A Design Methodology for Behavioral-Level Power Exploration

Hartej Singh
Daniel D. Gajski

Technical Report 96-44
March 18, 1998

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA
(714) 824-7063

hsingh@ics.uci.edu
gajski@ics.uci.edu

Abstract

This report describes an integrated design methodology focusing on power exploration at the behavioral level. The design process is iterative and based on designer interaction. Scheduling and power estimation are the two major components in the design process. Power analysis is done in both steps. It is approximate in the first, but detailed in the second step. A combination of empirical and analytical approaches is used for power estimation. Evaluation of power performance trade-offs is done for different implementations for a given behavioral specification.

Contents

List of Figures

1 Introduction

Recent advances in VLSI technology allow fabrication of chips containing over one million transistors. At the same time, several application fields are opening up, for example, multimedia or interactive video applications, that require ever more complex design due to increased functionality and high performance demands. As chip complexity increases, designers need to initiate the design process at higher levels of abstraction rather than gate or logic level. This is because functionality and trade-offs are better understood at behavioral or system levels. In this context, behavioral synthesis is an important element of the design process.

Behavioral synthesis is a process of synthesizing a design from a given behavioral description to a register-transfer level (RTL) description. Behavioral descriptions are usually programs, algorithms, flowcharts, dataflow graphs etc. The RTL structure refers to a set of inter-connected components, along with a netlist. The components may be functional units (ALU's or multipliers), storage units (registers and memories) or interconnection units (muxes or buses).

In general, behavioral synthesis consists of three major tasks: allocation, scheduling and binding. **Allocation** refers to specifying the number and type of resources needed for the design. **Scheduling** is the task of partitioning the behavior into different control steps or states, such that each step is executed in one clock cycle. **Binding** deals with assignment of each variable to a storage unit, each operation to a functional unit and data transfers to interconnect units. These three tasks are closely related to each other.

2 Motivation

Conventionally, behavioral synthesis algorithms have targeted delay and area as optimization metrics. However emerging applications have thrust low-power oriented design methodologies to the fore-front. Due to reasons of portability, battery life, reliability and packaging cost, optimization for power is really important for any reasonably complex design. Power optimization approaches at the behavioral level are significant since research results indicate that higher levels of abstraction have greater potential for power reductions.

The three design quality metrics of power, area and delay, make the design space three-dimensional. These metrics have complex inter-relationships that are design-specific. For a given behavioral specification, there are many implementation choices. Also, different designs have different priorities for the metrics of power, area and delay. A designer needs to select an optimum configuration for some specification, with regard to given constraints. An integrated framework is required for the designer to visualize the overall relationship as well as characterize completely some point in the design space.

For effective exploration, there are three main requirements. These are:

Interactive synthesis: Control of design flow by designer is needed for efficient designs.

Efficient evaluation: To enable complete exploration, each point in design space should be evaluated rapidly.

Characterization of design space: An effective means to represent metric relationships in design space is required. This is because these relationships are usually quite complex, and sometimes the larger perspective is lost.

Next, a design methodology is proposed that provides an integrated approach to behavioral synthesis for low power. It incorporates the above features as well as optimization techniques for power reductions. Different trade-off options can be explored by the designer.

3 Related work

There has been a lot of research in the area of power estimation and optimization for power. An overview of power estimation techniques in VLSI circuits is given by Najm in [6]. Traditionally, circuit level simulators such as SPICE have been used for power estimation. Since transistor level simulation is too slow for use with large designs, logic level simulation was proposed (Irsim). Even this does not allow fast exploration. An architectural power estimator based on profiling and state trace is described in [4]. Landman [7] proposes a modeling approach for module characterization. Mehra and Rabaey [8] address the problem of estimating power from a behavioral description using a combination of analytical and stochastic techniques. A profile driven approach is presented for low power behavioral synthesis by Kumar in [9]. A tool framework for low power behavioral synthesis was developed, using scheduling and allocation based on compatibility graph [10,11].

However, other than estimation, most of these approaches do not accomodate design space exploration. Even though an integrated CAD environment for low power design was proposed in [12], it has certain limitations. It is vague with respect to actual behavioral synthesis of the designs, does not specify the level of interaction or automation, even though it does employ a modeling approach at different levels of abstraction.

The contribution of this report is a concise and effective methodology for low power oriented exploration of design space at behavioral level.

4 Proposed methodology

The target is to enable the designer to select an optimal low power design, starting from a given behavioral specification. As illustrated in Fig.1, there are two important steps in the design process. An architecture implementation is chosen in the first step. For the second step, the chosen architecture implementation is evaluated for power dissipation vs. delay trade-offs.

The first step uses allocation techniques. A schedule is prepared for the design. An approximate evaluation of the scheduled design is performed for power and performance. The user constraints on power, delay and area may not be satisfied the first time, hence this step may iterate several times.

In the second step, the motivation is exploration of design space for power. So performance constraints are varied through a range of values and corresponding power dissipation estimated for specified allocation and schedule. In contrast to the first step, the power estimation method used is comprehensive and accurate. This step repeats each time the performance contraint is violated.

When the user constraints are flexible, the designer may choose to go in for further exploration. The exploration is enabled by repeating the design process after modifying the constraint values. Thus the design process is essentially iterative in nature, to accommodate constraint variations and enable complete exploration. A brief outline of the main steps involved is given below.

