaling helps in reducing power, but not energy [Pering et al. 2000; Burd and Brodersen 1995]. The

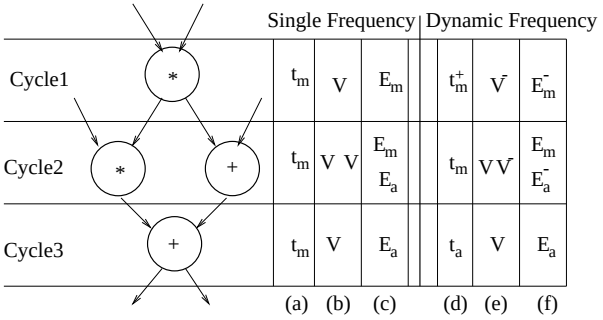


Fig. 3. Example DFG

frequency reduction creates an opportunity to operate the different functional units at different voltages, which in turn helps in energy reduction. With the help of an example, we illustrate how dynamic frequency clocking or frequency scaling can be helpful in energy reduction while maintaining performance. Let us consider the example data flow graph (DFG) shown in Fig. 3. Let t_a and t_m be the delays of the adder and the multiplier respectively at the maximum supply voltage V. The DFG has a schedule of three control steps. Let us consider three possible modes of operation, such as (i) single supply voltage and single frequency, (ii) multiple supply voltage and single frequency, and (iii) multiple supply voltages and dynamic frequency. It may be noted that in case (ii), the energy overhead of the level converters has to be taken into account. Similarly, energy overhead of level converters and that of DCU need to be considered in case (iii).

(i) Single supply voltage and single frequency : Each cycle has a clock width dictated by the slowest operator delay t_m. The total energy consumption is given by $E_S = 2E_m + 2E_a$ and the total delay is $T_S = 3t_m$.
(ii) Multiple supply voltage and single frequency : Let, E_m^- and E_a^- be energy consumption values for multiplier and adder, respectively, when operating at a lower voltage V^-. At this supply voltage, let t_m^+ and t_a^+ be the delay of multiplier and adder, respectively. It is evident that E_m^- and E_a^- are smaller than E_m and E_a, and t_m^+ and t_a^+ are larger than t_m and t_a, respectively. In this case, the clock frequency will be driven by the larger delay, the delay of multiplier. Thus, the energy consumption of the DFG is given by $E_M = E_m + E_a + E_m^- + E_a^-$, and the total delay is, $t_M = 3t_m^+$. Since, $E_M < E_S$, and $T_M > T_S$, in other words the energy savings comes at the cost of time penalty.
(iii) Multiple supply voltages and dynamic frequency : In this case, the total energy consumption of the DFG is given by $E_D = E_m + E_a + E_m^- + E_a^-$ (same as the E_M in the case (ii)). Moreover, in this case we have variable clock. So, the delay for cycle 1, cycle 2, and cycle 3 are t_m^+, $\max(t_m, t_a^+)$, and t_a, respectively. Thus, the total delay is $T_D = t_m^+ + \max(t_m, t_a^+) + t_a$. It is obvious that $t_a < t_m^+$, and $\max(t_m, t_a^+) < t_m^+$, so $T_D < T_M$. Since, $E_D = E_M$ and $T_D < T_M$, we conclude that the time penalty has reduced compared to case (ii) for same energy reduction. Now, let us compare with case (i). Since, the delay of an adder is much less than delay of a multiplier, without loss of generality we can assume that $\max(t_m, t_a^+) \approx t_m$, and also that $t_m^+ + t_a \approx 2t_m$ (as $t_m^+ > t_m$ and $t_a < t_m$); it is possible that $T_D \approx 3t_m = T_S$. Thus, we have $E_D < E_S$ and $T_D \approx T_S$, in other words, energy reduction is achieved without degrading performance.

4. TARGET ARCHITECTURE AND DATAPATH SPECIFICATIONS

The target architecture model assumed for the scheduling schemes is shown in Fig. 4. Each functional unit feeds one register and has a multiplexor also. The register and the multiplexor operate at the same voltage level as that of the functional units. Level converters are used when a low-voltage functional unit is driving a high-voltage functional unit [Johnson and Roy 1997a; Shiue and Chakrabarti 2000]. A controller decides which functional units are active in each control step and the inactive ones are disabled using the multiplexors. The controller has a storage unit to store the parameters cfi_c obtained from the scheduling. The cycle frequency f_c is generated dynamically using DCU and a functional unit operating at one of the supply voltages is activated. It may be noted that while the level converters are many and internal to the datapath circuit, the DCU is only one and external to it, and both are considered as overheads for multiple voltage and dynamic clocking based design.

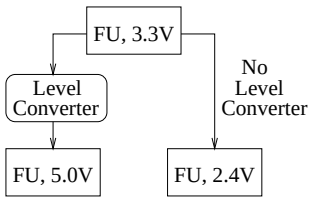


Fig. 4. Level converters needed for stepping up signal

The datapath is specified as a sequencing data flow graph (DFG) [Micheli 1994]. Each vertex of the DFG represents an operation and each edge represents dependency. In this work, we are considering the signal processing applications, in which the dynamic frequency clocking scheme is useful for energy reduction. So, we assume that the datapath circuit is represented as directed acyclic DFG. The DFG does not support the hierarchical entities and the conditional statements are handled using comparison operation. Each vertex has attributes that specifies the operation type.

The delay of a control step is dependent on the delays of the functional unit and the multiplexer and register pairs. Let, d_{reg} be the delay of the register, d_{mux} be the delay of the multiplexor, d_{fu} be the delay of the functional unit and d_{level} be the delay of the level converter. The worst case operational delay of a functional unit can be written as :

$$d_{FU} = d_{reg} + d_{mux} + d_{fu} + d_{level} \tag{4}$$

The register delays include the set-up and propagation delays. The delay of control step d_c is the delay of the slowest functional unit in the control step c. Using the above delay model, the worst case delays of the library components are estimated. For a given base frequency (f_{base}), maximum frequencies of each FU is scaled down to operating frequencies given by ($\frac{f_{base}}{cfi_c}$), where, $cfi_c = 1, 2, ...,$ any positive integer. In general, the value of cfi_c is bounded by total number of resources raised to the power number of frequency levels. Assuming two resources, such as multiplier ($MULT$) and arithmatic logic unit (ALU), for three frequency levels, the possible frequencies are, $ALU_{High}(cfi_c = 1)$, $ALU_{Med}(cfi_c = 2)$, $ALU_{Low}(cfi_c = 4)$, $MULT_{High}(cfi_c = 2)$, $MULT_{Med}(cfi_c = 4)$ and $MULT_{Low}$ $(cfi_c = 8)$. For example, if the base frequency fed to the DCU is $36MHz$, then the frequencies generated are, $18MHz$, $9MHz$ and $4.5MHz$. The clock

frequency for a given control step is the minimum of the operating frequencies of all FUs active in that step.

5. TIME CONSTRAINED SCHEDULING

The objective is to the minimize the energy consumption without violating the timing constraint while the resources are operating at different supply voltages and are available in unlimited numbers. The inputs to the algorithm are an unscheduled DFG, the scaled down operating frequencies, and the execution time constraint T_c for the whole schedule. The output produced by the algorithm are scheduled DFG, voltage assignment for each node, cycle frequency indices, energy estimates. To get more energy savings and at the same time maintain performance, the multipliers are to be operated at as low frequencies as possible and the adders at as high frequencies as possible. This objective can be achieved if adders or subtractors are not operated alongwith multipliers in the same duty cycle. In the cases, when they are to be operated during the same cycle to meet the time constraint, energy savings will come from the multipliers only. Initially, TC-DFC generates a schedule such that the low frequency operators are scheduled at earlier steps and the high frequency operators are scheduled at later steps. Later on, the TC-DFC 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 assigns appropriate voltage.

