Available online at www.elixirpublishers.com (Elixir International Journal)

Electrical Engineering

Elixir Elec. Engg. 53 (2012) 12005-12022

ISSN: 2229-712X

Power estimation techniques for embedded and VLSI system: A survey

R.Prabakaran[1], S. Arivazhagan[2] and S.Prabakaran[3]

[1]Department of Electrical and Electronics Engineering, Anna University of Technology, Tiruchirappalli.
[2]Department of Electrical and Electronics Engineering, Mepco Schlenk Engineering College, Sivakasi
[3]Pervaisve Comouting Technology, Anna University of Technology, Tiruchirappalli.

ARTICLE INFO

Article history:
Received: 12 March 2012;
Received in revised form:
7 December 2012;
Accepted: 11 December 2012;

Keywords
Transistor level,
Gate level,
RTL level,
Behavior level,
System level.

ABSTRACT

Advancement in the field of embedded system and VLSI has induced the researcher in designing low power embedded systems and VLSI circuit design. The embedded systems are mostly batteries operated in nature. The power loss during static, dynamic and switching characteristics are tabulated. The switching nature in cmos constitutes a large value of power loss during the switching condition. Many research papers have been proposed in reducing the switching loss, and low power estimation, this paper clearly demonstrates the comparison of them. The main features of the dominated design techniques and methodologies of transistor level, gate level, RTL level, behavior level and system level are reviewed. The corresponding advantages and drawbacks, as well as comparisons between the techniques and the methodologies are also presented. The low-power design process such as transistor level, gate level, RTL level, behavior level and system-level models are explained.

Introduction

Power estimation:

Power estimation is defined as the process of calculating power and energy dissipated with a considerable accuracy at different phases of the design process. Power estimation also refers to the estimation of the average power dissipation of a digital circuit, which is different from worst case instantaneous power estimation, often referred as the voltage drop problem. Chip heating and temperature are related to the average power. Power estimation can be performed with various models of a design: low-level analytical models, C-based architectural models, and structural RTL (Register Transfer Level) models, gate-level models with and without layout data and circuit-level models. Estimating power using each of these models has its own advantages and disadvantages [1].

Need for Low Power Design:

In the early 1970s high speed digital circuit design reduction in area is the main design challenge. Most of the EDA tools were designed specifically to meet these criteria. Power consumption estimation was also a major part of the design process but not very obvious. The reduction of area of digital circuits is simplified today with new IC production techniques capable of fabricating millions of transistors in a single IC. Shrinking sizes of circuits have resulted in reduced power consumption leading to extended battery life. Also in submicron technologies, proper functioning of circuits is limited by the heat generated by power dissipation. The market demands low power devices to have a better lifetime and also should be reliable, portable, should provide better performance, reduced cost and a better time to market. This is true in the field of personal computing devices, wireless communications systems, home entertainment systems, which are becoming very popular now-a-days. High-performance computing devices need to dissipate less power to function for an extended period of time [3]. Keeping these in mind, low power designs have become one of the most important design parameters for VLSI (Very Large Scale Integration) systems.

Design Flow with and without Power:

Figure 1 illustrates a top-down ordinary VLSI design approach. The figure summarizes the flow of steps that are required to follow from a system-level specification to the physical design .The approach was aimed at performance optimization and area minimization. However, introducing the third parameter of power dissipation made the designers to change the flow as shown in the right-hand side of the Figure 1. In each of the design levels, there are two important power factors, namely power optimization and power estimation. Power optimization is defined as the process of obtaining the best design knowing the design constraints without violating design specifications. In order to meet design and required goals, a suitable power optimization technique should be employed. Power estimation is defined as the process of calculating power and energy dissipated with a considerable percentage of accuracy and at different phases of the design process. Power estimation techniques evaluate the effect of various optimizations and design modifications on power at different abstraction levels.

Generally, a design performs a power optimization step followed by a power estimation step, but at a certain design level, there is no specific design procedure. Each design level includes a large collection of low power techniques. Each may result in a significant power dissipation reduction. However, a certain combination of low power techniques may lead to better results than another series of techniques. Generally, power is consumed by the capacitors in the circuits during their charging and discharging due to switching activities. This power dissipation is conserved by shutting down portions of the system when they are not needed which reduces the switching activity.

Tele:
E-mail addresses: prabakaran3787@gmail.com

Large VLSI circuits contain different components like a processor, a functional unit and controllers. The idea of power reduction is to stop the processor components when they are idle so that power dissipation will be reduced when the processor is operating [3].

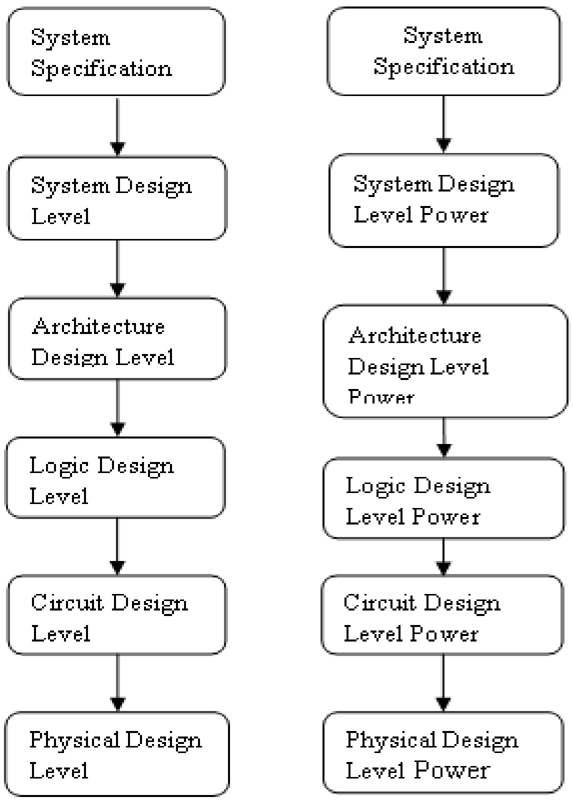


Figure 1 VLSI Design Flow

Relationship between Different Abstraction Levels:

Figure 2. Illustrates the relationship between design abstraction level and power estimation. The power estimation at the higher level is much faster, but the accuracy will be reduced due to the limited design information A number of CAD techniques for power estimation at lower levels of abstraction, such as transistor-level [3] or gate-level, have been proposed.

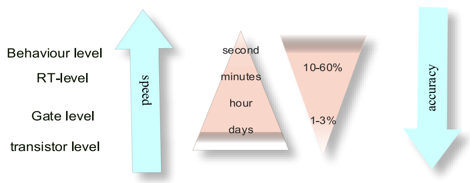


Figure 2 Relationship between different abstraction level & Power estimation techniques

Generally speaking, they can provide more accurate estimation results. However, they may become unpractical for complex designs due to the whole system simulation requires too many computation resources in such as low abstract levels. In addition, when the design has been specified down to gate level or lower, it is too expensive to shift back in fixing high-power problems. Most importantly, IP vendors may not expose such a low-level description to protect their knowledge.