4.1 Architecture selection

Allocation and scheduling, followed by binding are performed for the CDFG representation of the specification. The designer has two options for scheduling: resource-constrained or time-constrained. For either approach, a schedule and allocation are specified using applicable algorithms. Next, a quick analysis of this schedule is performed for power dissipation, area and delay. Since this analysis has to be rapid, it is neither very detailed nor comprehensive, but aims to have a quick approximate value for the metrics. A refinement step follows the analysis, which tries to modify the allocation or schedule to minimize the constraint violations. Several iterations may be needed before reaching a satisfactory architectural implementation.

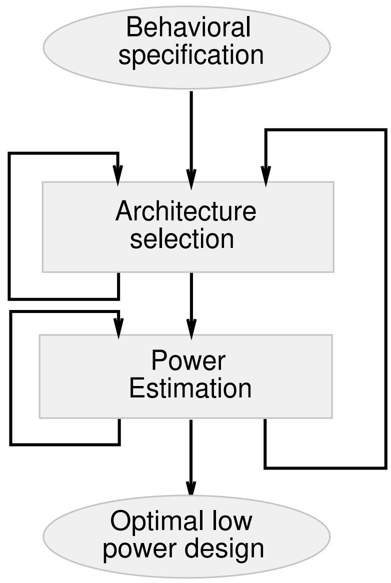


Figure 1: Proposed methodology

4.2 Power estimation

For a given circuit the power dissipation is computed as

$$P = 0.5 * V_{dd}^2 * C * f \quad (1)$$

where V_{dd} is supply voltage, C is the average switched capacitance and f is the switching frequency.

The basic steps involved in power estimation are: (a) estimation of switching activity, (b) relating the switching activity to switched capacitance and (c) assigning the supply voltage for the specified execution time.

(a) The switching activity is estimated based on number of accesses for each functional and storage unit. The accesses can be determined from the profile and state trace information related to the CDFG.

(b) The average switched capacitance per access for each unit is known from the component library. Combining this information with the total accesses for each unit, the switched capacitance is calculated. The cpacitance switched in the interconnect can be approximated through an appropriate model.

(c) For a specific execution time, delay vs. supply voltage curves for the functional units are used to generate a minimum value of the supply voltage. The execution time value is varied over a range and corresponding supply voltage values selected. This iteration forms the basis for analysis of power-performance trade-offs.

5 Design methodology

This section discusses the detailed features of the design methodology. The two basic steps of scheduling and power evaluation, are composed of many sub-steps, with several options available to the designer. The interactive nature, fast design loop and representation of trade-offs will be illustrated in the following descriptions.

5.1 Scheduling and allocation

This step deals with obtaining an architectural implementation from the behavioral specification that will be optimized for low power dissipation under specified constraints. An allocation and schedule are specified for the CDFG using either time-constrained or resource- constrained scheduling. This is subjected to a quick analysis for power, area and delay. If the constraints on these metrics are violated, the allocation and schedule is modified and again analyzed. Since the scheduling, allocation and refinement algorithms differ for the resource-constrained approach as compared to time-constrained approach, they are discussed in separate sections. Figure 2 illustrates the steps involved for both approaches.

5.1.1 Resource constrained scheduling

In Fig. 2, the boxes 1 through 4 depict the sequence of steps involved. The tasks are to select an allocation, generate a schedule, perform an analysis of the schedule followed by allocation refinement to satisfy constraints. Each time the allocation is modified, the schedule is generated again and analyzed. When the constraints are satisfied, this step terminates.

An initial allocation for the CDFG is specified by the designer or optionally by a heuristical analysis of the CDFG (box 1).

5.1.1.1 Schedule generation Once the resources for the CDFG have been allocated, it is scheduled by some algorithm such as list scheduling (box 2). In fact, any other algorithm of the resource-constrained scheduling category might be used here. The goal is that schedule generation should be efficient.

5.1.1.2 Analysis A quick analysis of the scheduled CDFG is then performed (box 3). The objective is to get approximate estimates for power dissipation, chip area, performance and values for minimum state slack for each functional unit. The difficult part is to estimate the power dissipation, the other metrics are easily computed.

Power analysis needs to take into account the number of register and memory accesses, number of F.U. operations, number of bus loads and bitwidths of resources used. This information is combined with power characterization models for resources to estimate power dissipation. At this step, models of interconnect as well as controller for power dissipation may not be used. Relative accuracy rather than absolute accuracy is more important here, since a detailed evaluation is again done in power estimation step. Approximate models may be used to facilitate quick analysis.

The concluding part of analysis compares values obtained for average power dissipation, chip area and performance with initial constraints. For each metric, if the value is within some margin of constraint, it is assumed to satisfy it. In case the value is sufficiently larger, this is a violation of constraint. But if the value is much smaller than constraint, there is slack in constraint.

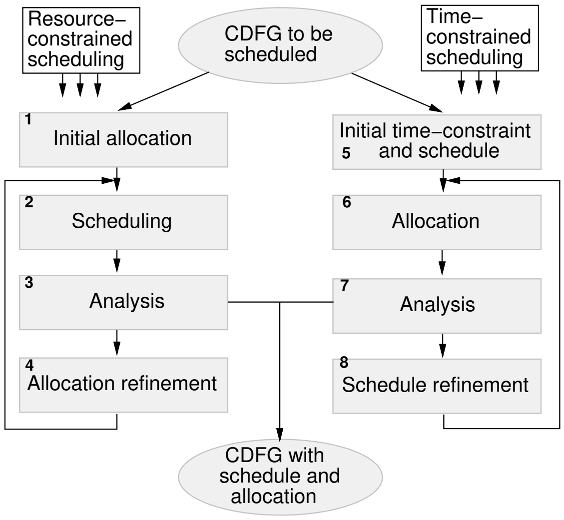


Figure 2: Architecture specification

5.1.1.3 Allocation refinement The task here is to reduce the constraint violations or slack detected in the analysis step by making refinement changes to the allocation, still aiming to have minimum power (box 4).