Step 1	: Find an ASAP schedule for the sequencing UDFG.
Step 2	: Create a priority list of vertices using the ASAP schedule in Step 1.
Step 3	: Assign control steps to the operations such that the higher priority vertices are scheduled at earlier time stamps, precedence is satisfied and also the multiplications and ALU operations are not scheduled in the same cycle.
Step 4	: Find the cycles having only ALU operations, those with only multiplications and those with both ALU operations and multiplications (mixed) for the schedule obtained.
Step 5	: Create a priority list of clock cycles such that cycles with only ALU operations get higher priority than the cycles with only multiplications or those with mixed operations (cycles with only multiplications get higher priority than the cycles with mixed operations).
Step 6	: Initialise the operating frequency of each cycle.
Step 7	: If time constraint is not satisfied, the highest priority cycle is assigned the next higher frequency and repeat the step for the next higher priority cycle if necessary.
Step 8	: If all cycles having multipliers operating at the highest frequency, then eliminate the cycle having minimum number of ALU operations, adjust the schedule and go to Step 4.
Step 9	: Do voltage assignment and determine energy details.
Step 10	: Find the cycle frequency index for each cycle.

Fig. 5. TC-DFC Scheduling Algorithm Flow

5.1 TC-DFC Algorithm

TC-DFC scheduling algorithm is presented in brief in Figure 5. In step 1, an as soon as possible (ASAP) schedule for the given input unscheduled data flow graph is determined. In step 2, the scheduler creates a priority list of the vertices in which higher priority is given to the vertices which are to be scheduled at earlier control steps. This priority approach ensures scheduling of energy hungry resources at earlier control steps, other resources at later control steps and avoids their concurrent operations. The priority list is created as follows, all multiplications (i.e low frequency operations) are grouped with higher priority

than the ALU operations (i.e. high frequency operations, such as additions, subtractions, etc.). Among the multiplication operations, higher priority is given to the operations with smaller ASAP time stamp. Similarly, among the ALU operations higher priority is assigned to the operations with small ASAP time stamp. In step 3, the vertices are time stamped in ASAP manner using the vertex priority list such that no multiplication and ALU operations scheduled to function concurrently. Moreover, it is ensured that operation precedence is satisfied and higher priority vertices are scheduled at earlier time stamps.

Table I. TC-DFC Frequency Selection : From Left $\rightarrow$ Right in each Step

	$MULT_{Low}$	$MULT_{Med}/ALU_{Low}$	$MULT_{High}/ALU_{Med}$	ALU_{High}
Frequency	$4.5MHz$	$9MHz$	$18MHz$	$36MHz$
cfi_c	8	4	2	1

In step 4, for the current schedule, the cycles are categorised as, cycles having only ALU operations, only multiplication, and both ALU operations and multiplication (mixed operations). It may be noted that the aim of the scheduling is to avoid concurrent scheduling of ALU operations (use low energy resources) and multiplication operations (use high energy resources) as much as possible, but it may not be possible when the time constraint is very strict. In step 5, a priority list of clock cycles is created such that the higher priority cycles are preferred candidates for higher frequency assignment. The cycle priority list is created as follows. The cycles with only ALU operations get higher priority than the cycles with only multiplications and the cycles with mixed operations. The cycles with only multiplications get higher priority than the cycles with mixed operations. Further, among the cycles with only ALU operations higher priority is given to the cycle having lesser number of ALU operations. Similarly, among the cycles with only multiplication operations higher priority is given to the cycle having lesser number of multiplication operations. However, among the cycles with mixed operations higher priority is given to cycles having lesser number of multiplications. In step 6, initial cycle frequency is assigned as left most operating frequency from the Table I. It may be noted that the Table I shows two types of resources, such as multilpiers and ALU, and three frequency levels for each, however, it can be extended to accomodate more than two types of resources and frequency levels in similar manner. In step 7, in order to fulfil time constraint, the frequency of highest priority cycle is increased using Table I. If needed the process is repeated for the next higher priority cycle. The repetition is necessary, if it is found that the maximum frequency assignment for highest priority cycle did not satisfy the time constraint. In step 8, if it is found that all cycles with multiplication (low priority cycles) are operating at highest frequency to satisfy the time constraints then the cycle having minimum number of ALU operations is eliminated and the schedule is adjusted. This is necessary due to the fact that if the multipliers (energy hungry resources) operate at highest frequency, then there will not be any energy reduction. The adjustment involves reducing the time stamp of the vertices scheduled in the cycle to be eliminated and the time stamp of its successors and predecessors. In step 9, voltage assignment is done and the energy estimate for entire DFG is found out. At this step, the minimum allowable voltage that meets the cycle frequency is assigned. In step 10, the cycle frequency index for each cycle is based on Table I.

A detailed representation of the above algorithm in the form of pseudo-code is given in figure 6. The list of functions needed in implementation of the algorithm is given in

```
TC-DFCAlgorithm(UDFG, T_c, Operating Frequency)
{
(01) ASAPScheduler (UDFG); CreateVertexPriorityList (ASAPSchedule); cycle = 1;
(02) TC-DFCSchedSteps_{v_0} = 0; ScheduledVertexList = v_0; // source vertex scheduled
(03) while(VertexPriorityList ≠ NULL) {
(04)     v_i = TOP (VertexPriorityList);
(05)     if(v_i ∉ ScheduledVertexList and AllPredecessor_{v_i} ∈ ScheduledVertexList) then {
(06)         if( CheckFrequencyConstraint (cycle)) then cycle = Max (TC-DFCSchedSteps) + 1;
(07)         else schedule in current cycle;
(08)         TC-DFCSchedSteps_{v_i} = cycle; VertexPriorityList = VertexPriorityList − v_i;
(09)         ScheduledVertexList = ScheduledVertexList ∪ v_i; } // end if (05) } // end while (03)
(10) TC-DFCNoOfSteps = Max (TC-DFCSchedSteps);
(11) CreateCyclePriorityList (CurrentSchedule, TC-DFCNoOfSteps);
(12) CycleFrequencyList = InitializeFrequency (Table I);
(13) T_s = CalculateDelay (CycleFrequencyList);
(14) while (T_s > T_c) {
(15)     c_i = TOP (CyclePriorityList);
(16)     while (T_s > T_c) {
(17)         if ( CycleFrequencyList_{c i} < Highest operating frequency depending on v_i ∈ c_i ) then {
(18)             CycleFrequencyList_{c_i} = FindNextFrequency (Table I);
(19)             T_s = CalculateDelay (CycleFrequencyList); } //end if (17)
(20)         else c_i = TOP (CyclePriorityList); } // end while (16)
(21)     if (All cycles having multipliers are operating at highest frequency) then {
(22)         c_i = FindCycleWithMinimumALU (for all cycle c_i);
(23)         for each v_i ∈ c_i do reduce time stamp of v_i and adjust Predecessor_{v_i} and Successor_{v_i}
(24)         TC-DFCNoOfSteps = Max (TC-DFCSchedSteps);
(25)         CreateCyclePriorityList (CurrentSchedule, TC-DFCNoOfSteps);
(26)         CycleFrequencyList = InitializeFrequency (Table I);
(27)         T_s = CalculateDelay (CycleFrequencyList);
(28)     } // end if (21) } // end while (14)
(29) if (All multipliers are operating at highest frequency) then
(30)     print ("Energy efficient schedule is not possible for this time constraint.");
(31) Do voltage assignment, find cycle frequency index, and Estimate energy consumption;
} // End Algorithm TC-DFC
```

Fig. 6. Pseudo-code for TC-DFC Scheduling Algorithm

Table II. Similarly, the data structures or the identifiers used in the algorithm description are summerized in Table III.