Basic Concepts for Power:

The power dissipation of digital CMOS circuits can be described by

$$P_{avg} = P_{dynamic} + P_{short\text{-}circuit} + P_{leakage} + P_{static}$$

P_{avg} is the average power dissipation; $P_{dynamic}$ is the dynamic power dissipation due to switching of transistors; $P_{short\text{-}circuit}$ is the short-circuit current power dissipation when there is a direct current path from the power supply down to be ground , $P_{leakage}$ is the power dissipation due to leakage currents, P_{static} and is the static power dissipation [3].

Static Power:

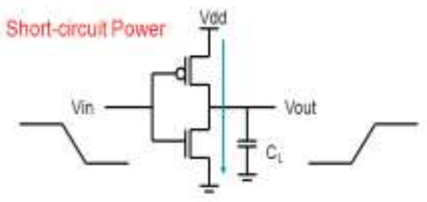
Static power is defined as the power dissipated by a gate when it is inactive or static. Ideally, CMOS (Complementary Metal Oxide Semiconductor) circuits dissipate no static (DC) power since in the steady state; there is no direct path from V_{dd} to be ground. This scenario can never be realized in practice, because in reality, the MOS transistor is not a perfect switch. There will always be leakage currents, sub threshold currents, and substrate injection currents, which dissipate certain amounts of power. The largest percentage of static power results from source-to-drain sub threshold voltage, which is caused by turning of the gate by reducing threshold voltages [3].

Dynamic Power:

Dynamic power (*Pswitching*) is the power consumed during switching events in the core or I/O of an FPGA. Toggle rate is a function of voltage, frequency, and capacitance. The toggles may be in internal logic modules, conducting wire of interconnect, or external package pins [2]. In the deep sub-micron meter process, dynamic power can be reduced with smaller transistors but static power is increasing because a leakage current in smaller transistors becomes bigger. Therefore, the proportion of static power in the overall FPGA power consumption is growing. It is important to understand both power types and their variation under different operating conditions so that they can be properly optimized to meet the power budget. For dynamic power calculation, the essential quantities are a toggle count of transistors and traces, capacitance, and toggling rate. Transistors are used for logic, and programmable interconnects between metal traces in the FPGA. The capacitance consists of a transistor parasitic capacitance and metal interconnect capacitance. In CMOS circuits, dynamic power consumption is related to charging and discharging parasitic capacitances on gates and metal traces, which accounts for the overall power consumption of the chip. According to the way of reducing dynamic power includes the reduction of capacitance, operating voltage, frequency and toggles. [2].

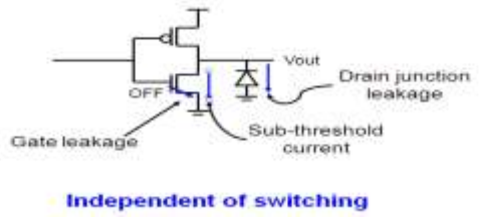
Short-Circuit Power:

The last power component is called static power, because it is not related to the signal transition. This component appears as long as the circuit is powered. Comparing to dynamic power, this component dissipates less total power which makes this component negligible in some designs. Static power highly depends on technology and design. In current technologies, the transistor sizes are reduced and this increases leakage currents in the circuit which increases the amount of static power in the circuit. In order to have a better performance, alternative design methods such as pseudo logic, domino logic, etc can be used that affects the amount of static power.



Above Fig illustrates the pseudo NMOS logic as an example to show the design of a circuit that can affect the static power. In this circuit, the single NMOS transistor is always on since its gate is connected to the V_{DD}. As long as the output is zero, there is a path in the pull dawn network to connect V_{DD} to the ground. This path carries the current from V_{DD} to be ground and cause the static power dissipation. [10][28].

Leakage Power: The PMOS and NMOS transistors in a CMOS logic circuit usually have non-zero reverse leakage and sub-threshold currents. These currents can contribute to the total power dissipation even when the transistors are not carrying out a switching action. The leakage power dissipation, $P_{leakage}$ is caused by two types of leakage currents [3]. Advancement in the field of embedded system and VLSI has forced the researchers in the low power embedded system and VLSI circuit design. Most of the embedded systems are battery operated. The power loss during static, dynamic and switching characteristics are tabulated. The switching nature in cmos incorporates large power loss. Many research papers have been proposed on reduce switching loss and low power estimation. This paper clearly demonstrates the comparison among them. The main features of the dominated design techniques are methodologies of transistor level, gate level, RTL level, behavior level and system level are reviewed. The corresponding advantages and drawbacks, as well as comparisons between the techniques and the methodologies are also presented. The low power design process such as transistor level, gate level, RTL level, behavior level and system level models are explained [7].

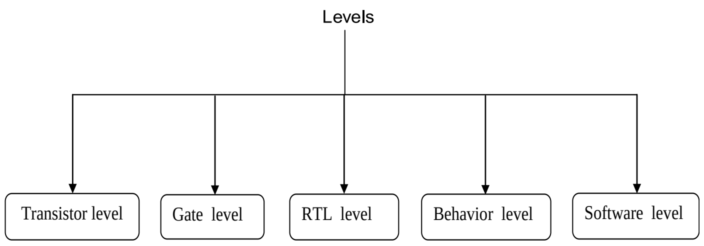


The leakage power dissipation, $P_{leakage}$ is caused by two types of leakage currents

a) Reverse-bias diode leakage current

b) Sub threshold current through a turned-off transistor channel

Low Levels of power estimation: The level of detail in the modeling performed by the power simulator influences both the accuracy of estimation as well as the speed of the simulator. In this section we survey the models frequently used at low level as these power consumption estimation techniques cover a range of abstractions such as the circuit/transistor level, logic gate level, RT level, and architectural level.



Transistor level power estimation:

Introduction:

A transistor-level programmable technology is composed of two key mechanisms. One is that a MOS transistor is divided into parallel connected sub-transistors, so that the transistor possesses various characteristics by switching the parallel connection. The other mechanical changes V_{th} and GM of the transistor by adjusting the bulk potential based on body effect. In the post-layout simulation, the results indicated that our mechanisms could tune the circuit performance such as the gain continuously as well as over a wide range [5][6]. Up to now several logic-level power estimation, techniques have been described. In some cases, these techniques may suffer from the problem of inaccuracy, since they ignore short-circuit currents and the glitching power consumption is strongly dependent on the delay model incorporated. However, the power estimation methods, based on transistor-level power simulation, have been providing more accurate results. Running such a simulator with a given set of simulation vectors, provides efficient estimation of the power dissipated in digital circuits, since glitches and short-circuit currents also take into consideration. While such techniques are more effective than some logic-level estimation techniques, they consume excessive CPU time, hence they are not suitable in large circuits estimation [8]. We call V the original vector set and S and NS the transistor-level simulated and non-simulated vector sets, respectively. Running a logic simulation for each set computes the switching activity E_{sw} for each internal node, providing thus a transition measure Φ, which in fact is the total effective capacitance of the circuit, that is:

$$\Phi = \sum_{\substack{all \\ nodes}} E_{sw} C_L$$

On the other hand, the application of transistor-level simulation to set S results in the accurate calculation of its power dissipation. The next step is to compare this result with the transitional measure found by logic simulation. Therefore, a relationship factor $R\ (S)$ is extracted from the simulation process for set S, which is the ratio of power dissipation of transition measure:

$$R(S) = P(S) / \Phi(S)$$

According to the authors this factor can then be used in order to estimate the power dissipation of the non simulated setting, as follows:

$P_{estimated}\,(NS) = R(S) * \Phi\ (NS)$

until the approximation error becomes minimum (stopping criterion).