There are two types of refinement changes: optimizations and trade-offs. Optimization changes are those which have no associated cost in terms of some other metric. Trade-off changes imply an improvement for some metric is made at the cost of another metric. For example, power may be reduced by using smaller units, however smaller units generally will have larger delay. In case the available slack allows use of the smaller unit, the performance of the schedule will not be affected. This is termed as optimization. But if there is no slack, using the smaller unit will lower the performance. In the design flow, optimization approaches precede trade-off strategies, since they have no associated cost. However, refinement allows designer interaction for making decisions at several points.

Refinement changes will be invariably accompanied by change in constraint violation or slack. So, an important aspect of allocation refinement step is keeping track of modifications to constraint violations or slack by the changes.This will enable in terminating the refinement process once the constraint violations have been reduced to permissible levels. The task of evaluating effect of refinement on violations/slack is simple in the case of area related changes. However in case of power violation, information about the unit accesses is required. Again, for performance violations, it may sometimes be difficult to reflect refinement changes in terms of modification to constraint, especially since the schedule may also change.

The following descriptions illustrate paradigms for allocation refinement for the case of violations in constraints. Similar paradigms, although not discussed here, would be applicable for the case of slack in constraints. At the completion of these steps, it is desirable to reschedule and analyze for constraint violations.

Power constraint: Violations in power constraint imply that power hungry units need to be replaced. The optimization approach tries to replace functional units with lower power units, such that available state slack is not violated. Once the optimization potential is finished, and constraint is still not satisfied, trade-off strategies are followed.

For trade-offs, unit power per access per bit, frequency of accesses, bitwidth and unit multiplicity is used to create a list of units sorted by power dissipation. The units responsible for greater power dissipation are replaced by lower power (slower units) versions, if available in the resource library. Also, heavily utilized units are replicated so that number of states in the schedule may be reduced.

Performance constraint: The violations for performance can be reduced by substituting slower units with faster ones or replicating heavily utilized units. Every change may be associated with higher area or higher power costs. It may be important not make unbalanced trade-offs for power. For example if the performance gain offered by a substitution is marginal but the power dissipation is greatly increased, the trade-off is not viable.

Area constraint: For violations of the constraint for area, the need is to use smaller units or decrease the multiplicity of units. In case optimizations are possible, they are carried out. This would refer to making substitutions of larger units by smaller ones within the available slack, considering smaller units will generally be slower. For trade-offs, such substitutions are carried out, even if available slack is violated. However in the case of reducing the multiplicity of units, unit utilization would be the criterion. An instance of an unit would be removed only if utilization is below some value.

5.1.1.4 SRA example Next, a small example is considered to illustrate the above steps. This example is designed to compute the square-root approximation (SRA) of two signed integers, a and b, by the following formula:

$$\sqrt{(a*a+b*b)} = max((0.875x+0.5y), x) \quad (2)$$

where

$$x = max(|a|, |b|), y = min(|a|, |b|) \quad (3)$$

A flowchart description for design steps of scheduling, analysis and allocation refinement is provided in Fig. 3. The specification details are provided in Appendix A and component library is described in Appendix C. For purposes of analysis, functional unit delay is assumed to be linearly related to supply voltage.

The CDFG for SRA is scheduled using ASAP algorithm, to find the length of critical path, which is six states. The initial allocation is fixed, based on types of operations (box 1). The CDFG is scheduled for current allocation using list scheduling (box 2). The next step is analysis (box 3). Power is estimated based on frequency of accesses for functional units, number of loads for registers and buses and average switched capacitance per access. Power dissipated is 7.0 mW @ 5V supply voltage. The minimum clock period is 18 ns and number of states is seven. Area cost for functional units is approx. 800 transistors.

For this example, user constraints specify minimization of power, even if area is lost. So after analysis, in the refinement step (box 4), there are