We now explain the working of the above algorithm using HAL benchmark DFG. We start with an unscheduled HAL DFG from [Micheli 1994]. Using step 1, the ASAP time stamps are assigned and the ASAP scheduled DFG shown in figure 7 is obtained. The vertex priority list for this DFG is given in Table IV as created in step 2. Then in step 3, using this vertex priority list another schedule is obtained, which is the DFG in figure 8(a) without voltage or cfi_c assignment, in which no multipliers or ALUs operations scheduled concurrently. For this DFG the cycle priority list shown in Table V obtained using step 4. Using this cycle priority list, frequency assignments are done in the step 6 and step 7 to meet the time constraints, say, $T_c \approx 2 * T_{cp}$. Finally, step 9 and step 10 do the voltage assignment and cycle frequency index calculation, respectively, and the final DFG shown in figure 8(a) is obtained. Similarly, the time constraint, $T_c \approx 1.75 * T_{cp}$ could be met using the same cycle priority list (Table V) with higher frequency assigned to next priority

Table II. List of Functions used in the TC-DFC Algorithm

Functions	Description	Complexity
ASAPScheduler	: Determines the ASAP time stamp of the vertices.	$\Theta(\|V\| + \|E\|)$
CreateVertexPriorityList	: Creates a priority list of vertices such that the vertex with lower operating frequency gets the higher priority.	$\Theta(\|V\|)$
TOP	: Finds the first vertex from the priority list array	$\Theta(1)$
CheckFrequencyConstraint	: Checks the frequency constraint in a control step.	$\Theta(1)$
Max	: Finds the maximum value from an array.	$\Theta(c)$
CreateCyclePriorityList	: Constructs the cycle priority list in an array.	$\Theta(c)$
InitializeFrequency	: Initialize operating frequency of each cycle	$\Theta\left(L_f\right)$
CalculateDelay	: Calculate critical path delay using *CycleFrequencyList*	$\Theta(c)$
FindNextFrequency	: Find the next available frequency.	$O\left(L_f\right)$
FindCycleWithMinimumALU	: Finds the control step with minimum number of ALU operations.	$\Theta(cR_T)$
Adjust Predecessor	: Adjusts time stamp of predecessor	$O(\|V\|)$
Adjust Successor	: Adjusts time stamp of successor	$O(\|V\|)$
Voltage Assignment	: Assigns voltage to each vertex.	$\Theta(\|V\|)$
Find Cycle Frequency Index	: Finds cycles frequency indices of all cycles.	$\Theta(c)$

Table III. List of Variables and Data Structures used in the TC-DFC Algorithm Description

Data Structures	Descriptions
ASAPSchedule	: An array used to store ASAP time stamp of each vertex.
TC-DFCSchedStep	: An array used to store TC-DFC time stamp of each vertex.
ScheduledVertexList	: An array used to store vertices already scheduled.
VertexPriorityList	: An array used to store vertices in a priority order.
CyclePriorityList	: An array used to store control steps in a priority order.
TC-DFCNoOfSteps	: Total number of control steps of TC-DFC schedule.
CycleFrequencyList	: An array used to store frequency of each cycle.
cycle	: A Temporary variable.

cycle. Say, now we want a schedule with time constraint $T_c \approx 1.5 * T_{cp}$. Using the cycle priority list (Table V), step 7 attempts assigning frequency. It is found that the cycle with multipliers scheduled are in highest operating frequency to meet such constraint. So, using step 8, cycle 5 is eliminated from DFG in figure 8(a) without voltage or cfi_c assignment that was obtained in step 3 before. The new DFG is the DFG shown in figure 8(c) without voltage or cfi_c assignment. For this DFG, we obtain the cycle priority list shown in Table VI. Using the step 6,7,9,10 as above, we obtain the final scheduled DFG in figure 8(c).

Table IV. Vertex Priority List for HAL DFG (Step 2)

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

5.2 TC-DFC Time Complexity

Let there be $|V|$ number of vertices and $|E|$ number of edges in the DFG. Suppose the number of control steps found out from the ASAP scheduling is c. Let L_f denote the number of frequency levels and R_T denote the number of resource types. Based on the time

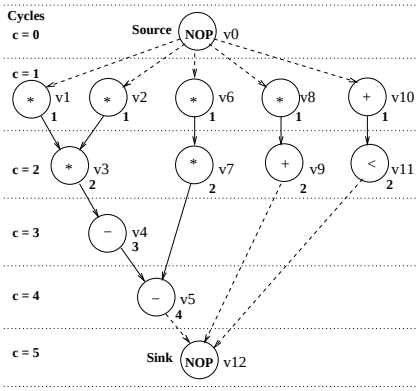


Fig. 7. HAL Differential Equation Solver Benchmark DFG with ASAP labels (Step 1)

Table V. Cycle Priority List for HAL DFG :$T_c \approx 2 * T_{cp}$ or $1.75 * T_{cp}$ (Step 4)

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

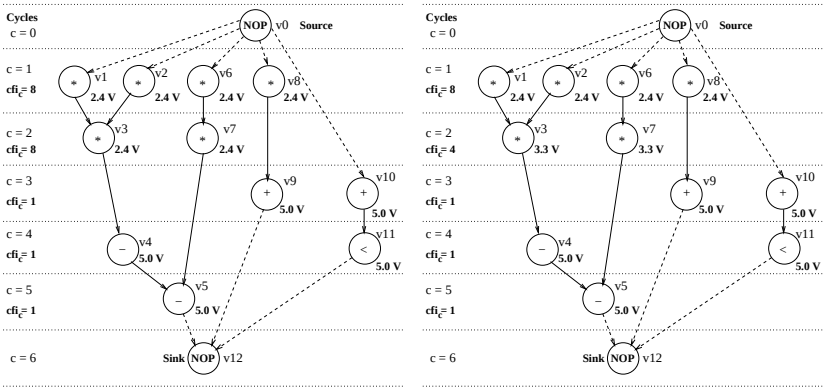
Table VI. Cycle Priority List for HAL DFG $T_c \approx 1.5 * T_{cp}$ (Step 4 after Step 8)

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

complexity of the different functions given in Table II, we provide the following analysis for the worst-case running time of the TC-DFC algorithm. Time taken by the instruction from line 01-02 is $\Theta(|V| + |E|) + \Theta(|V|)$. The running time of the code-segment line 03-09 is $\Theta(c|V|)$. Similarly, $\Theta(c) + \Theta(L_f)$ is the running time of the code segment line 10-13. The while loops in line 14 and 16 terminate when the time constraint is satisfied, which involves a search in the frequency selection table. So, the number of times these while loops are executed is independent of the input size $|V|$ or $|E|$. Thus, the time complexity of the code segment in line 14-31 is $\Theta(cR_T) + \Theta(|V|) + \Theta(L_f) + \Theta(c) + \Theta(c) + \Theta(L_f)$, which is same as writing (from algorithm complexity point of view) $\Theta(cR_T) + \Theta(|V|) + \Theta(L_f) + \Theta(c)$. Without loss of generality, we can assume that the R_T, L_f and c are upper bounded by the number of vertices $|V|$. Using this assumption the overall running time of the algorithm is expressed as : $\Theta(|V| + |E|) + \Theta(|V|\,|V|)$. For strongly data-dependency, we have $|E| \approx |V|^2$ and for weak data-dependency $|E| << |V|^2$. In either case, *the simplified time-complexity of the TC-DFC scheduling algorithm is $|V|^2$; in other words, the time-complexity is polynomial to the number of vertices in the data flow graph.*