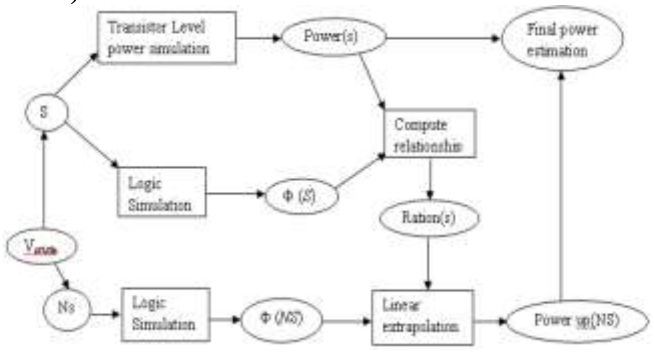


Fig 3. The mixed-level power estimation process

Which is the power estimated over all the original vectors. The whole process, step by step, is shown in Figure 4. After simple algebraic manipulations, this is equivalent to compute

$$P_{estimated}\,(V) = R(S) * \Phi\ (V)$$

Consider the extreme case that power simulation (we mean transistor-level simulation) has been done on the entire set of original vectors, (i.e. S=V), so the estimated power is equal to the total power dissipation of the circuit. But, as we mentioned earlier,this process would require very large CPU time. In our case study, when only a subset of vectors is used for power simulation, a certain degree of error will be introduced by this approximation. The approximation error is:

$$E(V) = |R(V) - R(S)| * \Phi(V)$$

And converges zero very fast as the number of vectors increases. Sometimes it is not possible to predict the number of input vectors that must be simulated in order to achieve a possible accurate result. Therefore, an alternative process which consists of dividing the original set of input vectors into a large number of packets (each packet consists of a fixed number of vectors) and then simulating sequenced these packets,

Transistor Level Simulation:

1. Circuit level
2. Switch level

1. Circuit level: Circuit level simulators are responsible for estimating power. This is done by calculating the average current (I) that flow from the circuit power source and then the average power can be obtained.

SPICE is an example of an accurate circuit level simulator.

$$P = I_{avg}VDD$$

$$I_{avg} = \int I_{dt}/T$$

At this level there are accurate models for circuit devices such as transistors. Capacitances and resistors have values close to the reality so the output of estimation is accurate and testable. Estimation in the case of large circuits is complex and time consuming. The first limitation is that we have to solve complex systems of equations. This is only feasible for fairly small circuits. SPICE solves an equation for each node of the circuit and a growth in the circuit size will dramatically increase the complexity of the problem. The other limitation is the length of the input vectors must be short; otherwise the simulation would be time consuming. R. Marculescu and C. Ababei has proposed a technique to reduce the length of this sequence "by an order of magnitude" without losing accuracy (around %5 lost). This can be done by obtaining a" compact representation" of input vectors with an acceptable approximation [10]. Because of these limitations power estimation at the circuit level with high accuracy using realistic models is only performed for small circuits. A simpler power estimation method is to use simple models for transistors instead of complex ones. Each transistor can be modeled as a switch that is either conducting or not in a switch level simulation. As this method simplifies transistor models, it can estimate larger circuits and use more input vectors. Although the data obtained from the power estimation is not as accurate as the previous method especially in the estimation of static power consumption, dynamic and short circuit Power estimation can be done by these techniques and then the static power should be estimated based on its specific techniques.

2. Switch level:

Switch level simulators like MOSSIM and IRSIM view transistors as bidirectional switches and circuit nodes as charge storage nodes. When a transistor is in an ON state, the switch closes creating a conduction path between the drain and source nodes of a transistor. In this model, simulation can be performed with an approximate RC calculation, thus making it faster than the normal transistor level analysis. Switch level simulators can be extended for power analysis by calculating the approximate switching capacitance for dynamic power estimation. Though other components like leakage and short circuit power can be estimated, these are not very accurate compared to transistor level analysis. For example, short circuit power must be accounted for by examining the time in which the switches form a path from power to ground. A switch level simulator does not accurately model timing. Besides, the modeling does not consider the output load capacitance which leads to further inaccuracies.

Transistor-Level Programmable:

We propose a transistor-level programmable technology in analog circuits by introducing two mechanisms for pseudo sizing of MOS transistors as follows [9].

1) A MOS transistor is divided into two or more parallel connected sub-transistors. The number of active sub-transistors can be switched using the transfer gates.

2) A MOS transistor has an independent well area, and the threshold voltage and gm can be adjusted by changing the bulk potential (i.e. body biasing).

Each mechanism can adjust the transistor characteristic after manufacturing the chip. The former is associated with the tuning of channel-width (W) in a usual MOS analog circuit design. This serves the tuning over a wide range but discrete. Meanwhile, the latter is associated with the tuning of channel-length (L), and the tunable range is narrow but continuous. To check the feasibility of our ideas, we embed these programmable mechanisms into a common-source-amp part of an op amp, and evaluate the programmability of the circuit performance. In the post-layout simulation with considering the parasite, it was shown that our programmable mechanism combining the sub-transistor switching and the body biasing could adjust the gain continuously as well as over the widest range. To the best of our knowledge, this is the first work of transistor-level programmable technology with body biasing.

Transistor Sizing and Body Biasing:

We propose a mechanism for tuning analog circuit performance continuously by adjusting the bulk potential of MOS transistors. This makes use of body effect of MOS transistor. Plus, we are aiming to associate the adjustment of the bulk potential with the channel-length (L) sizing of the MOS transistor, because L sizing is used for optimizing an analog circuit in a usual design [9].

