High Level Synthesis Techniques for Low Power Design : A Tutorial Review

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

Abstract

High-level synthesis is defined as a translation process from a behavioral description into structural description. The high-level synthesis process consists of three independent phases such as allocation, assignment (binding) and scheduling. The order of the three phases varies depending on the design flow. There are three important quality measures used to support design decision, namely size, performance (throughput) and power consumption. Recently, the power consumption has become 3rd important parameter. The aim of low power high-level synthesis is to schedule operations to minimize switching activity and select low power modules while satisfying timing constraints. This paper surveys the works of various researchers who have addressed the problem of minimizing power dissipation during various stage of high-level synthesis.

1 Introduction

In the *analysis* problem we study the characteristics or behavior of a given circuit. *Synthesis* process is reverse of analysis process. In synthesis, a circuit is to be designed that has to have a specified characteristics/behavior. The *high-level synthesis* process is defined as a translation process from behavioral description to a structural description [12, 13, 14]. This is analogous to "compiler" that translates high-level language like C/Pascal to assembly language. The high-level synthesis is also known as *behavioral-level* synthesis or *algorithm-level* synthesis.

The high-level synthesis process consists of three phases [14]:

- allocation
- assignment (binding)
- scheduling.

The three phases are independent. The order of these three phases varies depending on the design flow. The allocation phase determines how many instances of each resources are needed, the binding phase selects on what resource a computational operation will be performed and the scheduling determines when the operation will be executed.

Because of the increasing demand for personal computing devices and wireless communications equipment the demand for designing low power consuming circuits has increased. "Power" has become the 3rd important parameter alongwith area and throughput. The need for low power synthesis is driven by several factor [19, 26, 27]:

- **Increased demand of portable systems :** Emergence of portable devices like laptop computers, mobile phones etc. for which battery life is an important factor.
- **Thermal considerations :** If power dissipation can be reduced the cost of cooling and packaging would be reduced.
- **Environmental concerns :** The smaller the power dissipated, the lower the heat pumped into the rooms, the lower electricity consumed and, therefore the less impact on the environment.
- **Reliability issues :** If the power consumption is higher, the temperature is increased. This may lead to the phenomenon like *elctromigration* and *hot-electron* effects. This causes reduction in reliability of the system. In fact, it is seen that every 10^oC increase on operating temperature roughly doubles failure rate for the components.

The average power dissipation is a critical design concern for the 1st three factors whereas peak power is the critical design concern for the reliability issues.

Reduction in power in the VLSI ciruits can be achieved by the following four ways:

- Reducing chip and package capacitance
- Scaling the Supply Voltage
- Employing better design techniques
- Using power management strategies.

Reduction in chip and package capacitance can be done through process development with partially/fully depleted wells, submicron device size and advanced interconnect substrates. This approach is very effective but is quite expensive. Scaling the supply voltage approach is very effective in reducing dynamic power dissipation but it needs new IC fabrication technique and support circuitry for voltage-level conversions. With lesser investment power reduction can be achived if better design technique is adopted. Using power management strategies significant power savings can be achieved. During high-lvel synthesis power saving is achieved using various power management strategies.

Traditionally high-level synthesis has ignored power minimization. Due to above mentioned reasons it has become necessary to develop high-level synthesis techniques

that target lower power dissipation in the circuits [14]. Researcher have focused on the problem of minimizing power dissipation during various phases of high-level synthesis. This paper surveys those research works.

The paper is organized as follows. Section2 discusses high-level synthesis in general and also tells why we go for high-level synthesis. The various sources of power consumption are discussed in Section3. Section4 discusses algorithms those use "multiple supply voltage" for minimizing power dissipation. Section5 discusses algorithms based on frequency variation for power minimization. Section6 presents the algorithms that use "switching activity" as parameter for power reduction. The "module/register allocation and binding" algorithms for low power synthesis have been highlighted in Section7. Section8 outlines some guidelines for low power synthesis.

2 Fundamentals of High-Level Synthesis

The task of synthesis process is to take the specifications of the behavior required for a system and a set of constraints and goals to be satisfied, and to find a structure that implements the behavior while satisfying the goals and constraints [1, 2]. The "behavior" of the system refers to the ways the system or its component interact with their environment (mapping from inputs to outputs). The "structure" refers to the set of interconnected components that constitute the system (described by a netlist). Usually there are many different structures that can be used to realize a given behavior. So one of the tasks of synthesis is to find the structure that best meets the constraints, such as area, cycle time, power, cost, and so on. The aim of low power synthesis is to meet the power constraint while roughly satisfying the other constraints. Since digital circuits are designed at several levels of abstraction, synthesis can take place at various levels of abstraction as shown in Fig.1.

High-level synthesis is a transformation from an algorithm level specification of the behavior of a system to a register-transfer level specification [1, 2, 12, 13]. The number of reasons of being high-level synthesis popular are the followings [1, 2]:

- **Shorter design cycle :** If more of design process is automated faster products can be made available at cheaper price.
- **Fewer errors :** Since the synthesis process can be verfied easily chances of error is less and less debugging.
- **The ability to search the design space :** As a good synthesis system can produce several designs in a small time the designer has more flexibility to choose proper design considering different trade-offs.
- **Documenting the design process :** The automated system keeps record of design decisions and effect of those decisions.
- **Availability of IC technology to more people :** Since the design is automated non-experts get oppertunity to produce chip satisfying a set of specifications.