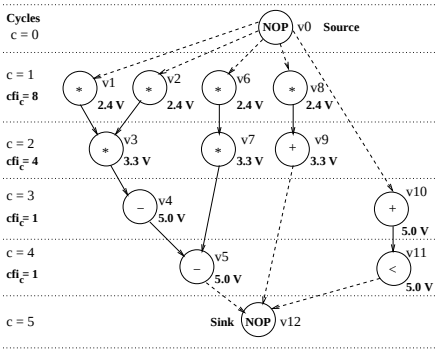
6. RESOURCE CONSTRAINED SCHEDULING

In resource constrained algorithm, the objective is to minimize energy consumption by maximizing the utilization of low supply voltage resources from given set of resources operating at different supply voltages while reducing the time penalty as much as possible. The combined reduction of energy consumption and time penalty translates to reduction



(a) Time Constrained : $T_c \approx 2.0 * T_{cp}$ (Step 6,7,9,10)

(b) Time Constrained : $T_c \approx 1.75 * T_{cp}$ (Step 6,7,9,10)



(c) Time Constrained : $T_c \approx 1.5 * T_{cp}$ (Step 8,6,7,9,10)

Fig. 8. Schedules Obtained for HAL Benchmark for Different Time Constraints using TC-DFC

of the energy-delay-product. Thus, the objective of RC-DFC is to minimize the energy-delay-product while assigning a schedule for the DFG. For a resource i operating in clock step c, let, (i) $\alpha_{i,c}$ be the switching, (ii) $C_{i,c}$ be the load capacitance and (iii) $V_{i,c}$ be the operating voltage. If a level converter is needed, it is considered as a resource needed in the particular clock cycle in which it needs to step up the signal. If N is the total number of clock cycles for the DFG, NR_c is the number of resources active in cycle c, and f_c is the cycle frequency, then, the total energy consumption of the DFG E_M and the energy-delay-product EDP_D are characterised by equation 5. The inputs to the algorithm are an unscheduled DFG, the resource constraints which include the number of resources, their corresponding operating voltages and the scaled down operating frequencies. The algorithms generates various outputs, such as scheduled DFG with node voltages assigned, cycle frequency indices, energy, delay estimates, and energy-delay-product estimates.

$$\begin{aligned} E_D &= \sum_{c=1}^{N} \sum_{i=1}^{NR_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 \\ EDP_D &= E_M * T_D = \left(\sum_{c=1}^{N} \sum_{i=1}^{NR_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 \right) * \sum_{c=1}^{N} \frac{1}{f_c} \end{aligned} \quad (5)$$

Table VII. Frequency Selection : From Left to Right in each Step

FUs in a Cycle		Frequency Priority Order
MULT	-	$MULT_{Low}, MULT_{Med}, MULT_{High}$
MULT	ALU	$MULT_{Low}, ALU_{Low}, MULT_{High}$
-	ALU	$ALU_{High}, ALU_{Med}, ALU_{Low}$

Table VIII. Resource Look-up Table (order, from Left to Right)

Clock	MULT			ALU		
Cycle	2.4 V	3.3 V	5.0 V	5.0 V	3.3 V	2.4 V
c	1	2	1	1	1	0

RC-DFC attempts to operate the multipliers at as low frequency as possible, the resulting decrease in performance is compensated by operating the ALUs at as high frequency as possible. Depending on which functional units are active in a given cycle, the algorithm determines the frequency using a lookup table (LUT), called "frequency selection LUT", such as the one shown in Table VII scanning it left to right. In a schedule, if only multipliers are needed in a particular cycle the frequency selection is in the order $MULT_{Low}$, $MULT_{Med}$, $MULT_{High}$. If both multipliers and the ALUs are all operating in a given clock cycle, the frequency selection is in the order $MULT_{Low}$, ALU_{Low}, $MULT_{High}$. If only ALUs are operating in a control step, then the frequency selection is in the order ALU_{High}, ALU_{Med}, ALU_{Low}. Another lookup table called "resource assignment LUT" is constructed considering the resource constraints and the table is used to match the selected frequency with a corresponding voltage level. The resources are assigned scanning the LUT, from left to right. Moreover, the scheduling algorithm uses heuristics to minimize the number of times level the conversions are needed. An example of resource assignment LUT, is shown in Table VIII with resource constraints: one MULT at $2.4V$, two MULT at $3.3V$, one MULT at $5.0V$, one ALU at $3.3V$ and one ALU at $5.0V$. The arrangement of the MULTs is in the order from low to high voltage, whereas for the ALUs it is from high to low. The LUT is updated during each assignment to make sure that the resource-constraints are not violated. The dimension of this LUTs Table VII and Table VIII depend on the total number of clock cycles of the schedule and/or the number of resource types. The tables can be extended to accomodate more types of resources, and voltage and frequency levels. It has to be ensured that the arrangement of the energy hungry resources is in the order from low to high voltage, whereas for the lesser energy consuming resources is from high to low.

Step 1	: Derive ASAP and ALAP schedules for the unscheduled DFG.
Step 2	: Determine the number of resources (multipliers and ALUs) at different operating voltages.
Step 3	: Modify the schedules obtained in Step 1 using number of resources determined in Step 2.
Step 4	: Calculate the total number of control steps which is the larger those of ASAP and ALAP schedules from Step 3.
Step 5	: Construct the "resource assignment LUT" and "frequency selection LUT".
Step 6	: Find the vertices having non-zero mobility and vertices with zero mobility and assume ASAP schedule in Step 3 as the current schedule.
Step 7	: Do voltage and frequency assignment using the current schedule and the LUTs.
Step 8	: Taking a vertex with non-zero mobility time stamp it using LUTs such that the energy-delay-product of the whole DFG is minimum.
Step 9	: Adjust current schedule, predecessor and successor time stamps, LUTs, and repeat Steps 7 and 8 to time stamp each of the remaining vertices with non-zero mobility.
Step 10	: Determine the clock frequency index for each cycle.

Fig. 9. RC-DFC Scheduling Algorithm Flow

6.1 RC-DFC Algorithm

Figure 9 shows the flow of the proposed algorithm. In step 1, the scheduler determines the ASAP and the ALAP schedules for the UDFG. For example, if the resource constraint is 2 ALUs at $2.4V$, 1 ALU at $3.3V$, 1 multiplier at $2.4V$, and 3 multipliers at $5.0V$, then the number of ALUs is 3 and the number of multipliers is 4. In this step, the time constraint for ALAP schedule is obtained from ASAP schedule. In step 2, the total number of resources is found out as the sum of each resource at different voltage levels. In step 3, the ASAP and ALAP schedules of step 1 are modified using the number of resources found in step 2 so that the resource constraints are not violated. In this process, the mobility of the vertices are restricted to great extent and the search space for the following steps reduces. In step 4, the total number of control steps for both ASAP and ALAP schedule are found out and the number of control steps for the final steps is assumed to be the maximum of the two. This step is necessary due to the fact that the total no of clock cycles may be different from that of original ASAP or ALAP schedule, and also they may not remain same for the both in the process of satisfying the stringent resource constraints. In step 5, the "resource assignment LUT" and "frequency selection LUT" are constructed. The resource assignment LUT is constructed (similar to Table VIII) whose size depends on number of control steps, number of resource types, and number of voltage levels. In step 6, the vertices having non-zero mobility (with different ASAP and ALAP time stamps) and the vertices with zero mobility (with same ASAP and ALAP time stamps) are found out and the current schedule is initialized as the ASAP schedule obtained in step 3. In step 7, voltage and frequency assignments are made for the current schedule using the LUTs. This steps returns two lists: one containing the assigned voltage of each vertex and the other containing the selected frequency for each cycle. In step 8, the scheduler finds a proper step for each vertex having non-zero mobility such that the energy-delay-product of the whole DFG is minimum. In step 9, current schedule and resource assignment LUTs are adjusted to satisfy the precedence. In step 10, cycle frequency indices are found for all cycles which would be stored in the controller and would be fed to the DCU for clock generation. The algorithm terminates once all non-zero mobility vertices are scheduled.