Body Effect:

A MOS transistor is regarded as a device with four terminals; gate, source, drain and bulk. In general, the transistor is used so that the potential between the bulk and the source is the same (i.e. V_{sb}=0V). Plus, the threshold voltage V_{th} of the transistor is dependent on $V_{sb.}$ This dependency is called body effect, and formulated as Eq. (1) [9].

$$V_{th}=V_{tho}+\gamma(\sqrt{|2\Phi_F-V_{sb}|} - \sqrt{|2\Phi_F|}) \text{———} 1$$

Where ΦF is the surface potential of the channel, and γ is the body effect coefficient. The drain-to-source current I_{ds} are dependent on V_{th} as described in Eq. (2).

$$I_{ds}=1/2\ \mu_n C_{ox} W/L\ (V_{gs}-V_{th}) \text{———} 2$$

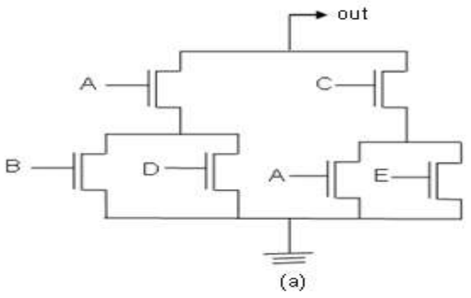
In nmos transistor, as the potential of the bulk is becoming lower than that of the source, Vth is growing up and Ids is decreasing. Furthermore, changing of Ids influences the trans-conductance gm of the transistor as shown in Eq. (3).

$$g_m=\sqrt{2\mu_n C_{ox}(W/L)I_{ds}} \text{———} 3$$

As is well known, gm is an important parameter for describing the circuit of analog circuit. This means that the bulk potential is an available parameter for tuning the analog circuit performance. Methodologies that deal with the transistor-level implementation of a gate are mainly directed towards minimizing the total number of transistors with indirect gains in other performance criteria. That is, the power minimization is implicit in these methodologies. Two main approaches exist: factorization methods and graph-oriented methods. All these methodologies try to minimize the total number of transistors but they do not take into account about certain inputs that are more critical (i.e., Have a higher switching activity) than others. Some rules for transistor-level design styles with respect to low voltage, low power, and power-delay products have been given in [27]. Some other existing transistor-level approaches deal with specific design styles such as pass transistor logic. For achieving low power consumption, a solution should minimize the number of transistors driven by "critical" inputs (the ones with high activity) at the expense perhaps of the number of transistors driven by non-critical inputs. In the proposed methodology we explicitly address this issue by considering the switching activity of each particular input, and obtain a transistor-level implementation that has low power consumption and, secondarily, minimizes the number of transistors required for the implementation [4].

The synthesis technique:

In order to demonstrate the effect of proper transistor level synthesis for CMOS gates the function $F = AB + AC + AD + CE$ is examined. Two implementations for the n-MOS part of function F are shown in Fig5. Assuming that all transistors are the same size then an approximation of the power can be given by the equation $(\zeta_A\alpha_A + \zeta_B\alpha_B + \zeta_C\alpha_C + \zeta_D\alpha_D + \zeta_E\alpha_E) * f * C * V^2$, where ζ_X is the number of transistors driven by input X and αX is the switching activity of input X. With $P = f * C * V^2$, the power for the implementation of Fig. 5a will be $P_1 = (2\alpha_A + \alpha_B + \alpha_C + \alpha_D + \alpha_E) * P$, and for Fig.5b, $P_2 = (\alpha_A + \alpha_B + 2\alpha_C + \alpha_D + \alpha_E) * P$. Different profiles of the input switching activities can determine which implementation has the lowest power consumption. For example, with a profile of switching activities $\alpha_C = 0.5$ and $\alpha_A = \alpha_B = \alpha_D = \alpha_E = 0.1$, $P1 = 1 * P$and $P2 = 1.4 * P$ and thus the design of Fig. 5b has 40%more power consumption than that of Fig. 5a. However, for a profile of $\alpha A = 0.5$ and $\alpha B = \alpha C = \alpha D = \alpha E = 0.1$,$P1 = 1.4 * P$ and $P2 = 1 * P$ and thus now the design of Fig.5a has 40% more power consumption. It is explicitly clear that the selection of an appropriate transistor-level design for a gate with a known switching activity profile at the inputs can result in reduction of the overall power consumption. In this paper a transistor-level synthesis algorithm that takes into account these profiles in order to give a low power design is given. We assume that the vector of the switching activity values for the inputs of the super gate has been given. Namely, if the super gate has n inputs, the switching activities will be denoted by $\alpha_1, \alpha_2 ... \alpha_n$. For our comparative purposes, the quantity that represents the power consumption of a given candidate implementation for the gate is taken to be $S = \sum^{n}_{i=1}\zeta_i\alpha_i$, where ζ_i is the total number of transistors (assumed all to be of the same minimum size) in the candidate implementation driven by input i or its complement. (The power consumption is proportional to S.). The fundamental part of the proposed methodology is, given a partial transistor diagram D (i.e., a diagram implementing a subset of the given product terms) and a product term τ (represented as a set of literals (transistors)) to be placed next, to find the most power-efficient placement of τ in D without creating any escape paths. The placement is done by considering three alternatives ("parallel,""splice," and "bridge") as was done in [4]. This part is described below as procedure PLACE (D, τ). (The terminals of the network are referred to as "VDD" and "GND" for ease of reference (in reality only one of them will be VDD or GND)).



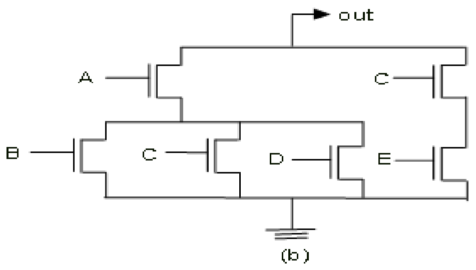


Fig 4: The effect of switching activities on transistor-level implementations

Logic function $F = AB + AC + AD + CE$. With switching activity vector $[\alpha_A = 0.1, \alpha_B = 0.1, \alpha_C = 0.5, \alpha_D = 0.1, \alpha_E = 0.1]$, the network in (a) is better than (b). With switching activity vector $[\alpha_A = 0.5, \alpha_B = 0.1, \alpha_C = 0.1, \alpha_D = 0.1, \alpha_E = 0.1$, the network in (b) is better than (a).

Transistor Folding:

Transistor sizing is essential to produce high performance circuits. Many tools are able to perform individual transistor sizing to optimize timing and power consumption [5]. Layouts produced in the 1D layout style with different sized transistors tend to waste area since the height of each diffusion row is adjusted accordingly to its tallest transistor.