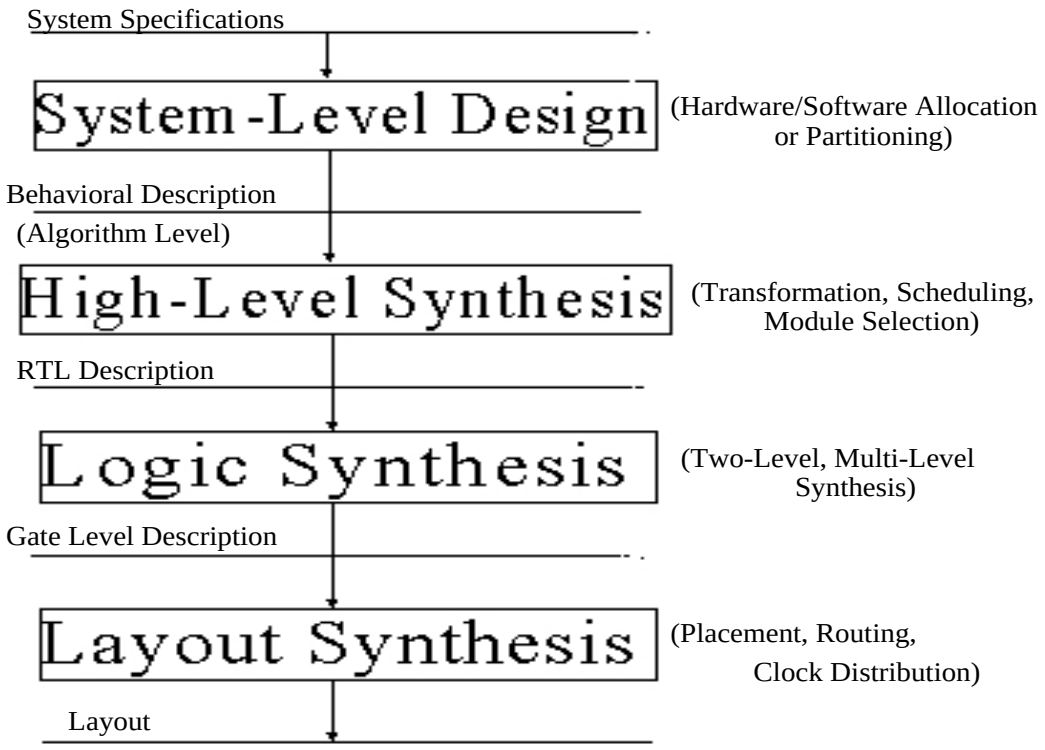

Figure 1: Synthesis Flow

The detailed phases of high-level synthesis are shown in Fig.2. The input to the high-level synthesis algorithm is usually a control/data flow graph (**CDFG**). The primary jobs involved in behavioral synthesis are scheduling and allocation. *Scheduling* assigns each operation to an execution cycle, thus fixing the cycle-by-cycle behavior of the circuit. Scheduling algorithm can be gropued into two classes [1, 2, 3] :

- Tranformational
- Iterative/Constructive

The tranformational class of algorithms begin with a defualt schedule and apply transformations to obtain other schedules. Transfomational algorithms differ in how they choose what transformations to apply. The iterative/constructive class of algorithms build up a schedule by adding operations one at a time until all the operations have been scheduled. These algorithms differ in how the next operation to be scheduled is choosen and in how they determine where to schedule each operation. The scheduling algorithms like As Soon As Possible (ASAP), As Late As Possible (ALAP), List scheduling, Freedom-based scheduling, Force-directed scheduling, and Force-Direct List scheduling belong to iterative/constructive class.

The *allocation* process binds register and functional modules respectively to variables and operations in the CDFG and specifies the interconnection among modules and registers in terms of multiplexers or buses. Allocation is further divided into

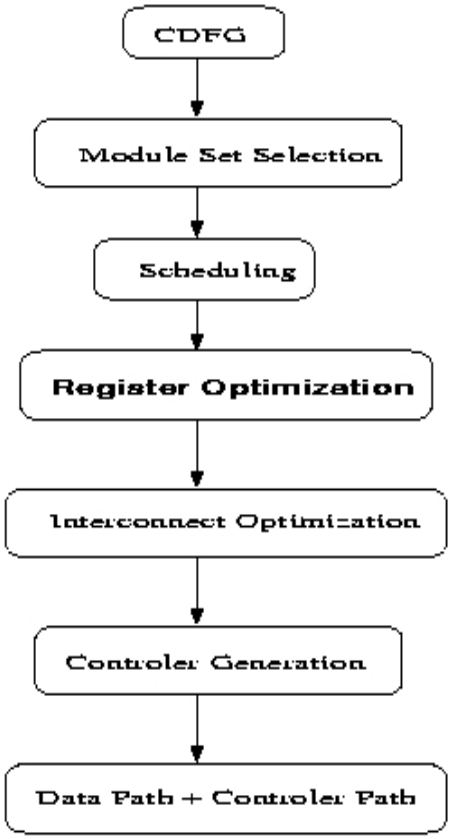

Figure 2: Phases of High-Level Synthesis

following tasks:

- registers allocation
- module allocation
- interconnect allocation.

Usually these three operations are handled sequentially, and the resulting circuit is often require more interconnect than necessary. Resource sharing refers to the use of same hardware resource (functional unit or register) to perform different operations or store more than one variable. Register sharing is also known as register optimization.

The high-level synthesis is different form *logic* synthesis. In the case of logic synthesis, the system is specified in terms of logic equations, which must be optimized and mapped into a given technology. Logic synthesis can be used after high-level synthesis in a design process.

3 Sources of Power Dissipation

The sources of power consumption or dissipation in digital CMOS circuit are as listed below [9, 11, 19, 20, 26, 27]:

- **Leakage Current**: This is determinde by the fabrication technology and consists of reverse bias current in the parasitic diodes and subthreshold current that arises from the invesion charges that exists at the gate voltages below the threshold voltage.
- **Standby Current**: It is the DC current drawn continuously from V_{dd} to ground.
- **Short-Circuit Current**: This is the current due to the DC path between the supply and ground during output transition.
- **Capacitance Current**: This flows to charge and discharge capacitance loads during logic changes.

The details of power dissipations is shown in Fig.3.