The pseudo-code for the algorithm is shown in figure 10. The list of functions needed in implementation of the algorithm is given in Table IX. Similarly, the data structures or the

```
RC-DFCAlgorithm(UDFG, FUs, Voltage Levels, Operating Frequencies)
{
(01)ASAPScheduler(UDFG); ALAPScheduler(UDFG);
(02)MULT = ∑ Multipliers of different voltage levels; ALU = ∑ ALUs of different voltage levels;
(03)ModifySchedule(ASAPSchedule, MULT, ALU); ModifySchedule(ALAPSchedule, MULT, ALU);
(04)NoOfControlSteps = Max(ASAPControlSteps, ALAPControlSteps);
(05)ConstructResAssignmentLUT(NoOfControlSteps, FUs);
(06)FindResTypeForEachVertex(UDFG); ConstructFreqSelectionLUT(Operating Frequency);
(07)FindMobileVertexList(ASAPSchedule, ALAPSchedule); CurrentSchedule = ASAPSchedule;
(08)while(NonZeroMobilityVertexList is NOT empty) {
(09)    max = −∞; AllocateVoltAndFreq(CurrentSchedule, LUTs);
(10)    CurrentEDP = CalculateEDP (VoltageArray,FrequencyArray);
(11)    for each v_i ∈ NonZeroMobilityVertexList {
(12)        start = CurrentSchedule[v_i]; end = ALAPSchedule[v_i];
(13)        for cycle = start → end in steps of 1 {
(14)            TempSchedule = AdjustSchedule(CurrentSchedule, v_i, cycle);
(15)            AllocateVoltAndFreq(TempSchedule, LUTs);
(16)            TempEDP = CalculateEDP(VoltageArray,FrequencyArray);
(16)            ExtraEDP = CurrentEDP − TempEDP;
(17)            if(ExtraEDP > max) {
(18)                max = ExtraEDP; CurrentVertex = v_i; CurrentCycle = cycle; } // end if (17)
(19)        } // end for (13) } // end for (11)
(20)    CurrentSchedule = AdjustSchedule(CurrentSchedule, CurrentVertex, Currentcycle);
(21)    Update the "resource assignment LUT";
(22)    ZeroMobilityVertexList = ZeroMobilityVertexList ∪ CurrentVertex;
(23)    NonZeroMobilityVertexList = NonZeroMobilityVertexList − CurrentVertex; }//end while(08)
(24)AllocateVoltAndFreq(CurrentSchedule, LUTs);
(25)EnergyAndDelayDetails(VoltageArray, FrequencyArray); FindCycleFreqIndex(FrequencyArray);
} // End Algorithm RC-DFC
```

Fig. 10. Pseudo-code for RC-DFC scheduler

Table IX. List of Functions used in the RC-DFC Algorithm

Functions	Description	Complexity
ASAPScheduler	: Determines the ASAP time stamp of the vertices.	$\Theta(\|V\| + \|E\|)$
ALAPScheduler	: Determines the ALAP time stamp of the vertices.	$\Theta(\|V\| + \|E\|)$
ModifySchedule	: Modifies the unconstrained schedules to incorporate voltage relaxed resource constraints.	$\Theta(\|V\| + \|E\|)$
ConstructResAssignmentTable	: Constructs resource assignment LUT.	$\Theta(cL_v R_T)$
Max	: To find maximum of control steps.	$\Theta(1)$
FindResTypeForEachVertex	: Identifies the FU needed for the operation at each vertex.	$\Theta(\|V\|)$
ConstructFreqSelectionLUT	: Constructs frequency selection LUT.	$\Theta\left(L_f\right)$
FindMobileVertexList	: Find the mobility of each vertex.	$\Theta(\|V\|)$
AllocateVoltAndFreq	: Allocates the voltage level and frequency levels using LUTs and current schedule.	$\Theta(c\,\|V\|\,L_v R_T)$
CalculateEDP	: Calculate energy delay product of the whole DFG	$\Theta(\|V\|)$
AdjustSchedule	: Adjust predecessor and successor time stamps such that precedence is satisfied for a particular vertex	$O(\|V\|)$
Update Resource Assign. LUT	: Constructs resource assignment LUT.	$\Theta(1)$
FindEnergyAndDelay	: Determines energy consumption and delay.	$\Theta(\|V\|)$
FindCycleFreqIndex	: Finds cycles frequency indices of all cycles.	$\Theta(c)$

Table X. List of Variables and Data Structures used in the RC-DFC Algorithm Description

Data Structures	Descriptions
ASAPSchedule	: An array used to store ASAP time stamp of each vertex.
ALAPSchedule	: An array used to store ALAP time stamp of each vertex.
CurrentSchedule	: An array used to store current schedule time stamp.
TempSchedule	: An array used to store temporary schedule time stamp.
MULT	: Number of multipliers at all voltage levels.
ALU	: Number of ALUs at all voltage levels.
ASAPControlSteps	: Total number of control steps of ASAP schedule.
ALAPControlSteps	: Total number of control steps of ALAP schedule.
NoOfControlSteps	: Number of control steps of the schedule.
ResAssignmentLUT	: Resource assignment look-up table.
FreqSelectionLUT	: Frequency selection look-up table.
max, start, end, cycle	: Temporary variables.
CurrentEDP, TempEDP, ExtraEDP	: Temporary variables.
CurrentVertex, CurrentCycle	: Temporary variables.
VoltageArray	: An array used to store operating voltage for each vertex.
FrequencyArray	: An array used to store operating frequency for each cycle
ZeroMobilityVertexList	: An array storing the vertices with zero mobility.
NonZeroMobilityVertexList	: An array storing the vertices with non-zero mobility.

identifiers used in the algorithm description are summerized in Table X. It may be noted that the algorithm can easily be extended to handle more than two types of resources, in such a scenario the dimension of "resource assignment LUT" and "frequency selection LUT" are going to change. Moreover, the multiplier will be replaced with the highest energy consuming resource, the ALU will be replaced with the lowest energy consuming and others will be fall in between. A final schedued DFG obtained using this algorithm is shown in figure 11 for the resource constraint (one MULT at $2.4V$, one MULT $3.3V$, one ALU at $3.3V$ and one ALU at $5.0V$).