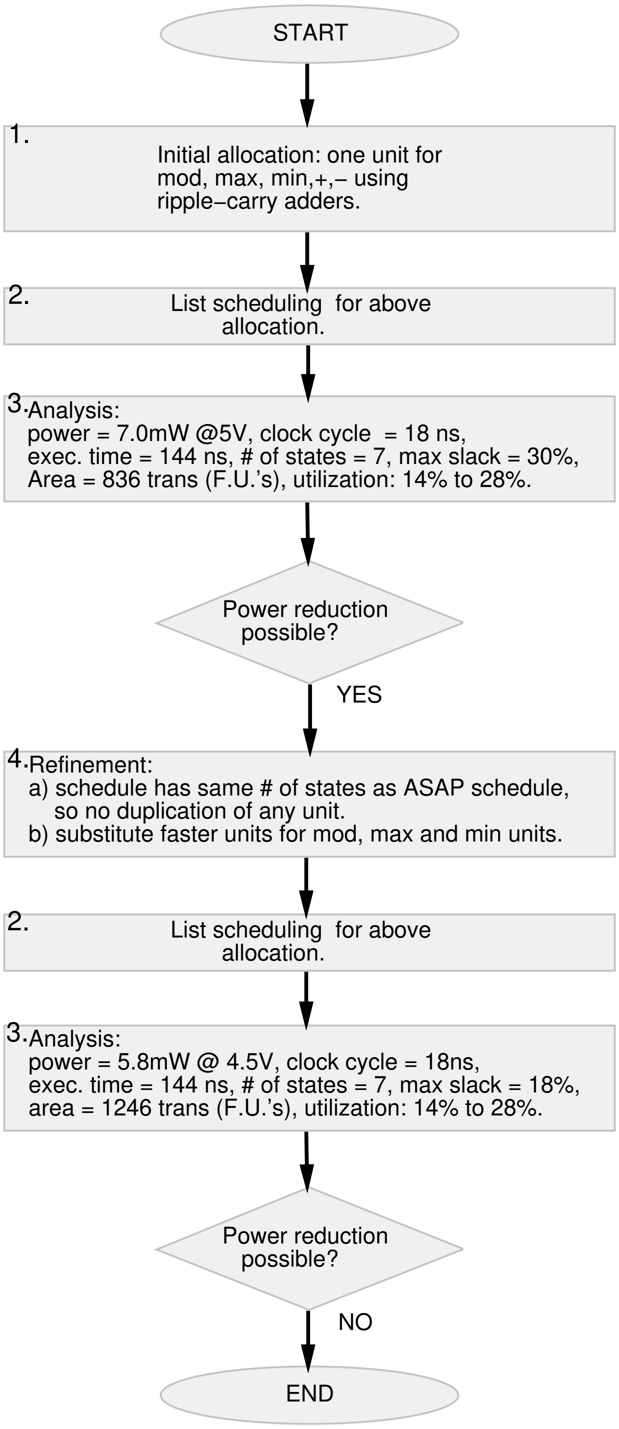


Figure 3: SRA example

two considerations. One is to check if any unit multiplicity can be useful. The other is to use faster components for higher delay operations. This means that if the units with greater delay are replaced by faster units, the state delay will decrease. Or, for the same performance, a lower supply voltage may be assigned for lower power, even though faster units have higher power dissipation than slower units.

Since present schedule has only one state more than ASAP schedule, therefore any multiplicity will not be very useful. But for operations with large delays (mod, max and min), the library has faster versions (using carry look-ahead adders in place of ripple carry adders). So mod, max and min units are replaced by faster versions.

Again, the schedule is prepared (box 2). It is same as the previous one, but analysis (box 3) yields lower power value of 3.6 mW @ 3.6V (decrease of 50% in power) for same clock period of 18 ns. The area cost increases to 1250 transistors for functional units. For given component library, there is no further scope for refinement, hence the scheduling process concludes.

This example demonstrates the sequence and motivation of steps for power optimization.

5.1.2 Time-constrained scheduling

The steps involved in this approach are assigning the time constraint, generating a schedule, specifying allocation for the generated schedule, analyzing the scheduled CDFG and carrying out schedule refinements for meeting constraints. These are depicted in boxes 5 through 8 of Fig. 2.

The time constraint is specified by the designer in terms of number of states and minimum cycle time. Next, a schedule is generated by using heuristics such as the force-directed scheduling algorithm (box 5). Once the schedule is ready, the allocation is performed, so as to satisfy the schedule.

5.1.2.1 Allocation The resource library is referred for specifying the allocation (box 6). If a priority is given to the constraints, automatic allocation may be good enough. Else the designer is able to influence the choice of allocated units with regard to constraints. For example there might be different multiplier versions in the resource library. If power constraint has a higher priority than performance, a low power but slower version may be chosen. Similarly, the designer would be able to make changes to allocation to account for exceptions to given constraint priority.

5.1.2.2 Analysis This step (box 7) is identical to quick analysis step described for resource-constrained scheduling case. The same considerations and descriptions apply here, too.

5.1.2.3 Schedule refinement The step (box 8) is analogous to allocation refinement step for resource-constrained scheduling. The objective is same: to reduce constraint violations/slack by making appropriate changes. However, the difference is that changes are made directly to the schedule by varying the number of states. The possible modifications for a schedule are much limited than for allocation. The number of states constraint can either be increased or decreased. This makes schedule refinement much more simpler in concept as well as implementation, as compared to allocation refinement. However, the effects are related to actual changes in more complex fashion. The granularity of modification effects may be larger. It may also be difficult to anticipate accurately the effect of schedule changes on area and power constraints.

An important point to note is that designer input for schedule refinement decisions is more relevant and necessary, for each change may have unpredictable effects. This is different than for allocation refinement, where automation can produce good results. The designer's experience and intuition can be used here to control the optimizations and trade-offs.

Power constraint: In case of violation of power constraint, the number of states needs to be reduced, so that for given execution time, the state delay may be increased. This in turn, would enable a lower supply voltage for the units, lowering the power dissipation.

Performance constraint: Similar to above, the two options available are to change the constraint on number of states or vary the minimum cycle time value. For violations of performance requirements, the number of states needs to be decreased and/or the cycle time should be reduced.

Area constraint: If the area constraint is violated, then the number of states needs to be increased. This may result in lower multiple operations of same type, and reduce the multiple instances of units. Alternatively, the minimum cycle time may be increased so that smaller units can substitute for larger ones. The opposite approach can be taken if there is slack in the area constraint.