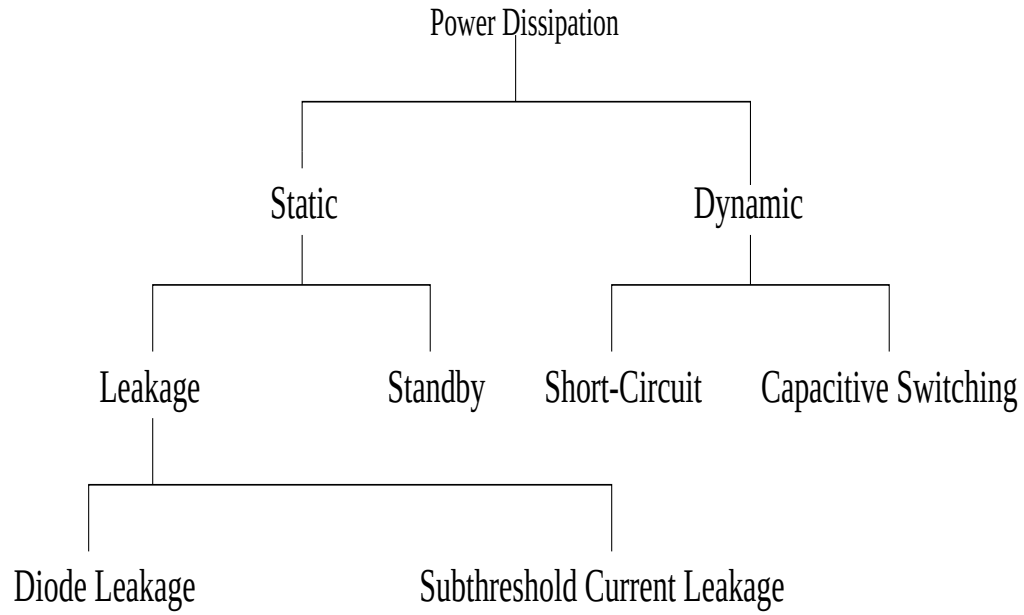

Figure 3: Sources of Power Dissipation

Capacitive Switching power dissipation is caused by charging and discharging of parasitic capacitance in the circuit. It is given by equation.1

$$P_{dynamic} = \frac{1}{2} C_L V_{dd}^2 N f \tag{1}$$

where C_L is load capacitor, V_{dd} is supply voltage, N is average or expected number of transitions per clock cycle (switching activity), and f is the clock frequency. During transition from either 0 to 1 or 1 to 0, both NMOS and PMOS are ON for a short period of time. Because of this there is flow of current from V_{dd} to V_{ss} (short

current pulse). The power dissipation corresponding to this is called *short-circuit power* dissipation. which is quantified as in equation.2

$$P_{short} = \frac{\beta}{12}(V_{dd} - 2V_t)^3 \frac{t_{rf}}{t_p} \tag{2}$$

where β is the transistor gain factor, V_{dd} is supply voltage, V_t is the threshold voltage, t_{rf} is the rise/fall time (under the assumption that $t_r = t_f$) and t_p is the period of the input waveform. The *leakage power* dissipation occurs because of reverse-biased diode (formed between diffusion regions and substrate) current and subthreshold current. The static power dissipation is the product of the transistor leakage current and the supply voltage. The total static power dissipation is obtained from equation.3:

$$P_{static} = \sum_{i}^{n} leakagecurrent * supplyvoltage \tag{3}$$

where n is the number of transistors.

Leakage power dissipation is small in comparison to other components. In a well designed circuit, short-circuit power dissipation is less than 20% of dynamic power [21]. This shows that the main power dissipation that need to be taken care of is the dynamic power dissipation. From the dynamic power dissipation expression given in equation.1 we can say that the parameters that can be varied to affect power as well as energy consumption are :

- supply voltage
- the clock frequency
- the switching activity per clock cycle at various signals in the circuit
- the parasitic capacitance.

It is important to note that these parameters are not independent. It is necessary to take into account the interactions and trade-offs among these parameters to minimize power consumption [26]. The key principles used for low-power design are as follows [20] :

- using the lowest possible supply voltage,
- using the smallest geometry, highest frequency devices, but operating them at lowest possible frequency,
- using parallelism and pipelining to lower required frequency of operation,
- power management by disconnecting the power source when the system is idle, and
- designing systems to have lowest requiremnts on subsystem performance for the given user level functionality.

4 Multiple Supply Voltage Based Scheduling Algorithms

It is possible to use high supply voltage in the critical paths of a design to achieve required performance while the off-critical paths of the design use lower supply voltage to achieve low-power dissipation. Whenever a logic gate operating at a low voltage is driving a logic gate operating at a higher voltage, level converters are required. On the other hand, when a logic gate operating at a higher voltage drives logic operating at a lower voltage, a level converter may not be required. As a whole, the multiple-supply voltage design may affect IC layout in following ways [20]:

- If the multiple supplies are generated off-chip, additional power and ground pins will be required.
- It may be necessary to partition the chip into separate regions, where all modules in a region operate at the same voltage.
- Some kind of isolation will be required between regions operated at different voltages.
- There may be some limit on the voltage difference that acn be tolerated between the regions.
- Protection against latch-up may be needed at the logic interfaces between regions if different voltages.
- New design rules for routing may be needed to deal with signals at one voltage passing through a region at another voltage.

The aim of scheduling phase is to assign operations to control steps so as to minimize a given objective function while meeting constraints. A control step is the fundamental sequencing unit in synchronous system and it corresponds to a clock cycle. There are many scheduling algorithms with objective function as power/energy minimization. The multiple-voltage scheduling (MVS) is **NP-hard**. In this section we discuss few of those algorithms.

Johnson and Roy [15] present a method **MESVS** (Minimum Energy Schedule with Voltage Selection) by using **ILP** (Integer Linear Programming) to optimize the schedule, supply voltage levels, and allocation of resources. Multiple supply voltage selection and level conversion costs have been incorporated into energy optimization of datapaths. The *input* to the MESVS are as listed below :

- DFG (a directed acyclic graph) specifying operations, data flows and latency constraints
- specifications of discrete set of permitted supply voltages
- a limit on the number of supply voltages that can be selected
- a minimum difference between voltages that can be selected
- average switching activities for each datapath operation
- nominal propagation delay

- average energy dissipation values for each datapath resource.

The *objective function* for MESVS is an estimate of datapath energy dissipation expressed as a function of supply voltages as given in equation4.

$$\begin{aligned} energy \;=\; & \sum_{i}\sum_{l}\sum_{s}(x_{i,l,s} * (onrgy(i,s) + rnrgy(i,s))) \\ & + \sum_{i,j\epsilon E}\sum_{s1}\sum_{s2}(cnrgymulti(i,j) * cnrgy(s_1,s_2) * v_{i,j,s_1,s_2}) \end{aligned} \quad (4)$$

The parameter arrays $onrgy(i,s)$ and $rnrgy(i,s)$ contain estimate of the energy expended to perform operation i an store the result for a single change of input values at voltage s. $cnrgymult(i,j) * cnrgy(s_1,s_2)$ gives the energy dissipation of the level conversion from voltage s_1 to s_2 applied to a single change in the output operation i destined for operation j. The parameter arrays give energy estimates for each possible choice of supply voltages. The voltage assignments indicated by $x_{i,l,s}$ and v_{i,j,s_1,s_2} are used to select one energy estimate from the parameter arrays for each operator, register, and level converter. The *output* of the MESVS are the followings :

- a datapath schedule
- an energy estimate
- selection of optimal set of supply voltage
- assignment of supply voltage to each operation
- allocation of resources to each supply voltage.