6.2 RC-DFC Time Complexity

Let there be $|V|$ number of vertices and $|E|$ number edges in the DFG, out of which $|V_m|$ number of vertices have mobility and the maximum mobility of any mobile vertex is t_m. Let L_v denote the number of voltage levels and L_f denote the number of frequency levels. Suppose the number of control steps found out from the ASAP scheduling is c. Assumming that $L_V$and L_f are upper bounded by $|V|$, the running time of the code segment from line 01-07 is $\Theta(|V| + |E|) + \Theta(cL_V R_T)$. The time-complexity of the instruction in line 11-19 is $\Theta(c|V| L_V R_T |V_m| t_m)$. The code-segment line 09 to 19 has running time $\Theta(c|V| L_V R_T |V_m| t_m) + \Theta(|V|) + \Theta(c|V| L_V R_T) = \Theta(c|V| L_V R_T |V_m| t_m)$. The running time of the code segment line08-19 is $\Theta\left(c|V| L_V R_T |V_m|^2 t_m\right)$. The time complexity of line 20-25 is $\Theta|V| + \Theta(c|V| L_V R_T) + \Theta(c) = \Theta(c|V| L_V R_T)$. So, the running time of the overall algorithm is $\Theta(|V| + |E|) + \Theta(cL_V R_T) + \Theta\left(c|V| L_V R_T |V_m|^2 t_m\right) + \Theta(c|V| L_V R_T) = \Theta(|V| + |E|) + \Theta\left(c|V| L_V R_T |V_m|^2 t_m\right)$. Assuming that $|E|$ is upper bounded by $|V|^2$ and $|V_m|$ is upper boounded by $|V|$, the above expression can be simplified to $O\left(c|V|^3 L_V R_T t_m\right)$.

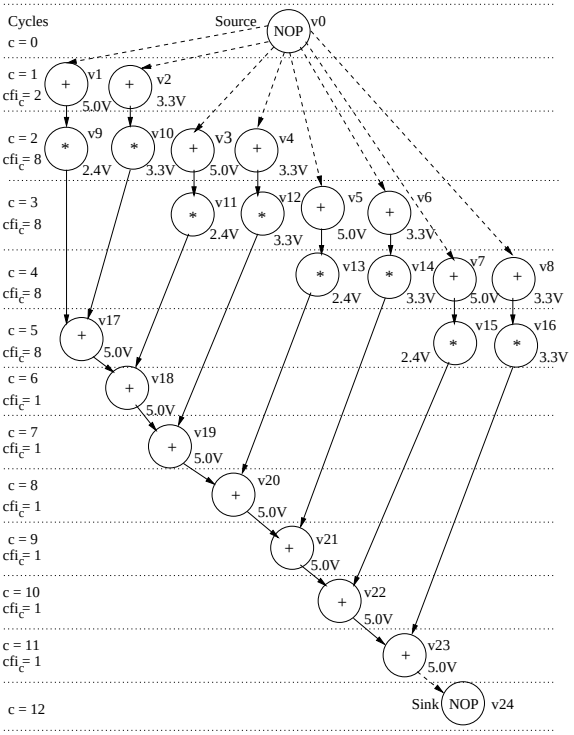


Fig. 11. Final schedule of FIR filter DFG (using RC-DFC)

7. EXPERIMENTAL RESULTS

Both RC-DFC and TC-DFC schedulers were implemented in C and tested with selected benchmark circuits. The benchmarks used are :
(1) Auto-Regressive (ARF) filter [Antola et al. 1998],
(2) Band-Pass filter (BPF) [Papachristou and Konuk 1990],
(3) Elliptic-Wave filter (EWF) [Kollig and Al-Hashimi 1997],
(4) DCT [Fetweis et al. 1993],
(5) FIR filter [Kumar and Bayoumi 1999], and
(6) HAL differential equation solver [Micheli 1994].
The FUs used are ALUs and multipliers. The energy values are computed using the datapath components given in [Mohanty and Ranganathan 2003; Mohanty et al. 2002]. The following notations are used to express the results :
(i) E_S and E_D are the total energy consumption (in pJ) for single supply voltage and multiple supply voltage operations respectively,
(ii) EDP_S and EDP_D are the energy-delay-products (in $10^{-18}J - s$) for single supply voltage and single frequency and for multiple supply voltage and dynamic clocking operations respectively,
(iii) T_S and T_D are the corresponding delays (in ns) for the two modes of operations and
(iv) N_S denotes the number of clock steps of the schedule for single supply voltage and and single frequency operations and

(v) N_D is the equivalent clock steps of T_D found out taking the delay of slowest functional unit as the base clock width in case of multiple voltage operation.
The percentage energy savings is calculated as, $S_E = \frac{(E_S - E_D)}{E_S} * 100$. In similar manner, we calculated percentage reduction in EDP which is denoted as S_{EDP}.

Table XI. Resource Constraints used for Performing our Experiments

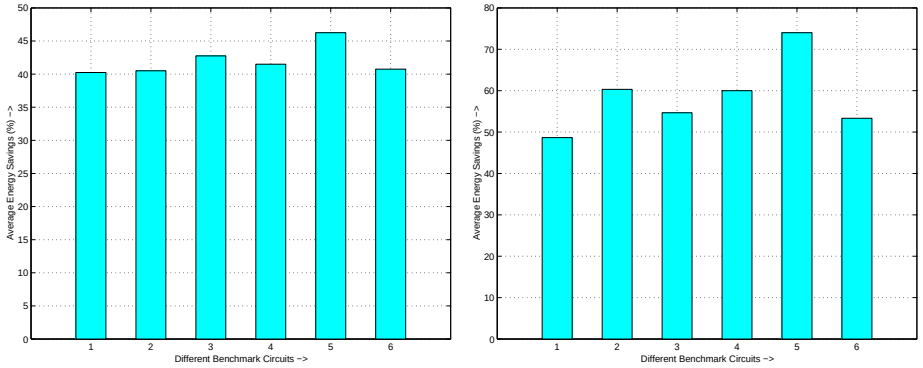
Multipliers		ALUs		Assigned
3.3 V	5.0 V	3.3 V	5.0 V	Serial No.
2	1	1	1	1
3	0	1	1	2
2	0	0	2	3
1	1	0	2	4

Table XII. Energy or EDP Estimates for Different Benchmarks using RC-DFC Scheduler

		Energy Estimates (pJ)			Energy-Delay-Product $\left(10^{-18} Js\right)$					Time Estimates			
		E_S	E_D	S_E	EDP_S	EDP_M	EDP_D	S_{EDP}		N_S	T_S	T_D	N_D
1	1	36168	21768	40	20093	24186	19954	17	1	10	556	917	9
A	2	36168	18205	50	20093	20227	16688	17	17	10	556	917	9
R	3	36168	19065	47	20093	21183	18006	15	10	10	556	944	9
F	4	36168	27617	24	26121	44877	31452	29	NA	13	722	1139	10
2	1	27654	16491	40	13827	16490	14659	11	NA	9	500	889	8
B	2	27654	14175	49	13827	14174	12600	11	9	9	500	889	8
P	3	27654	14827	46	13827	14827	12356	16	11	9	500	833	8
F	4	27654	20172	27	26118	42864	23253	45	11	17	944	1153	10
3	1	19404	10802	44	17248	19203	12902	32	25	16	889	1194	11
E	2	19404	10802	44	17248	19203	12902	32	25	16	889	1194	11
W	3	19404	10853	44	17248	19293	11154	42	35	16	889	1028	10
F	4	19404	11922	39	29106	40235	17055	57	41	27	1500	1431	12
4	1	30675	17846	42	25547	29743	26274	11	NA	15	833	1472	14
D	2	30675	17846	42	25547	29743	26274	11	NA	15	833	1472	14
C	3	30675	18008	41	25548	30013	25511	14	0	15	833	1416	13
T	4	30675	18008	41	49392	65278	37267	42	25	29	1611	2069	17
5	1	18678	9979	47	11414	12196	6653	45	42	11	611	667	7
F	2	18678	9979	47	11414	12196	6653	45	42	11	611	667	7
I	3	18678	10126	45	11414	12377	6470	47	43	11	611	639	6
R	4	18678	10127	46	15565	18987	12096	36	22	15	833	1194	10
6	1	13596	8927	34	3021	3967	2728	31	10	4	222	306	3
H	2	13596	6433	53	3021	2859	1966	31	35	4	222	306	3
A	3	13596	6648	51	3021	2954	2401	18	21	4	222	361	4
L	4	13596	10211	25	3777	6382	4396	31	NA	5	278	431	4