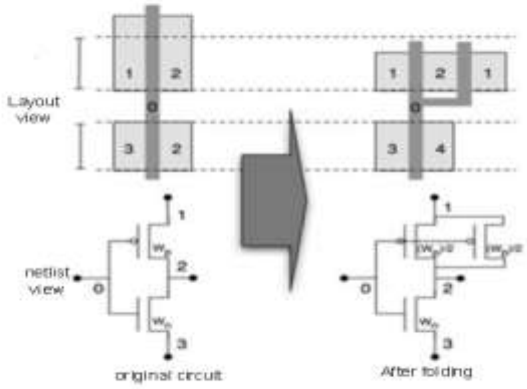


Fig 5. Transistor folding in a large transistor

To solve this problem, one of the most used methods is the transistor folding as illustrated in the Figure 6. It consists of breaking bigger transistors into smaller ones connected in parallel to keep short the cell height, at the expense of a little increase in the cell width. According to [5], the folding problem can be classified as static/dynamic placement with static/dynamic folding. Our approach addresses the dynamic placement with the static folding problem. Given the diffusion row limits, we fold the transistors by directly modifying the cell netlist. Creating new transistors in parallel, before the execution of the placement step. This approach s gives more freedom to the placement algorithm so that it can achieve better results than the folding executed after the placement as in.

Logic-level power estimation:

Introduction:

Since power consumption has become one of the most important concerns in digital VLSI design, with special emphasis on portable applications, designers must follow a variety of power optimization techniques in order to reduce the total cost and improve the performance of such systems. The power management of a particular design adds to a list of problems that VLSI designers and design managers have to contend with. Computer Aided Design (CAD) tools are essential for power management tasks. Specifically CAD tools should be enhanced to estimate power dissipation during the design phase in order to meet the power specifications without a costly redesign process [2][6].

Power estimation techniques:

Power consumption of a gate for each transition is calculated by $E_c= 1/2C_{out}V^2_{DD}$.In this equation, C_{out} can be calculated using “parasitic gate and wire capacitance models” and then obtaining each gate’s switching activity will enable us to estimate the dynamic power . Although switching level techniques are as accurate as circuit level estimation techniques they are faster in the order of magnitude. An approach to obtain power dissipation is to calculate dynamic power by switching level techniques and static power dissipation using aforementioned methods [10][29].The power estimation techniques at the gate level and lower levels of abstraction can be broadly classified into:

1. Simulation based techniques
2. Probabilistic techniques and
3. Statistical techniques

Simulation-Based Techniques:

In the earliest proposed simulation based techniques, the average power is calculated by monitoring both the supply voltage and current waveforms. These are too slow to handle very large circuits. Other simulation based techniques assume that the power supply and ground voltages are constant, estimating only the supply current waveform. Although these are efficient in handling very large circuits, the estimation is strongly dependent on the set of input vectors. Using a logic simulator the design is fed with test vectors to obtain the switching activity at each gate. There are two factors of consideration in this approach: The number of test vectors and the delay model.

1) Number of Test Vectors: It is important to take into consideration, the minimum number of test vectors needed for power estimation. R. Burch and F. N. Najm discussed some suggestions to calculate it. However, a small number of test vectors have been usually enough to estimate the power with a fair level of accuracy [10].

2) Delay Model: If we do not take the gate’s delay into account, a change in the test vector will result in at most one transition per each gate, and all gate’s transitions happen at the same time. This approach is fast and simple but ignores spurious activities of the signals in reality that causes underestimation of the dynamic power estimation. Then on-zero delay models propose that each gate has a delay and propagation of a signal through the circuit takes time. Applying this model will result in power estimation with a higher level of precision.

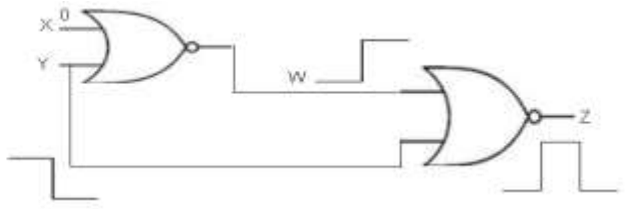


Fig. 6: A combinational circuits with two inputs

In Fig. 6 ’x’ is ’0’ and ’y’ is ’1’ so at the beginning ’w’ is ’0’ and ’z’ is ’0’. Then ’y’ goes from ’1’ to ’0’. The input to ’w’ and ’z’ change so ’z’ and ’w’ go high. Changing the ’w’ has affected on ’z’ and it switches back to 0[9].

Probabilistic Techniques:

In probabilistic techniques, user-supplied input signal probabilities are propagated into the circuit. To achieve this, special models for the components have to be developed and stored in the module library. Cirit first proposed power estimation based on the probabilities. Based on this, probabilistic simulation was proposed which accepts the specification of probability waveforms. It was further enhanced for more accuracy by Stamoulis et al., and Tsui et al. Other probabilistic approaches based on transition density and on Binary Decision Diagrams (BDDs) are proposed. All the above approaches are applicable only for combinational circuits. For sequential circuits various approaches have been proposed which assume that the future of the FSM is dependent only on its present state and independent of its past state. These techniques are based on propagating the statistics of input vectors through the circuit to calculate the internal switching probability [34].

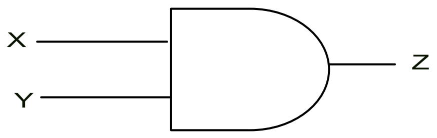


Fig. 7: A 2-input AND gate

Fig. 7 shows an AND gate with an input probability of P^1_x and P^1_y (P^1_x is the probability of high state for x) then for the output probability we have:

$$P^1_x\, P^1_y= P^1_z$$

The three major issues to be considered are Delay model, spatial correlation, and temporal correlation.

1) Delay Model: Using the non-zero model in Fig. 4 , if the delay of gates is Δ1 and Δ2 from left to right, signal z may have a transition at Δ2 and another at ’ Δ1 + Δ2’, the probability of transition of z is obtained by adding these two probabilities .

2) Spatial Correlation: Before calculating a gates probability we have to consider whether it has independent inputs or not, that is not having common inputs. In Fig. 7, $P_x = P_y=0.5$ so to calculate Pw we have

$$\begin{aligned} P_w &= (1- P_x).(1 - P_y) \\ &= 1- P_x - P_y+ P_xP_y \\ &= 1- 0.5 - 0.5 + 0.25 = 0.25 \end{aligned}$$

This type of calculation does not hold for ’z’ because its inputs are not independent. Therefore the following is a wrong calculation.

$$\begin{aligned} P_w &= (1- P_w).(1 -P_y) \\ &= 1- P_w-P_y + P_wP_y \\ &= 1- 0.50.25 + 0.125 = 0.375 \end{aligned}$$

To calculate z’s probability:

$$\begin{aligned} P_z &= (1- P_w).(1 - P_y) = 1 - P_w - P_y + P_wP_y \\ &= 1- [1 -P_x-P_y +P_xP_y] -P_y + P_y[1 - P_x-P_y +P_xP_y] \end{aligned}$$

$= 1 - 1 + P_x + P_y - P_xP_y - P_y + P_y - P_yP_x - P_yP_y + P_xP_yP_y$
$= P_x - P_xP_y + P_y - P_yP_x - P_y + P_xP_y = P_x - P_yP_x$
$= 0.5 - 0.25 = 0.25$

In this equation P_yP_y is the probability of P_y when P_y holds, so the second P_y is equal to '1' and $P_yP_y = P_y$.