Since the different resources need to operate at different voltages level conversion is needed. There are four possible schemes :

- omitting the level converter
- using a chain of inverters
- using an active or passive pullup
- using dual cascode voltage switch (DCVS) circuit.

The authors used DCVS circuit scheme. The authors claim that energy saving in the range of 46%-58% is obtained compared to 5V operation. The other observation is that two supply voltages reduce power dissipation substantially in some cases. In no case did three supplies decreased energy by more than 5% compared to two supplies.

Johnson and Roy [22] present an algorithm called **MOVER** (Multiple Operating Voltage Energy Reduction) to minimize datapath energy dissipation. Energy saving ranging from 0-50% obtained at the cost of area penalty of 0-170%. The maximum energy saving is 15% if three supplies used instead of two. The MOVER generates one, two, and three supply voltage designs for consideration by the circuit designer. The user has control over latency constraints, resource constraints, total number of control steps, clock period, voltage range, and number of power supplies. The MOVER iteratively searches the voltage range for minimum voltages that will be feasible in one,two, and three supply voltages. The MOVER uses an ILP (Integer Linear Programming) for followings :

- to evaluate the feasibility of candidate supply voltage selections
- to partition operations among different power supplies
- to produce a minimum area schedule under latency constraints once voltages have been selected.

The that objective function estimates the energy required for one execution of the datapath as a function of the voltage assigned to each operation is given in equation5.

$$
\begin{aligned}
Energy \quad = \quad & \\
& \sum_{j \epsilon V_{free} \cap V_{oper}} \sum_{l \epsilon R_j} \sum_{s=1}^{2} x_{j,l,s} * (onrg(j, v_s, c_{reg}) \\
& + rnrg(v_s, c_{fanout(i)}) + cnrg_{free}(j, s)) \\
& + \sum_{j \epsilon V_{fix} \cap V_{oper}} \sum_{l \epsilon R_i} x_{j,l} * (onrg(j, v_j, c_{reg}) \\
& + rnrg(v_j, c_{fanout(i)}) + cnrg_{fix}(j)) \qquad (5)
\end{aligned}
$$

For each j that has not been fixed to a supply voltage ($j \epsilon V_{free}$), the first nested summation accumulates the energy operation j ($onrg(j, s_j, c_{reg})$), the register at the output operation j $rnrg(s_j, fanout_j)$), and any level conversion required at the input to j ($cnrg_{free}(j, s)$). s_j is the index of the supply voltage assigned to operation j. $fanout_j$ is the fanout capacitive load on operation j. c_{reg} is the input capacitance of a register to which the operation ooutput is connected. The decision variables $x_{j,l,s}$ are used to select which lookup table values for operator, register. and level conversion energy are added into the total energy. V_{oper} is the set of DFG vertices (operations) that have been fixed to a particular voltage. V_{free} is the set of vertices that have not previously been fixed to a voltage. The MOVER has the following phases :

- determining maximum and minimum bounds on the time frame in which each operation must execute
- searching for minimum voltage
- partitioning datapath operations into two groups (those which will be assigned to higher supply voltage and those which will be assigned to a lower supply voltage).
- for three supply voltage schedule, portioning the lower voltage group.

The MOVER algorithm [22] is similar to MESVS algorithm [15] in the following ways :

- both use ILP formulation
- behavior with respect to latency, resource, ad supply voltage constraints
- both use DCVS (Differential Cascode Voltage Switch).

The difference between the to is that MESVS can only select a discrete set of voltages, whereas MOVER can select a continuous range of voltages. The ILP formulation no doubt handles timing and resource constraints and accounts for the cost if level shifters, however, it has following drawbacks :

- it does not address conditional branches
- doesn't consider functional pipelining
- energy model used is data-intensive which ignores the effect of input activities on the energy dissipation of a module
- it has exponential worst-case complexity and can't handle large example
- ILP is difficult to solve.

Chang and Pedram [7] present a **dynamic programming** technique for solving multiple supply voltage scheduling problem. They say, ILP formulation in [15, 22] doesn't take advantage of problem structure and ILP is difficult to solve. So they go for dynamic programming. They deal with both pipelining and non-pipelining datapath. Using three supply voltage levels an average saving of 40.19% have been obtained. The energy model used is given by equation6.

$$E_{FU_i} = F_i(\alpha_{i,1}, \alpha_{i,2})V_i^2 \tag{6}$$

where V_i is the supply voltage of functional unit FU_i, $\alpha_1^{FU_i}$ and $\alpha_2^{FU_i}$ are the average switching activities on the first and second input operands of FU_i, respectively, FU_i is a function of $\alpha_1^{FU_i}$ and $\alpha_2^{FU_i}$ and in general may be nonlinear. The algorithm has pseudo-polynomial complexity. The algorithm supports conditional branches.

5 Algorithms Based on Frequency Variation

In section3 we saw that to reduce dynamic power dissipation clock frequency should be reduced. In this section we study different algorithms based on frequency variation.

Ranganathan, et. al. [4] introduce the concept of **DFC** (Dynamic Frequency Clocking). The DFC scheme is more suitable for data flow intensive application (such as DSP and image processing). The DFC scheme is a combination of three concepts :

- reconfigurable architecture
- frequency synthesizer
- clock dividing strategy.

In the reconfigurable architecture, frequencies are switched as the circuit changes while in DFC scheme, frequency switching occurs based in the units being used. In the clock divider strategy, each unit receives a separate clock operating at a different frequency, whereas in DFC strategy, the same clock switches dynamically. Based on the DFC scheme the authors have proposed a linear VLSI array SIMD processor. The processor operates between 400Mhz to 50 MHz based on the operation being performed. The dynamic clocking unit (DCU) shown in Fig.4 generates different clock frequencies based on instruction words. The DCU contains two encoders :

- Clock encoder
- Enable encoder.