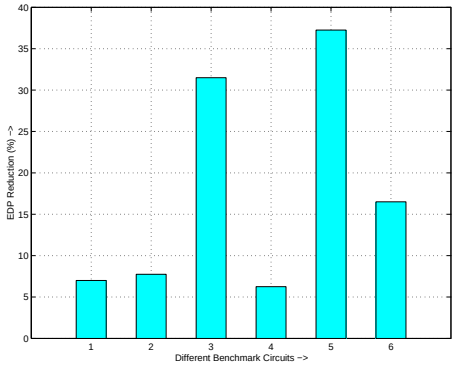
Table XIII. Configurations for Minimum EDP using RC-DFC

Benchmark	Multipliers		ALUs	
Circuits	3.3 V	5.0 V	3.3 V	5.0 V
ARF	3	0	1	1
BPF	2	0	0	1
EWF	2	0	0	1
DCT	1	1	0	1
FIR	2	0	0	2
HAL	3	0	1	1



(a) Energy reduction for RC-DFC

(b) Energy reduction for TC-DFC



(c) EDP reduction for RC-DFC

Fig. 12. Average Energy and EDP Reduction for Benchmarks

For RC-DFC scheduler, the experimental set-up is as follows. The algorithm was tested using the different sets of resource constraints listed in Table XI. The experimental results for various benchmark circuits are reported in Table XII. The energy estimation includes the energy consumption of the overhead units. It is assumed that each resource has equal switching activity. The results are reported for two supply voltages and for switching $= 0.5$. It is observed that the energy consumption is increased for higher switching and decreased for lower switching activity, but, under the assumption that switching is same for each resource, the percentage energy savings is not affected. There were very few resource constraints for which there was no reduction in the energy-delay-product for some benchmarks as reported "NA" in the table. The reduction in the energy-delay-product S_{EDP} is shown in two columns. The first column represents the reduction EDP_D with respect to EDP_M and the other column shows the reduction EDP_D with respect to EDP_S. In all benchmarks and for almost all resource constraints it is observed that $EDP_D < EDP_S < EDP_M$, which justifies the fact stated in Section 3 with a motivat-

ing DFG that using multiple supply voltages and dynamic frequency energy reduction is achieved without degrading performance. We also conducted experiments with three supply voltage levels and it is found that the percentage energy savings could only increase by 5%. Fig. 12(a) shows the percentage savings (average S_E) averaged over all resource constraints. From the chart it is evident that the scheduling yields approximately equal savings for all kinds of benchmark circuits. The EDP reduction (average S_{EDP}) averaged over all resource constraints are shown in Fig. 12(c). From the above, we may conclude that the scheduling algorithm yields appreciable energy savings and EDP reduction. In order to find the right combination of the types and the number of resources that will yield the best results in terms of energy reduction and high performance, we plotted energy consumption (%) versus time ratio ($\frac{T_D}{T_S}$), which is nothing but the the configuration corresponding to maximum S_{EDP}. Based on this analysis, the processor configurations that yield the lowest execution times for each benchmark are listed in Table XIII.

The TC-DFC scheduler was tested for three different time constraints: 1.5, 1.75 and 2.0 times critical path delay (T_{cp}). The voltage constraint is relaxed unlike the RC-DFC. The results for various benchmark circuits are reported in Table XIV. Fig. 12(b) shows the chart indicating the energy savings for different benchmarks averaged over all time constraints. Our observation is that circuits which require equal number of ALUs related operations (addition, subtraction or comparison) and multiplier operations save more energy. The energy savings increased as the time constraints relaxed from $1.5T_{cp}$ to $2.0T_{cp}$.

Table XIV. Energy Savings using TC-DFC Scheduler

Benchmark Circuits	Time Constraints	Energy consumption and savings		
		$E_S(pJ)$	$E_D(pJ)$	$S_E(\%)$
(1) ARF	$1.5T_{cp}$	36186	21491	41
	$1.75T_{cp}$	36186	18139	47
	$2.0T_{cp}$	36186	15274	58
(2) BPF	$1.5T_{cp}$	27672	15187	45
	$1.75T_{cp}$	27672	9350	66
	$2.0T_{cp}$	27672	8249	70
(3) EWF	$1.5T_{cp}$	19422	12335	36
	$1.75T_{cp}$	19422	8814	55
	$2.0T_{cp}$	19422	5341	73
(4) DCT	$1.5T_{cp}$	30675	14611	52
	$1.75T_{cp}$	30675	14489	53
	$2.0T_{cp}$	30675	7714	75
(5) FIR	$1.5T_{cp}$	18696	4910	74
	$1.75T_{cp}$	18696	4877	74
	$2.0T_{cp}$	18696	4820	74
(6) HAL	$1.50T_{cp}$	13614	7808	43
	$1.75T_{cp}$	13614	6821	50
	$2.0T_{cp}$	13614	4449	67

The energy savings from the proposed RC-DFC scheduling algorithm are listed alongwith other existing resource constrained multiple voltage scheduling algorithms in Table XV. The minimum and maximum range of energy savings are shown in the table. It is clear from the table that RC-DFC gives better energy savings for lesser time penalties.

Table XV. Savings in % and Time Penalty in Cycles for Various Resource Constrained Schedulers

Benchmark Circuits	RC-DFC		Shiue 2000		Sarrafzadeh 1999		Johnson 1997a	
	S_E	N_D	S_E	T	S_E	T	S_E	T
ARF	24-58	9-10	11-14	11-16	16-20	17-24	16-59	10-18
BPF	27-56	8-10	-	-	-	-	-	-
EWF	38-61	10-13	14-14	17-20	13-32	21-25	11-50	12-24
DCT	41-63	13-18	-	-	-	-	-	-
FIR	20-67	6-10	-	-	16-29	10-15	28-73	5-10
HAL	29-62	2-3	19-28	5-6	-	-	-	-

Table XVI. Percentage Savings for Various Time Constrained Schedulers

Benchmarks	TC-DFC	Chang 1997	Shiue 2000	Manzak 2002
ARF	41-58	40-63	38-76	25-61
BPF	45-70	-	-	-
EWF	36-73	44-69	13-76	10-55
FDCT	52-75	43-69	-	-
FIR	74-74	-	-	-
HAL	43-67	41-61	22-77	19-62

The energy savings for existing multiple supply voltage based time-constrained scheduling algorithms are shown in Table XVI. In all cases, the time constraints are $1.5 * T_{cp}$ to $2.0 * T_{cp}$. It may be noted that in Table XV and Table XVI we have shown a broad picture of the proposed work with respect to existing works in the literature. It is obvious that the existing methods use different benchmarks and different resource or time constraints in their experiments. Moreover, while existing works explore multiple supply voltages only, we used combined multiple supply voltages and dynamic frequency clocking. So, it is not possible to provide a fair comparison. However, to get a broad idea of our proposed work with respect to existing works, we have provided Table XV and Table XVI showing the range of energy reduction (not fixed value) for common benchmarks.