5.1.2.4 HAL example The HAL second order differential equation solver implmentation is discussed to illustrate this approach. The specification as well as DFG and component library for this example are given in Appendix B and C. The functional unit delay is assumed to be linearly related to supply voltage. Fig. 4 shows the iterative steps of scheduling, analysis and refinement.

The first step is to find the ASAP schedule, which in this case has four states. Next, an initial constraint of seven states is specified (box 5). Using force-directed scheduling, the schedule is specified for minimum allocation (box 6) which, in this case, is one unit each of a multiplier, adder, subtractor and comparator. The schedule is analyzed (box 7) in same manner as the SRA example, giving power dissipation as 19.1 mW @ 5.0V supply voltage. Clock period is 56 ns and area for functional units is 3900 transistors.

For refinement (box 8), the aim is to reduce power. This implies the number of states be reduced. So the constraint is set at six states. Again the schedule and corresponding allocation are analyzed (box 6). The schedule now requires two multipliers as compared to one for earlier schedule. Power dissipation for same execution time (392 ns) is 18.0 mW @ 4.5V, however area

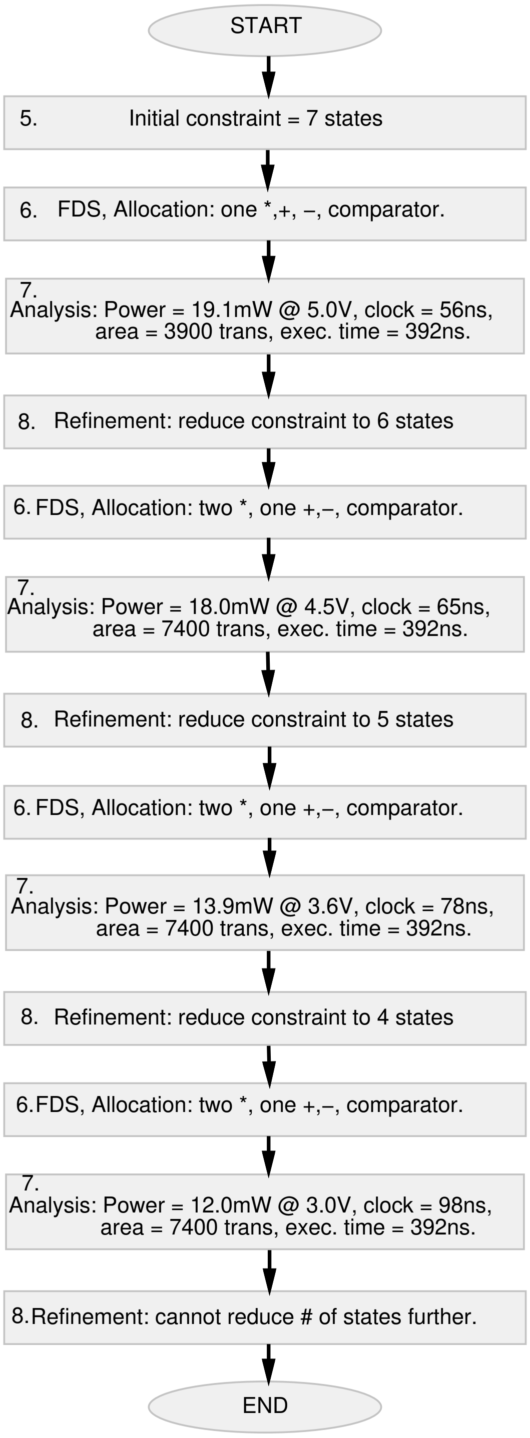


Figure 4: HAL example

for functional units increases to 7400 transistors (box 7).

Next, the constraint is reduced to five states (box 8). Allocation remains the same although schedule changes (box 6). Power dissipation for same execution time is 13.8 mW @ 3.6V and there is no increase in area (box 7). When the constraint is further reduced to four states (box 8), power decreases to 12.0 mW @ 3.0V for same allocation (box 6 & 7) . Since ASAP schedule also has four states, optimization and refinement process concludes (box 8).

For this example, final power decreased by more than 40% from the initial value. Area required is less than area for ASAP allocation, while performance is same.

5.2 Power estimation and trade-off

The power estimation is done at the architectural (RTL) level. At this level of abstraction the gate, circuit and layout level details have not been specified. A floorplan is also not available, making analysis of interconnect and clock distribution networks difficult. Therefore, some approximation is inherent in the estimation.

Estimation methods may be analytical or empirical. Analytical techniques relate the power dissipation to fundamental quantities that describe the physical capacitance and inherent switching activity of a design [3]. But since these methods use very little information as input, power predictions based on them have little relation to actual hardware. On the other hand, empirical methods employ a macromodeling approach. The library of resources is characterized for power dissipation, which implies that the power dissipations of resources for different supply voltages, under some assumptions for input data, are pre-computed. There may be multiple components having same functionality but different power, delay and area costs. Designs that have components from the library can be modeled for power using the characterizations.Usually these empirical models have a strong link to actual implementations. For the proposed methodology, a combination of analytical and empirical approaches is adopted based on Zhu [4] and Agrawal [5].

An important aspect of power dissipation is that it is proportional to the square of the supply voltage. This implies it is desirable to use minimum value of supply voltage. But functional unit delay is a function of supply voltage. Thus if available delay changes, the minimum supply voltage required is different. This relationship is the basis of power performance trade-offs. By incrementally varying the execution time and selecting corresponding minimum supply voltage, the trade-offs are evaluated. Figure 5 depicts the features of the power estimation and trade-off process.