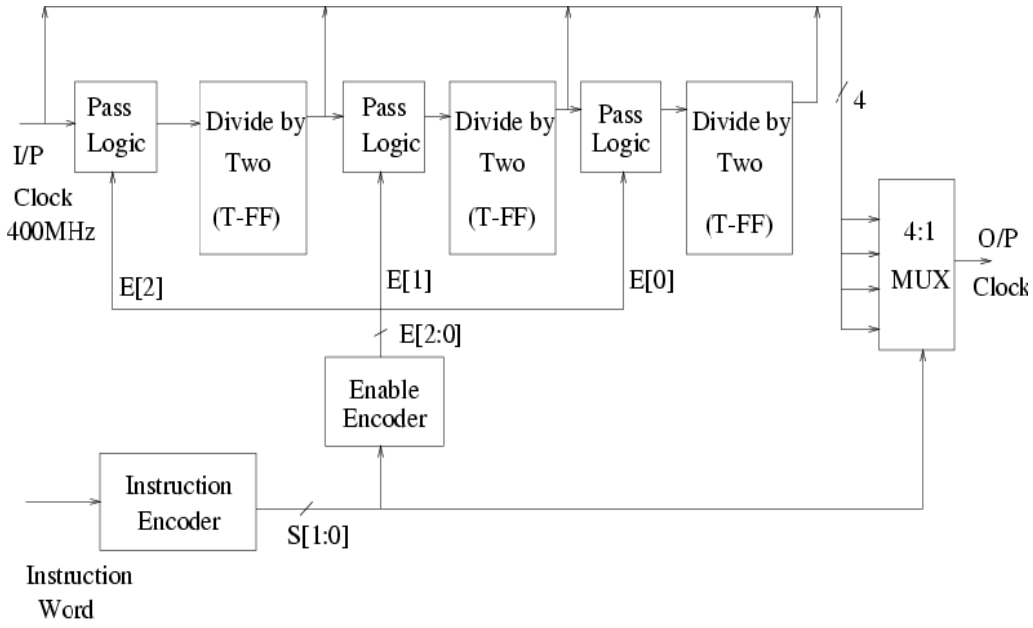

Figure 4: Dynamic Clocking Unit

Based on the status of the function enable bits each instruction is classified as belonging to one of the four instruction classes. The processor is a linear array of N PEs for an NxN image as shown in Fig.5. Each PE executes the same operation, but on different data sets stored within their local memory. A global clock is distributed to all PE's. The host computer broadcasts microinstructions to all the PEs in the array. The organization of each PE is shown in Fig.6. Each PE contains many functional units. Each functional unit is provided with a result register at its output to store computation results temporarily. The PE has a dynamic clocking unit (DCU) and Neighbour Communication Unit (NCU). The NCU is used to tranfer data to and from the neighboring PEs. Communications among the various funcational blocks within a PE is supported by a 64-bit internal bus. The PE executes one instruction every clock cycle. The instruction word (IW) is 57-bits wide (Fig.7). The IW has been organized to exploit the parallelism among the various funcational blocks. The author claim that speedup of the dynamic frequency over single constant frequency clocking ranges from 1.79 to 3 for the proposed architecture which uses single-cycle nonpipelined units.

Ranganathan, et. al. [5] use Dynamic Frequency Clocking **DFC** scheme alongwith Multiple Voltage Scaling to design energy efficient datapath. The Dynamic Frequency Clocking and Multiple Voltage Scaling **DFMVS** has two modules such as :

- Dynamic Freq-Sched
- Modify-Sched.

Dynamic Freq-Sched generates an initial schedule in which the control steps are clocked at different frequencies. *Modify-Sched* incorporates a schedule modifier, that regroups the operations of the initial schedule such that multiple voltages can be

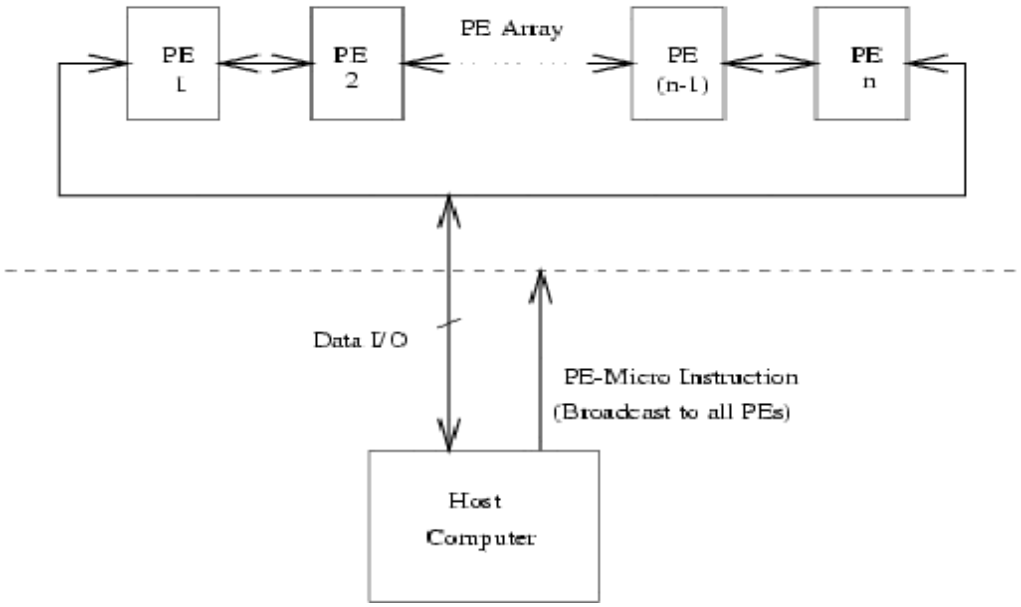

Figure 5: System Architecture

used to reduce the energy consumption. DFC utilizes the fact that different units such as adders, multipliers can be clocked at different frequency based on their critical path delay. Dynamic Clocking Unit (DCU) generates appropriate clock frequency based on units operating at that time. Since DCU is an additional circuit used, areawise this scheme is inefficient; so their has to be a trade-off. Ranganathan, et. al. [6] discuss the applicability of DFC in pipelined structures.