Temporal Correlation: As the following truth table depicts that the probability of all three bits is the same and is equal to 0.5. However it can be seen that 'x' has a transition activity of $\alpha_x = P_x^{01} + P_x^{10} = 0.125$ while the activity for 'z' is $\alpha_z = P_z^{01} + P_z = 0.5 + 0.5 = 1$. Transition activity is a major power dissipation factor and should be considered [10].

Figure 8 probabilistic techniques are about the input stream to estimate the internal switching activity of the circuit. These techniques are very efficient, but they cannot accurately capture factors like glitch generation, propagation etc. While in statistical techniques the circuit is simulated under randomly generated input patterns and monitoring the power dissipation using a Simulator. For accurate power estimation, we need to produce a record number of simulated vectors, which is usually high and cause run time problem. To handle this problem, a Monte Carlo simulation technique was superscalar RISC processor based on 64-bit MIPS instruction set. The highest frequency of the chip is 1.0GHz and the power dissipation ranges from 4.0 to 7.0 watts depending on the applications. We use

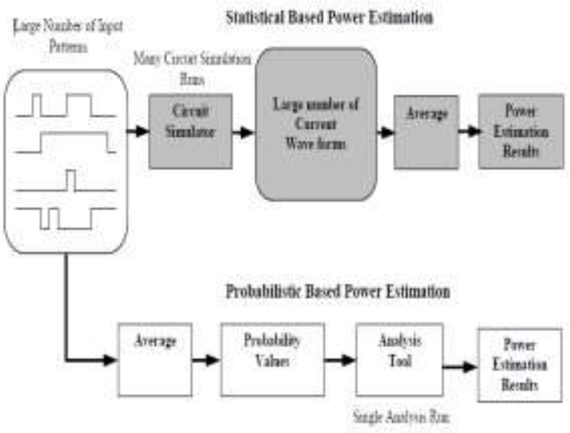

Statistical Based Power Estimation
Large Number of Input Patterns
Many Circuit Simulation Runs
Circuit Simulator
Large number of Current Wave forms
Average
Power Estimation Results
Probabilistic Based Power Estimation
Average
Probability Values
Analysis Tool
Power Estimation Results
Single Analysis Run

Fig 8: An alternative flow for power estimation

Presented in. This technique uses input vectors that are randomly generated and the power sample is computed [30].

Statistical techniques:

As opposed to simulation based techniques, statistical techniques do not require any specialized models for the components. The idea is to simulate the circuit with randomly generated input vectors until power converges to the average power. The convergence is tested by statistical mean estimation techniques.

Gate-level netlist based power analysis:

Figure 10 describes the typical gate-level simulation and power estimation flow. First, the gate level simulation gets design netlist, the Verilog simulation library and test bench, the simulation tool records all switching activities of inputs and outputs as well as all internal states of given circuits. The statistical results of switching activities can be recorded in the form of dynamic preservation, such as VCD (Value Change Dump) file, and also recorded in the form of the average statistics preservation, such as SAIF (Switching Activity Interchange Format) file. The gate level power calculation tool receives these switching activity files, power models for each cell type, wire parasitic file (SPEF), clock constraints, etc. To calculate the actual power consumptions of giving netlist circuit and benchmark. We have applied the flow mentioned above to the gate-level power analysis of a high-performance 64-bit general-purpose processor - Godson 2E [4]. The Godson series microprocessors are the first attempt to design general-purpose microprocessor in China [5]. The latest Godson-2E processor is a four-issue Cadence Nc-verilog to simulate the processor's netlist and Synopspy Prime Power to calculate the power. Figure 10 shows the average power consumption for each circuit type of Godson-2E processor [15].

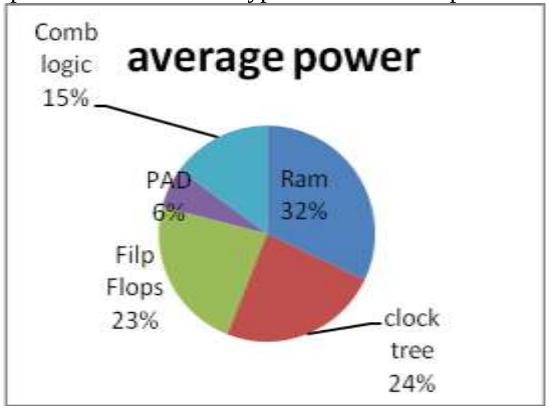


Fig 9. Average Power for Each Circuit Type of Godson-2E processor

Proposed speedup method for gate level power Simulation:

From the flow described above, the gate-level netlist based power analysis method focuses on accurate records of switching activities in all cells of a given circuit. However large records are needed for large circuits which has a large amount of cells when program is running dynamically. It slows down the simulation speed. This paper presents a methodology to accelerate the gate-level simulation and power estimation [15].Different from all previous gate-level power simulation methods, we utilize static probability propagation scheme and apply it in conventional

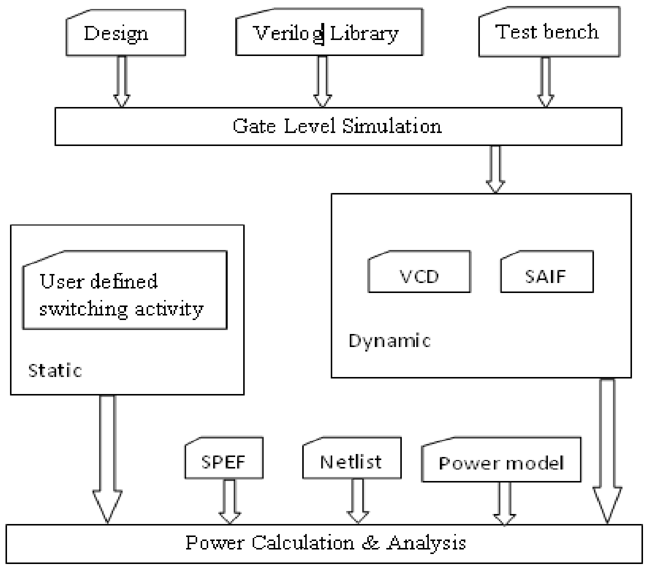


Figure 10. Gate-level Simulation and Power Analysis Flow

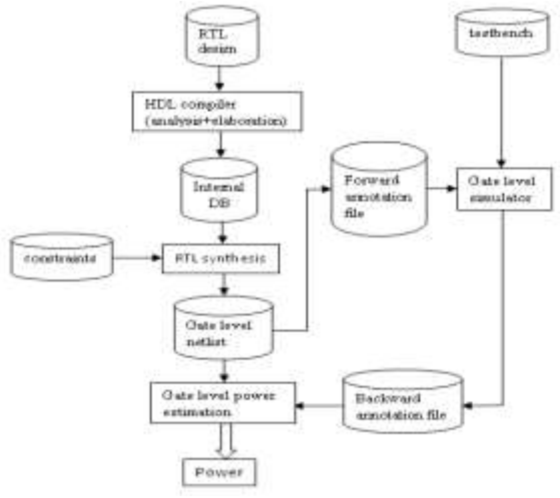


Figure 11 Gate power estimation flow

The comparison of the two flows exposes the sources of inaccuracy of RTL against gate-level estimation. First, the use of the internal database as a working description; the RTL description is structural, but has little to do with the synthesized gate-level netlist (but the functionality, of course).Secondly, the granularity of simulation; RTL simulation will annotate only a subset of the nets annotated in the gate-level flow. These two differences are also the sources of the speedup of RTL against gate-level estimation.