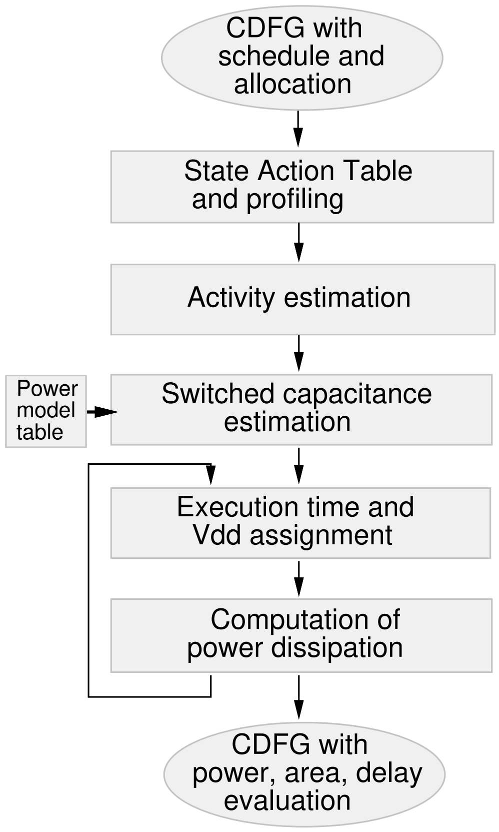


Figure 5: Power estimation

5.2.1 Power dissipation

As stated above, in order to estimate power, a combination of empirical and analytical approaches is taken. A dynamic activity analysis of the architecture is done for typical input vectors. This is performed by profiling the CDFG and generating a state trace [4]. Capacitance models of the resources are used to find the switched capacitance for the entire schedule and allocation.

Once the switched capacitance is known, the other variables are the supply voltage and frequency of operation. If the execution time is fixed, the frequency or clock period can be found from number of states in the schedule. This value of clock period is used to find the minimum value of supply voltage. Then power dissipation can be calculated.

5.2.1.1 Switched capacitance estimation

The library resources are characterized for power using white noise inputs [5]. For example, the capacitance for accessing a register of **N** bits is given. From activity analysis, the number of times a register is accessed is known. Thus the total capacitance switched for different register accesses can be easily calculated. Similarly for functional units, memories, etc. the equivalent switched capacitance is computed.

The components of architecture are divided into datapath, controller, memory and interconnect.

Datapath and controller: The activity vector technique [4] coupled with look-up for capacitance values for the datapath resources [5] generates values that are accurate enough. The controller is divided into four parts namely the state register, decoder, next state logic output logic. The approach followed by Zhu [4] has been shown to be robust enough to model the controllers for different examples.

Memory: The capacitance switched is related to number of memory accesses, word size, address space and specific memory capacitance.

Interconnect: The capacitance switched can be based on empirical models, or similar approximate analysis. For example a derivative of Rent's rule may be used.

5.2.1.2 Supply voltage assignment

As discussed above, the target is to use a minimum supply voltage. The supply voltage vs. delay curves for the functional units are referred to decide the minimum value possible for the specified execution time. Alternatively, for specified set of supply voltages, the execution time is computed. Units with large delays and units on the critical path constrain the value to be chosen. The assumption is that all units are powered using a single supply voltage.

5.2.1.3 ELL example

We discuss the fifth order elliptic wave filter (ELL) example to illustrate the power estimation and trade-off procedure. The allocation and schedule has been specified, there are 20 states in the schedule with 2 multipliers and two adders.

From profile and state action table, activity vector technique [4] is used to estimate the switching activity. Using power characterization models from [5], switching activity is translated to actual switched capacitance. For datapath, this value is 1046.9 pF. For controller, it is 336.8 pF.

Next, we compute execution time for supply voltages of 1.2, 1.5, 2.4, 3.3 and 5.0 volts, given that clock period is equal to multiplier delay. Multiplier delays are figured by scaling normalized delay-supply voltage curve (refer Appendix C, second table). The resulting delays or performance (in terms of clock period, number of states is 20 in all cases) values are used to compute the power dissipation. The power-performance curve depicts this relationship. The different relationships are represented in Figure 6.