Papachristou, et al. [24] have presented a multiple clocking scheme for low-power RTL design. Their contribution is in two folds:

- Using non-overlapping multiple clocking to design a partitioned datapath, so that each partition is assigned a distinct clock. If the effective frequency of entire circuit is f, then the working frequency running each partition module is f/n; the number of partitions being n. By this process, inactive partitions are literally "turned off" during their duty cycle to reduce power consumption.
- A multiple clocking allocation algorithm is proposed for power reduction.

Two distinct advantages of the porposed scheme over the conventional schemes are given below :

- Use of non-overlapping multiple clocks to power storage components and
- ALU isolation is based on clock-partitioning by allocation, avoiding extra turn-off logic.

The authors proposed two allocation techniques, namely "spilt allocation" approach and "integrated allocation" of the partitions. The proposed multiple clocking scheme can't be used for pipelined structures. The power saving achieved by this scheme is up to 50%.

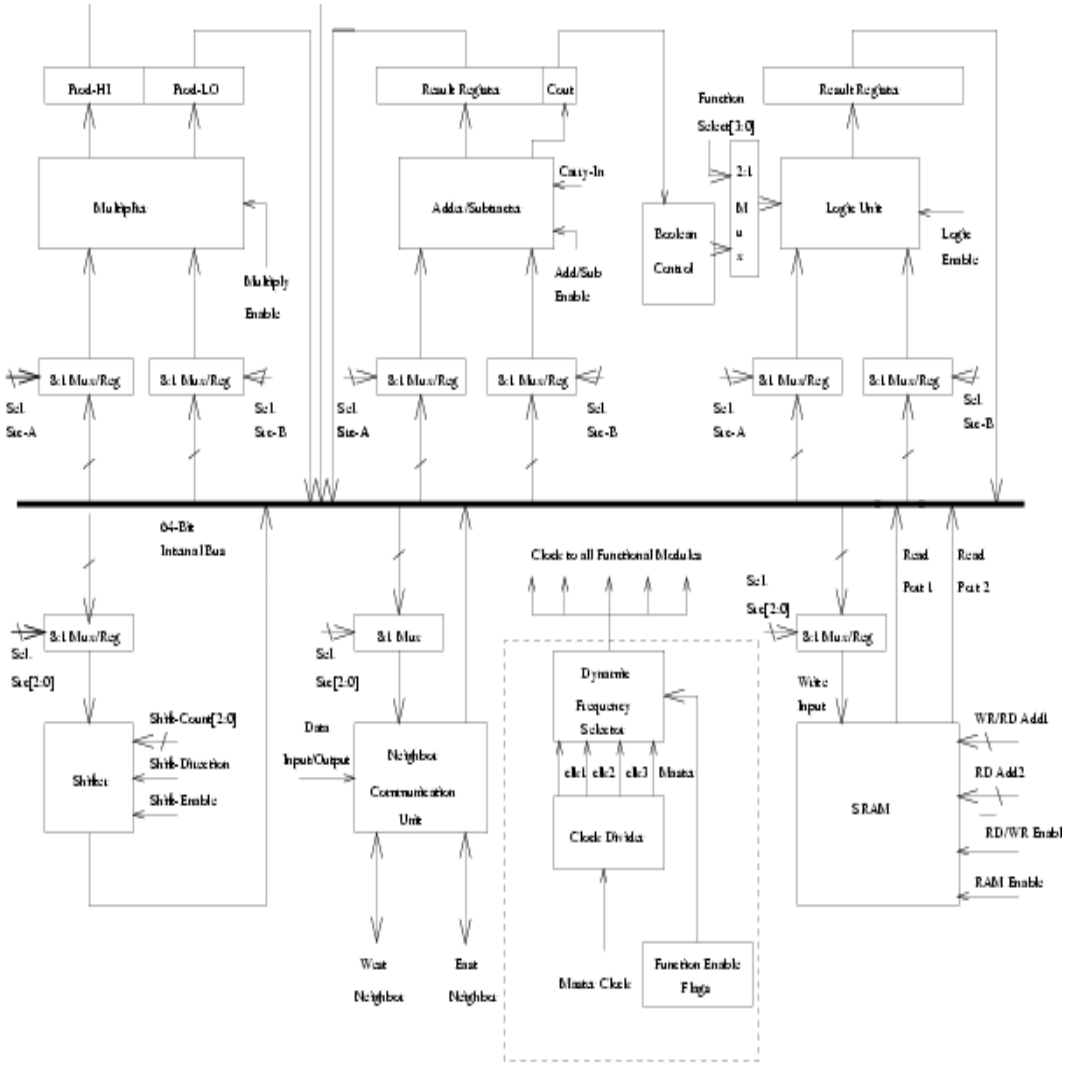

Figure 6: PE Organization

Hsu, et al. [25] proposed scheduling algorithm called Compiler-Directed Dynamic Frequency and Voltage Scheduling. A new compilation strategy has been proposed that provides same overall execution time while reducing the power. The compiler assigns a clock frequency and a voltage level for its execution. The authors have mentioned that modern architecture have a large gap between the speeds of the memory and processor. The techniques used to bridge this gap are:

- memory pipelining
- cache hierarchy
- large register sets.

These architectural features exploit temporal and spatial localities. But, if the computations have less spatial/temporal locality these schemes will not give desired perfomance. So the authors advocate for the new compilation stratrgy. The

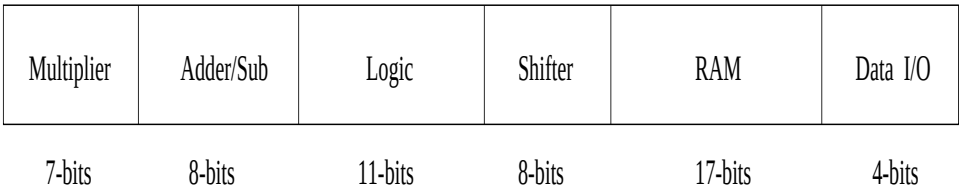

Figure 7: Instruction Format

maximum clock speed for a supply voltage can be estimated as:

$$f_{max} = k\frac{(V_{dd} - V_T)^{\alpha}}{V_{dd}} \tag{7}$$

where, V_T is the threshold voltage, α is a technology dependent factor and k is a constant. Total execution time (T) is divided into three portions:

- CPU busy
- memory busy
- both busy.