Architecture level/rt level power estimation

Introduction:

The architectural level is the design entry point for the large majority of digital designs. The design decisions at this level can have a dramatic impact on the power budget design. Once the architecture is defined and specified, using a functional or register- transfer level (RTL) description, a more refined power profile can be constructed which paves the way for more detailed optimizations. The functional blocks may be adders, multipliers, controllers, register files and memories [6]. Since the detailed information of each block is not available, generally speaking, RTL power estimation is less accurate than gate- or circuit-level power estimation. However, the increased complexity (i.e. millions of transistors) has become RTL power estimation and optimization a very critical step in the design procedure, since the existing CAD tools are mature for handling automatically the lower design levels [8][31].A register-transfer-level (RTL) data path is characterized by the use of predesigned components such as arithmetic components, register, ALUs, etc. (control units, buses, memories, and clock trees are excluded from this category). Expanding the RTL data path to a lower-description level, for instance gate level, where an accurate measurement can be performed, the power dissipation can be estimated with high accuracy.

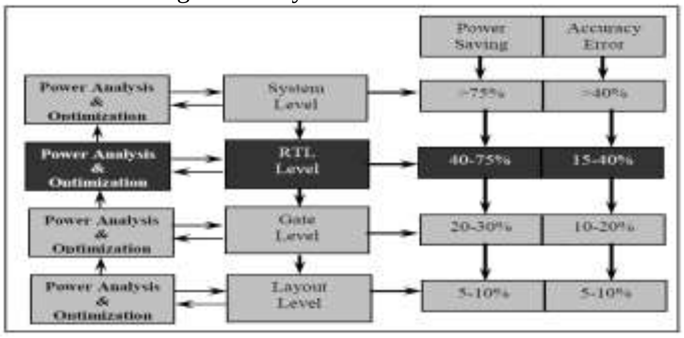

Power Saving
Accuracy Error
Power Analysis & Optimization
System Level
>75%
>40%
Power Analysis & Optimization
RTL Level
40-75%
15-40%
Power Analysis & Optimization
Gate Level
20-30%
10-20%
Power Analysis & Optimization
Layout Level
5-10%
5-10%

Fig 12: Power Estimation Levels

There are basically two advantages of doing this analysis:

➢If the design is not going to meet the project's objectives for power and/or performance, the designers will know it at the initial stage. Moreover, different architectural option can be tested in power before the logic design stage.

➢The biggest effects on power and/or performance can be achieved at the RTL and without architecture level analysis, the effects of proposed changes will neither be found out early nor determined

Activity Estimation for Correlated Inputs:

Activity estimator, we will refer to [41] for the definitions of *spatial* and *temporal* correlations.

Temporal correlation:

A signal *x* is said to be *temporally correlated* if an event (occurrence of certain logic state) at a given time is correlated to an event at some past time. In this work, we will concentrate only on correlations across one clock edge. For temporally correlated primary inputs, the temporal correlation parameter for the *it* input, *TCi*, is defined as

$$TC_i = \mathcal{P}\left\{x_i^t \wedge x_i^{t-1} = 1\right\}$$

Where *t*−1 and *t* are consecutive clock cycles and where *P{・}* denotes probability. Temporal correlation coefficient (γ_i) for *youth* input is defined as

$$\gamma_i = \frac{\mathcal{P}\left\{x_i^t \wedge x_i^{t-1} = 1\right\} - P(x_i)^2}{P(x_i)(1 - P(x_i))}$$

Where $P(x_i)$ is the probability at an input node *xi*, and the only quantity which is unknown in is $P\{x_i^t \wedge x_i^{t-1}=1\}$. Therefore it is possible to estimate *γi if* TC_i can be determined. In [41], the authors show that TC_i can actually be determined from the knowledge of *P* (*xi*) and *D* (x_i), and hence temporal correlation of the primary inputs is taken care by *P* (x_i) and *D* (x_i) without a need to introduce an additional parameter to represent it. The relationship between TC_i, *P* (x_i) and *D* (x_i) is given by

$$TC_i = P(x_i) - \frac{D(x_i)}{2}$$

Spatial correlation:

A signal *x* is said to be *spatially correlated* to another signal *y* if their events are correlated. In this work, we will concentrate only on pairwise correlations. Once again, referring to [41], we can define SC_{ij} , the spatial correlation between the *i*th and *j*th inputs as [43]

$$SC_{ij} = \mathcal{P}\left\{x_i \wedge x_j = 1\right\}$$

i.e., The probability of the inputs being high simultaneously. This definition of SC_{ij} as a measure of spatial correlation follows from the definition of the correlation coefficient as Introduced in [42]

$$\rho_{ij} = \frac{\mathcal{P}\left\{x_i \wedge x_j = 1\right\} - P(x_i)P(x_j)}{\sqrt{P(x_i)P(x_j)(1 - P(x_i))(1 - P(x_j))}}$$

From the definition given in, it is clear that SC_{ij} is sufficient to capture ρ_{ij} .

Instead of considering all the pairwise correlation coefficients, it is possible to define SC_{in} (*average spatial correlation coefficient*, i.e., average of all SC_{ij} terms). This parameter can be calculated as

$$SC_{in} = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathcal{P}\left\{x_i = 1, x_j = 1\right\}$$

where *n* is the number of primary inputs. In [41], the authors go on to find upper and lower bounds for *scene* as

$$\frac{nP_{in}^2 - P_{in}}{(n-1)} \leq SC_{in} \leq P_{in}$$

Where *Pin* is the average signal probability for primary inputs. In our work, we will use a parameterized measure of spatial correlation instead of directly using *SCin*

RT-level Power Management:

Digital circuits usually contain portions that are not performing useful computations at each clock cycle. Power reductions can then be achieved by shutting down the circuitry when it is idle.

Precomputation Logic:

Precomputation logic presented in [14], which explains the idea of duplicating part of the logic and computing the next circuit output values one clock cycle before they are required, and then uses these values to reduce the total amount of switching during the next clock cycle. In fact, knowing the output values one clock cycle in advance allows the original logic to be turned off during the next time frame, thus eliminating any charging and discharging of the internal capacitances. Obviously, the size of the logic that pre-calculates the output values must be kept under control since its contribution to the total power balance may offset the savings achieved by blocking the switching inside the original circuit. Several variants to the basic architecture can then be devised to address this issue. In particular, sometimes it may be convenient to resort to partial, rather than global, shutdown, i.e., To select for power management only a (possibly small) subset of the circuit inputs.