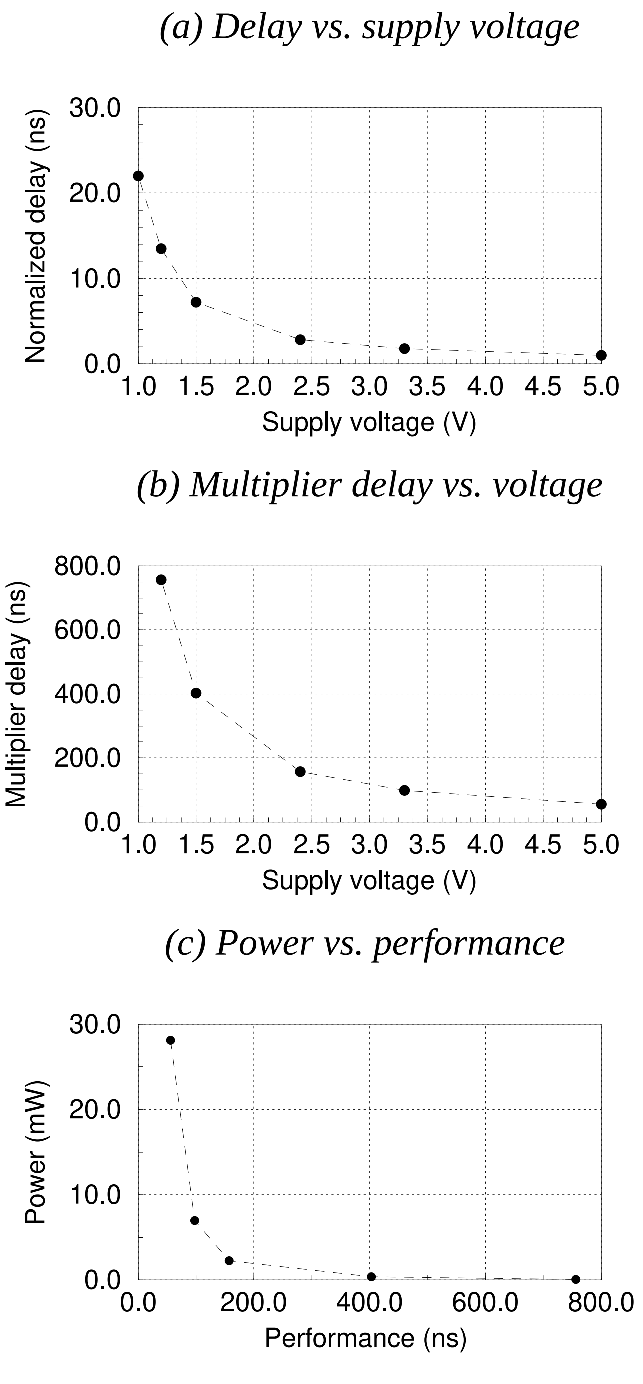


Figure 6: ELL example

5.2.2 Trade-off visualization

The basis of trade-offs for power is to vary the execution time. Larger the execution time, lower the supply voltage, hence lower the power dissipation. So execution time is iterated over a range of search space and corresponding power values are computed.

Visualization of trade-offs is important for designer to comprehend metric relationships. A series of plots for power vs. performance similar to the plot for ELL example (Figure 6), is prepared. Each curve on these plots has a specific area cost associated with it. So designer can choose the optimal design implementation subject to his constraints.

5.3 Design review

This is the point where designer evaluates the information available to him regarding the design space. If more exploration is required, then the design steps of scheduling and power estimation are repeated. Else the design exploration is concluded.

6 Conclusions

In this report, a novel methodology has been proposed to enable behavioral exploration for low power designs. The examples illustrate the benefit of this approach for designers since it incorporates multi-level power estimation, an interactive nature that allows controlled exploration and rapidity of design flow that encourages its applicability to complex designs. For future work , we aim to work on larger designs and implement the methodology in a CAD environment.

From the discussion in the earlier sections, a few conceptual and intuitive observations follow:

Interactive environment: Designer input and control of design flow is important since complex trade-offs are involved in power optimization.

Quick analysis: Design iteration has to be fast and efficient in order to explore larger design space.

Resource characterization: In general, better resource characterization for power leads to better power optimization.

Visualization: It is necessary for the designer to have clear perception of design space relationships and trade-offs for informed decision-making.

7 References

[1] D. Gajski, A. Wu, N. Dutt, S. Lin, *High Level Synthesis*, Massachussetts, Kluwer Academic Publishers, 1994

[2] Daniel D. Gajski, *Principles of Digital Design*, New Jersey, Prentice Hall, 1995

[3] Paul Landman, *High Level Power Estimation*, Intn'l Symposium on Low Power Electronics and Design, 1996.

[4] J. Zhu, P. Agrawal, D. Gajski, *RT Level Power Analysis*, Technical Report, Deptt. of Information and Computer Science, Univ. of California, Irvine.

[5] P. Agrawal, *PRESTO: A gate level power analyzer*,MS thesis, Deptt. of Electrical & Computer Engineering, Univ. of California, Irvine.

[6] Farid N. Najm, *A Survey of Power Estimation Techniques in VLSI Circuits*, IEEE Trans. on VLSI, Dec. 1994

[7] P. Landman, J. Rabaey, *Power Estimation for High Level Synthesis*, Proc. EDAC-EuroASIC, Feb. 1993

[8] R. Mehra, J. Rabaey, *Behavioral Level Power Estimation and Exploration*, Proc. First Int'l Workshop on Low Power Design', April 1994

[9] N. Kumar *et al*, *Profile-Driven Behavioral Synthesis for Low Power VLSI Systems*, IEEE Design and Test of Computers

[10] A. Raghunathan, N. K. Jha, *Behavioral Synthesis for Low Power*, Proc. ICCD, 1994

[11] S. Sokolov, A. Raghunathan, *VHDL Behavioral Synthesis Tool for Low Power Based on FRITS*, final report, SBIR Phase I, by ASC and Princeton Univ.

[12] P. Landman, R. Mehra, J. Rabaey, *An Integrated CAD Environment for Low Power Design*, IEEE Design and Test of Computers, 1995

[13] J. Chang, M. Pedram, *Energy minimization using Multiple Supply Voltages*,International Symposium on Low Power Electronics and Design, 1996

8 Appendix A

The appendices provide the specification and dataflow graphs for the square root approximation (Appendix A) and the HAL differential equation solver examples (Appendix B). A table giving the component characteristics for power area and delay is also given in Appendix C.