Mathematically,

$$T = cpuBusy + memBusy + bothBusy \tag{8}$$

If CPU speed is reduced by a factor δ, then the new execution time becomes:

$$T_{new}(\delta) = \delta * cpuBusy + max(memBusy + bothBusy, \delta * bothBusy) \tag{9}$$

The condition that must be satisfied is that both $T_{new}(\delta)$ and T should be very close. The compilation strategy that is proposed is as follows:

- Identify program regions as scheduling candidates
- Model expected performance, *i.e.* determine cpuBusy, memBusy and bothBusy times, and also compute slow-down factor δ.
- Generate voltage/frequency scheduling instructions for each scheduling candidate; adjust performance optimization, if necessary.

6 Algorithms Based on Switching Activity

Katkoori, et.al. [18] and Kumar, et. al. [17] present a profile driven approach to high-level synthesis. The synthesis flow of the synthesis system is know as *Profile Driven Synthesis System* (PDSS). The input to the PDSS are a subset of VHDL and constraints in terms of clock period and area. The PDSS generates a constraint-satisfying design with the least amount of estimated switching activity. In this system, using a user-specified input set of vector, the input specification is profiled to collect profile data for various operations and carriers. Using this profile data and the raw switching activity data of all modules in the library, the switching activity for each module sets that satisfy the user constraints is estimated. The module set with minimum estimate that is the design with lowest power consumption is picked up and further synthesized. The goal of profiling is to gather the following data :

- For each node (operation), determine the number of times the node is executed for a given *profiling stimuli*. Profile stimuli are the input vectors. This number is called the *event activity of the operation node*.
- For each edge, determine the number of times the edge is traversed during execution. This number is called the *transaction activity* of the edge.
- For each edge, determine the number of times the value on the edge has changed. This number is called the *event activity of the edge*.

The authors claim that the results obtained are within an accuracy of 10% of the actual switching activity measured at the switch level implementation of th design.

Raghunathan and Jha [19] present a comprehensive low-power datapath synthesis system that performs the various high-level synthesis tasks with an aim of reducing power consumption in the synthesized datapath. The author call the system as **SCALP**. The system considers both supply voltage and switching capacitance to reduce the power consumption. The authors claim that SCALP estimates switching capacitance accurately. SCALP can handles diverse module libraries and utilize complex scheduling constructs such as multicycling, chaining, and structural pipelining. The input to the SCALP is a CDFG, the input sampling period, and library of components to be used for datapath implementation. The SCALP minimizes power consumption both by V_{dd} scaling and switching capacitance reduction. This is done by first pruning the set of candidate supply voltages to a small set of supply voltages. For each supply voltage in the pruned set, a datapath is synthesized that has minimal capacitance. The best solution among these datapaths in terms of power consumption is then chosen.

7 Module/Register Allocation and Binding for Low Power

In the module selection phase of high-level synthesis functional units are selected to perform each operation in CDFG. Various researchers have addresses the problem of minimizing power power dissipation during module allocation and binding. We will discuss few algorithms here in this section.

Raghunathan and Jha [8] are the first researchers to purpose the allocation method for low power. The method is based on iterative improvement of some initial solution. The authors assume random input in a structurally pipelined design. The method can also handle non-random input sequences. The method is implemented in the framework of *Genesis* behavioral synthesis [23]. In the Genesis behavioral synthesis system, register and module allocations are performed simultaneously, while minimizing the amount of interconnect needed. A lifetime analysis is performed an the scheduled CDFG. Two variables are said to be compatible and can share hardware resources if they never need to be alive at the same time. Similarly, two operations are compatible if they never need to performed at the same time. Allocation is based on a weighted graph called *compatibility graph* (CG). Initially, each variable and operation corresponds to a node in the CG, with undirected edges

connecting compatible pairs. Weights are assigned to edges in the CG to indicate the preference on the two variables or operations for sharing the same resource. A single step of allocation selects the edge in the CG with the highest composite weight, and merges the two nodes it joins, thus maps the corresponding variable (or operation) to the same module (register). If two or more edges have the same composite weight, the tie is broken based in the corresponding transition activity weights (or some cases arbitrarily). Power reduction is achieved by the help of two factors :

- capacitance
- transition activity.

Capacitance is reduced by minimizing the number of:

- functional modules
- registers
- multiplexers.

The allocation scheme selects a sequence of operations (variables) for a module or register such that the transition activity is reduced.

Katkoori, et.al. [16] present an algorithm for register optimization during high-level synthesis. The technique employs a hierarchical optimization phase which exploits the properties of module call graph and information gathered during local carrier life-cycle analysis of each module. The algorithm works in tow phases:

- a local optimization phase groups carriers (all value-storing elements like variables, signals etc.) based on intermodule life-cycle analysis of carriers,
- a hierarchical optimization phases during which further grouping of carriers is considered.

In the first phase (intramodule optimization phase), for each module M_i the life-cycle analysis of all carriers is carried out. Then local compatibility graph LCG(M_i) is built, on which clique partitioning algorithm is invoked to obtain a near-minimal clique cover of the graph. The set of carriers in each clique so formed is called a local compatibility sets of module M determined by clique partitioning algorithm is designed by R_M. The parent-child compatibility sets for each child module are also determined. For any two pair of modules M_a and M_d where M_a is ancestor of M_d and all paths from M_a to M_d in the graph are equal then the graph is said to satisfy *equi-distance* property. A preprocessing stage ensures equi-distance property of the module call graph. Equi-distance property need to be satisfied for hierarchical optimization phase to work, In this phase compatibility sets formed during intramodule optimization are reevaluated and regrouped in order to arrive at a set of globally wellformed sets that can be bound to register. It is not clear from the paper how much power saving they achieve by this register optimization.

8 Some Guidelines for Low Power Synthesis

Rabaey, et. al. [10] propose design guidance for low power using properties of a given algorithms and architectures. They say, an efficient design must involve

a natural match between a particular algorithm and architecture. The properties which are indications for a low power design are measured from the control/data flow graph representation of the algorithm. The properties are :

- Number of Operations
- Critical Path
- Spatial Locality
- Regularity.