8. CONCLUSIONS

Our aim is to use frequency scaling concepts for energy-efficient high-performance special purpose processor (ASIC) design. The energy reduction is achieved by voltage reduction and the performance is maintained by using DFC alongwith multiple voltages. We developed resource-constrained and time-constrained datapath scheduling algorithms based on dynamic frequency clocking. The use of dynamic frequency clocking could generate enough slack to apply reduced voltages which in turn saves energy. It is observed that when using two supply voltage levels an average energy reduction of 41% and for three supply voltage levels, an average reduction of 46% is obtained for the benchmarks using the RC-DFC algorithm. Similarly, for TC-DFC, an average energy reduction of 46% (for $1.5 \times T_{cp}$) and 68% (for $2.0 \times T_{cp}$) are obtained. The processor configurations for various benchmark circuits that would result minimum energy-delay-product were determined through experiments. The integration of such a scheduler into a low power datapath synthesis tool will significantly benefit low power processor design especially for data intensive applications.

REFERENCES

ANTOLA, A., PIURI, V., AND SAMI, M. 1998. A Low-Redundancy Approach to Semi-Concurrent Error Detection in Datapaths. In *Proceedings of the Design Automation and Test in Europe*. 266–272.

BENINI, L., MACII, E., PNOCINO, M., AND MICHELI, G. D. 1998. Telescopic Units : A New Paradigm for Performance Optimization of VLSI Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 17,* 3 (Mar), 220–232.

BENINI, L., MICHELI, G. D., LIOY, A., MACII, E., ODASSO, G., AND PONCINO, M. 1999. Automatic Synthesis of Large Telescopic Units Based on Near-Minimum Timed Supersetting. *IEEE Transactions on Computers 48,* 8 (Aug), 769–779.

BRYNJOLFSON, I. AND ZILIC, Z. 2000a. Dynamic Clock Management for Low Power Applications in FPGAs. In *Proceedings of the IEEE Custom Integrated Circuits Conference*. 139–142.

BRYNJOLFSON, I. AND ZILIC, Z. 2000b. FPGA Clock Management for Low Power. In *Proceedings of the International Symposium on FPGAs*. 219–219.

BURD, T. AND BRODERSEN, R. W. 1995. Energy Efficient CMOS Microprocessor Design. In *Proceedings of the 28th Hawaii International Conference on System Sciences*. 288–297.

BURD, T., PERING, T. A., STRATAKOS, A. J., AND BRODERSEN, R. W. 2000. A Dynamic Voltage Scaled Microprocessor System. *IEEE Journal of Solid-State Circuits 35,* 11 (Nov), 1571–1580.

CHANG, J. M. AND PEDRAM, M. 1997. Energy Minimization using Multiple Supply Voltages. *IEEE Transactions on VLSI Systems 5,* 4 (Dec), 436–443.

FETWEIS, G., CHIU, J., AND FRAENKEL, B. 1993. A Low-Complexity Bit-Serial DCT/IDCT Architecture. In *Proceedings of the IEEE International Conference on Communications*. 217–221.

GRUNWALD, D., LEVIS, P., AND FARKAS, K. I. 2000. Policies for Dynamic Clock Scheduling. In *Proceedings of the 2000 Operating Systems Design and Implementation*.

HSU, C. H., KREMER, U., AND HSIAO, M. 2000. Compiler-Directed Dynamic Frequency and Voltage Scheduling. In *Proceedings of the Workshop on Power-Aware Computer Systems*. 65–81.

JOHNSON, M. AND ROY, K. 1997a. Datapath Scheduling with Multiple Supply Voltages and Level Converters. *ACM Transactions on Design Automation of Electronic Systems 2,* 3 (July), 227–248.

KIM, J. M. AND CHAE, S. I. 1996. New MPEG2 Decoder Architecture using Frequency Scaling. In *Proceedings of the IEEE International Symposium on Circuits and Systems*. 253–256.

KOLLIG, P. AND AL-HASHIMI, B. M. 1997. Simultaneous Scheduling, Allocation and Binding in High Level Synthesis. *IEE Electronics Letters 33,* 18 (Aug.), 1516–1518.

KUMAR, A. AND BAYOUMI, M. 1999. Multiple Voltage-Based Scheduling Methodology for Low Power in the High Level Synthesis. In *Proceedings of the International Symposium on Circuits and Systems (Vol. 1)*. July, 371–379.

LIN, Y. R., HWANG, C. T., AND WU, A. C. H. 1997. Scheduling Techniques for Variable Voltage Low Power Design. *ACM Transactions on Design Automation of Electronic Systems 2,* 2 (Apr), 81–97.

MANZAK, A. AND CHAKRABARTI, C. 2002. A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages. *IEEE Transactions on VLSI Systems 10,* 1 (Feb), 6–14.

MARTIN, T. L. AND SIEWIOREK, D. P. 2001. Nonideal Battery and Main Memory Effects on CPU Speed-Setting for Low Power. *IEEE Transactions on VLSI Systems 9,* 1 (Feb), 29–34.

MICHELI, G. D. 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, Inc.

MOHANTY, S. P. AND RANGANATHAN, N. 2003. Energy Efficient Scheduling for Datapath Synthesis. In *Proceedings of the International Conference on VLSI Design*. 446–451.

MOHANTY, S. P., RANGANATHAN, N., AND CHAPPIDI, S. K. 2003. An ILP-Based Scheduling Scheme for Energy Efficient High Performance Datapath Synthesis. In *Proceedings of the International Symposium on Circuits and Systems (Vol. 5)*. 313–316.

MOHANTY, S. P., RANGANATHAN, N., AND KRISHNA, V. 2002. Datapath Scheduling using Dynamic Frequency Clocking. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. 65–70.

MONTEIRO, J., DEVADAS, S., ASHAR, P., AND MAUSKAR, A. 1996. Scheduling Techniques to Enable Power Management. In *Proceedings of the ACM / IEEE Design Automation Conference*. 349–352.

MUSOLL, E. AND CORTADELLA, J. 1995. Scheduling and Resource Binding for Low Power. In *Proceedings of the 8th International Symposium on System Synthesis*. 104–109.

PAPACHRISTOU, C. A. AND KONUK, H. 1990. A Linear Program Driven Scheduling and Allocation Method. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*. 77–83.

PEDRAM, M. 1996. Power Minimization in IC Design: Principles and Applications. *ACM Transactions on Design Automation of Electronic Systems 1,* 1 (Jan.), 3–56.

PERING, T., BURD, T., AND BRODERSEN, R. W. 2000. Voltage Scheduling in the lpARM Microprocessor System. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 96–101.

POUWELSE, J., LANGENDOEN, K., AND H.SIPS. 2001a. Dynamic Voltage Scaling on a Low-Power Microprocessor. In *Proceedings of the 7th International Conference on Mobile Computing Network*.

POUWELSE, J., LANGENDOEN, K., AND H.SIPS. 2001b. Energy Priority Scheduling for Variable Voltage Processor. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 28–33.

RANGANATHAN, N., VIJAYKRISHNAN, N., AND BHAVANISHANKAR, N. 1998. A Linear Array Processor with Dynamic Frequency Clocking for Image Processing Applications. *IEEE Transactions on Circuits and Systems for Video Technology 8,* 4 (August), 435–445.

SARRAFZADEH, M. AND RAJE, S. 1999. Scheduling with Multiple Voltages under Resource Constraints. In *Proceedings of the IEEE Symposium on Circuits and Systems (Vol. 1)*. 350–353.

SHIUE, W. T. AND CHAKRABARTI, C. 2000. Low-Power Scheduling with Resources Operating at Multiple Voltages. *IEEE Transactions on Circuits and Systems-II : Analog and Digital Signal Processing 47,* 6 (June), 536–543.

Submitted on November 2003, Revised on April 2004, Final manuscript prepared on August 2004.