SRA example: The square-root approximation of two signed integers, a and b, is computed using the following formula:

$$\sqrt{(a * a + b * b)} = max((0.875x + 0.5y), x) \quad (4)$$

where

$$x = max(|a|, |b|), y = min(|a|, |b|) \quad (5)$$

A flowchart illustrating the computation is provided. The functional units used are absolute unit, max, min, adder and subtractor. The shifts are accomplished through appropriate connections. The two integers are read from two input ports and the result is written to an output port.

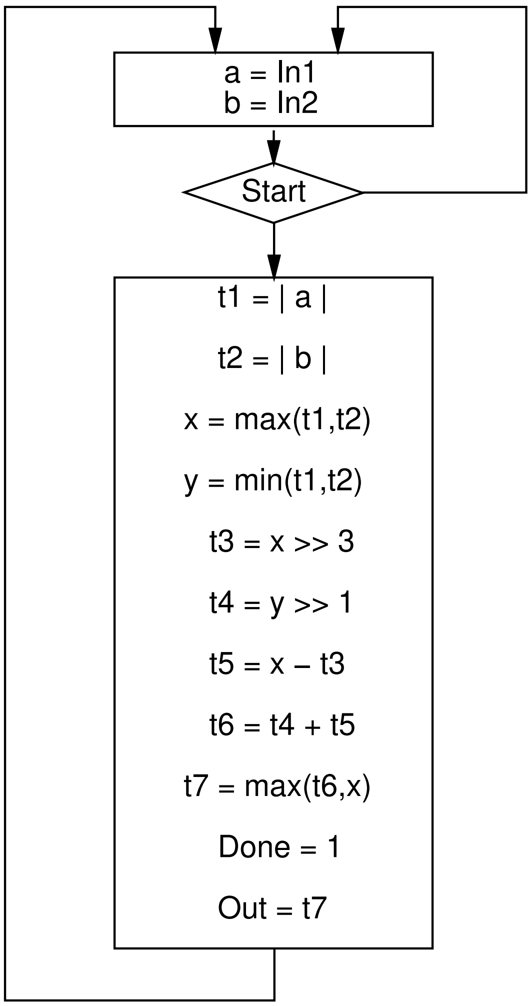
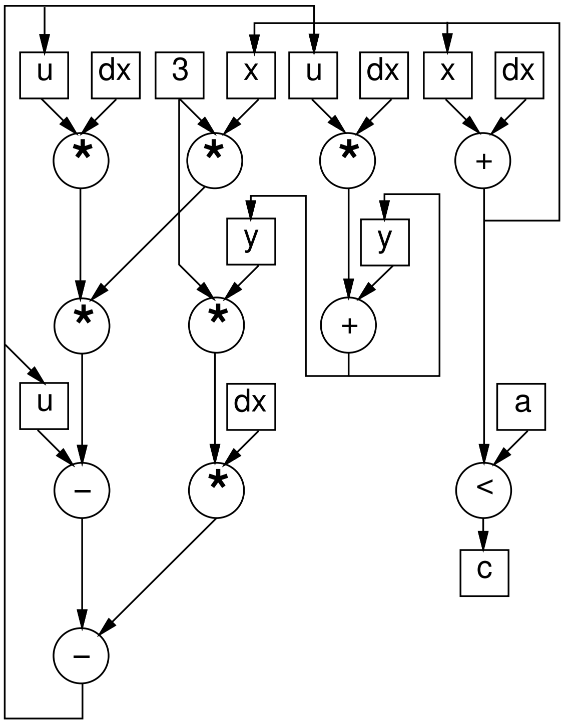


Figure 7: Square root algorithm flowchart

9 Appendix B

HAL example: The DFG as well as the textual description of the HAL example is given in Fig. 8.




```
while   ( x<a )   do
    x1 := x + dx ;
    u1 := u − ( 3 * x * u * dx ) − ( 3 * y * dx ) ;
    y1 := y + ( u * dx ) ;
    x   := x1 ; u := u1 ; y := y1 ;
endwhile
```

Figure 8: HAL: DFG and textual description

The second order differential equation solver uses multipliers, adders, subtractors and a comparator.

10 Appendix C

Component library: The different components used in the examples have been modeled for power dissipation, delay and area costs. The values for the average switched capacitance per access were generated using PRESTO [5]. The components were generated using 0.6 micron, 3.3V VTI standard cell CMOS library. these components are synthesized using only two input NAND and NOR gates and inverters. A tabular form for the values is provided.

Table 1: Component library

Unit	Cap./access (pf)	Delay (ns)	Area (trans)	Bitwidths
mod (ripple carry)	3.82	16.4	188	4
max (" ")	3.82	17.8	196	4
min (" ")	3.82	17.8	196	4
adder (" ")	3.18	12.6	124	4
subtr (" ")	3.36	14.0	132	4
array mult (" ")	59.2	117.4	2656	8
booth mult (" ")	58.6	109.6	2608	8
mod (carry look-ahead)	4.07	11.4	326	4
max (" ")	4.07	12.8	196	4
min (" ")	4.07	12.8	196	4
adder (" ")	3.43	9.0	124	4
subtr (" ")	3.54	10.4	132	4
array mult (skew carry)	71.6	61.6	3213	8
booth mult (" ")	79.4	56.2	3562	8

Table 2: Delay vs. supply voltage *

Voltage (Vdd in volts)	Normalized delay
5.0	1.0
3.3	1.75
2.4	2.8
1.5	7.2

* based on [13]