Number of operations is directly related to the amount of execution unit power in an implementation. It can be used as a first order estimate of the relative power of the two algorithms. Critical path can be used as a high-level guide of an algorithm's potential for low power implementation. Lower critical paths allow for more opportunity to lower the voltage while still meeting throughput requirements. Spatial locality relates to the degree to which an algorithm has natural isolated clustures of operations with few interconnections between the. Identifications and use of spatial local sub-structure can be used to guide hardware partitioning resulting in the minimization of global busses. This leads to the lower bus capacitance. The idea behind "regularity" is to capture the degree to which common patterns appear in an algorithm. Common patterns enable the design of less complex architecture and therefore simpler interconnected structures like muxes, buffer and buses. Resulted design also has less control hardware.

9 Conclusions

The paper discussed the fundamentals of high-level synthesis in great details. It is pointed out that "dynamic power" dissipation is the major factor that need to be considered for power dissipation. The paper reviewed the work of various researchers who worked on high-level synthesis with special emphasis on power minimization. Different parameters that can be varied are capacitance, voltage, frequency, switching activity and so on. I would like to mention that it is not the power that always need to be minimized, more emphasis should also be given to "energy" minimization; because it is the "energy" that determines the battery life and electricity bill.

Acknowledgment

The author acknowledges Dr.Srinivas Katkoori who allowed him to this survey work as a part of Low Power CMOS VLSI course requirement in the Dept. of CSE, USF, Tampa, USA.

Disclaimers

Some of the figures used in this paper are borrowed from other sources and used here only for academics purpose and the author doesn't claim any originality for the same.

References

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

[2] M.C.McFarland, et. al., "Tutorial on High-Level Synthesis" *Proc. of the 25th ACM/IEEE Design Automation Conference*, 1988, pp.330-336.

[3] P.G.Paulin and J.P.Knight, "Scheduling and Binding Algorithms for High-Level Synthesis" *Proc. of the 26th ACM/IEEE Design Automation Conference*, 1989, pp.1-6.

[4] N.Ranganathan, et. al., "A Linear Array Processor with Dynamic Frequency Clocking for Image Processing Applications", *IEEE Transactions on CSVT*, Vol.8, No.8, Aug. 1998, pp.435-445.

[5] N.Ranganathan, et. al., "Energy Efficient Datapth Synthesis Using Dynamic Frequency Clocking and Multiple Voltages", *Proc. of 12th Intl. Conf. on VLSI Design 1999*, pp.440-445.

[6] N.Ranganathan, et. al., "A VLSI Array Architecture with Dynamic Frequency Clocking", *Proc. of Intl. Conf. on Comp. Design (ICCD)*, 1996, pp.137-140.

[7] J.M.Chang, et. al., "Energy Minimization using Multiple Supply Voltages", *IEEE Transactions on VLSI Systems*, Dec.1997, pp.436-443.

[8] A.Raghunathan and N.K.Jha, "Behavioral Synthesis for Low Power", *Proc. of Intl. Conf. on Comp. Design (ICCD)*, 1994, pp.318-322.

[9] A.P.Chandrakasan and R.W.Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits", *Proc. of the IEEE*, Vol.83, No.4, Apr 1995.

[10] J.Rabaey, et. al., "Design Guidance in the Power Dimension", Invited Paper, ICASSP, May 1995.

[11] Luca Benin, et. al., "A Survey on Design Techniques for System-Level Dynamic Power Management", *IEEE Transactions on VLSI Systems*, Vol.8, No.3, June 2000, pp.299-316.

[12] Daniel Gajski et. al., "High-Level Synthesis: Introduction to Chip & System Design", Kluwer Academic Publishers, 1991.

[13] Raul Composano and Wayne Wolf, "High-Level VLSI Synthesis", Kluwer Academic Publishers, 1991.

[14] Jui-Ming Chang and Massoud Pedram, "Power Optimization and Synthesis at Behavioral and System Levels using Formal Methods", Kluwer Academic Publishers, 1999.

[15] M.C.Johnson and K.Roy, "Optimal Selection of Supply Voltages and Level Conversions During Datapath Scheduling Under Resource Constraints", *Proc. of Intl. Conf. on Computer Design: VLSI in Computers and Processors* (1996), pp.72-77.

[16] S.Katkoori, et. al., "A Hierarchical Register Optimization Algorithm for Behavioral Synthesis", *Proc. of 9th Intl. Conf. on VLSI Design*, Jan 1996, pp.120-132.

[17] N.Kumar, et. al., "Profile-Driven Behavioral Synthesis for Low-Power VLSI Systems", *IEEE Design and Test of Computers* , Fall 1995, pp.70-84.

[18] S.Katkoori, et. al., "A Profile-Driven Approach for Low Power Synthesis", *Proc. of Intl. Conf. on Computer Design* 1995, Austin, October 2-5, pp. 759-765.

[19] A.Raghunathan, et. al., "High-Level Power Analysis and Optimization", Kluwer Academic Publishers, 1998.

[20] K.Roy and S.C.Prasad, "Low-Power CMOS VLSI Ciruit Design", John Wiley and Sons Inc., 2000.

[21] H.J.M.Vendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and its Impact on the Deisgn of Buffer Circuit", *IEEE JSSC*, Vol.Sc-19, No.4, Aug. 1984, pp.468-473.

[22] M.C.Johnson and K.Roy, "Datapath Scheduling with Multiple Supply Voltages and Level Conveters", *Proc. of the IEEE Intl. Symposium on Circuits and Systems*, 1997, Hong Kong.

[23] S.Bhatia and N.K.Jha, "Behavioral Synthesis for Hierarchical Testability of Controller/Datapath Circuit with Conditional Braches", *Proc. of Intl. Conf. on Computer Design*, Oct. 1994.

[24] C.A.Papachristou, et al., "A Multiple Clocking Scheme for Low-Power RTL Design", *IEEE Transactions on VLSI Systems*, Vol.7, no.2, June 1999.

[25] C.H.Hsu, et al., "Compiler-Directed Dynamic Frequency and Voltage Scheduling", *Workshop on Power-Aware Computer Systems, PACS'00*, Cambridge, MA, Nov 2000.

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

[27] Luca Benini, et al., "System-Level Power Optimization : Techniques and Tools", *ACM Transactions on Design Automation of Electronic Systems*, Vol.5, No.2, Apr 2000, pp.115-192.