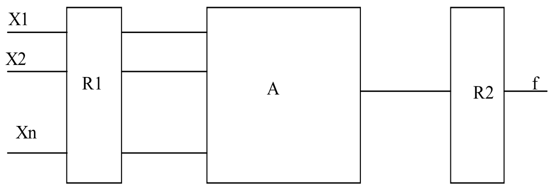


Figure 13: A pipeline stage of a data path

Figure 13 shows a combinational block A that implements n-input, single-output Boolean function precomputation logic for the complete input-disabling architecture.

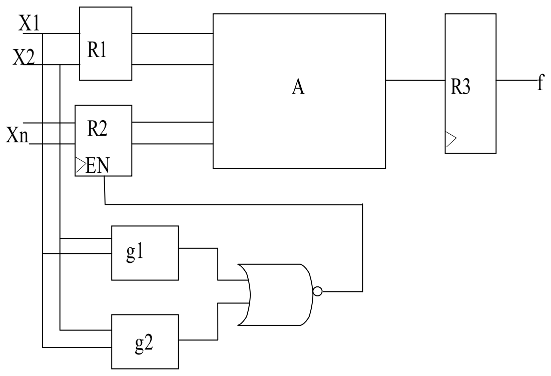


Figure 14: A precomputation logic realization of the pipeline stage (subset-input disabling architecture)

f, with registers R1 and R2 connected to its inputs and output pins, respectively. A precomputation architecture realization of this same logic block placed between register sets R1 and R2 is depicted in Figure 14. The key elements of the precomputation architecture are two n-input, single-output predictor functions g1 and g2, which satisfy the following constraints:

$$\begin{aligned} g1=1 &\Longrightarrow f=1 \\ g2=1 &\Longrightarrow f=0 \end{aligned}$$

If, at the present clock cycle, g1 or g2 evaluate to one, then the load enables signal, LE, goes to zero, and the inputs to block A at the next clock cycle are forced to retain the current values. Hence, no gate output transitions inside block A occur, while the correct output value for the next time frame is provided by the two registers located on the outputs of g1 and g2. Note that the precomputation logic is a function of a subset of the input variables; hence, it is called a "subset input-disabling architecture [14]." The synthesis algorithm suffers from the limitation that if a logic function is dependent on the values of several inputs for a large fraction of the applied input combinations, then no reduction in switching activity can be obtained. In, the authors focus on a particular sequential precomputation architecture in which the logic is a function of all of the input variables. The authors call this architecture the "complete input-disabling architecture." This complete input disabling architecture can reduce power dissipation for a larger class of sequential circuit's compared to the subset input-disabling architecture. The authors present an algorithm to synthesize

Clock Gating:

An approach to RT and gate-level dynamic power management, known as gated clocks [14] selectively stop the clock, and thus, force the original circuit to make no transition, whenever the computation that is to be carried out at the next clock cycle is redundant. In other words, the clock signal is disabled according to the idle conditions of the logic network. For reactive circuits, the number of clock cycles in which the design is idle in some wait states is usually large. Therefore, avoiding the power waste corresponding to such states may be significant. The logic for the clock management is automatically synthesized from the Boolean function that represents the idle conditions of the circuit (Figure 15.) It may well be the case that considering all such conditions results in additional circuitry that is too large and too power consuming. It may then be necessary to synthesize a simplified function, which dissipates the minimum possible power and stops the clock with maximum efficiency.

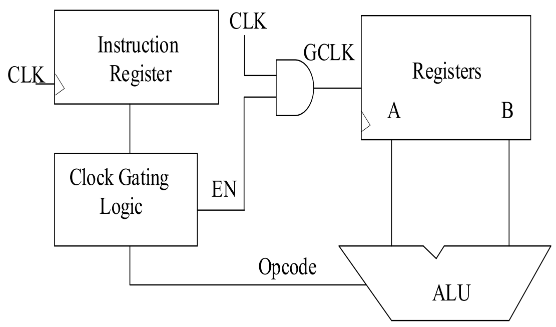


Figure 15: Clock gating logic for ALU in a typical processor micro architecture

With negative-edge triggered flip-flops

The use of gated clocks has the drawback that the logic implementing the clock-gating mechanism is functionally redundant, and this may create major difficulties in testing and verification. The design of highly testable-gated clock circuits are discussed in [15].

Proposed method:

The most frequently used RTL data-path components are arithmetic components (adders, subs tractors, multipliers), multiplexers, comparators, registers, multiply accumulate units, and ALUs. Their architecture includes a uniformly repeated primitive cell, e.g. 1-bit full adder, flip-flop, and are characterized by regularity, symmetry, and frequently by separability. Exploiting these properties, which have not properly been considered in previously reported models, an accurate and low-complexity power model for each component with N inputs and K primitive cells can be developed. It is assumed that the input vectors are applied on an RTL component in parallel and the component is stabilized before the loading of the next input vector [11]. The RTL power estimation is performed using a set of small LUTs instead of constructing one large LUT for the whole component. For each component two types of LUTs are used, the primary, and the secondary UT. Based on the principles of superposition and 'divide and conquer', and considering the architecture and functionality of each component, the component is partitioned into subcomponents to reduce the size of the primary LUT. In particular, the primary LUT corresponds to the power dissipation of a block of L ($1\leq L \leq K$) primitive cells. It contains the power dissipation values for any input vector pair as well as any additional useful information such as the transition number of the interconnecting signals among the component blocks, the steady values of the component output, etc. However, in many cases there is the data transmission into the interior of the component has been partitioned, an extra computation must be performed to consider this data transmission in terms of power dissipation estimation [32] [33].

The power values of the LUTs may be derived either by a real gate-delay simulator or by a circuit- or transistor-level simulator. Using values derived by a circuit-level simulator, issues such as slopes, wire capacitance and timing information could be considered making the model more accurate. However the complexity of the model in terms of time and memory will be increased. Additional information has also to be stored to consider circuit-level characteristics, while an extra computational cot will be paid to manipulate this information, considering the component's behavior at the circuit-level.Since the power values of the LUTs are calculated by a real delay gate-level estimator and the glitches from one blocks to the next are captured by performing functional simulation, the accuracy of the proposed model is identical to the accuracy of a power estimation based on a gate-level simulator. Moreover the model is not sensitive to the training set, since the primary LUT contains the power dissipation values that correspond to all possible input combinations. Finally, there is a trade-off between the computational time and the size of the LUTs, depending on the length L of the block used. It must be stressed that an RTL design is characterized by the instantiation of fixed pre design components, where the power consumption can be evaluated by summing the power dissipation of each component. The power value will be different after logic synthesis, as it will be after placement and routing too. However at a high level such as RTL, we need a power estimation to select different designs in terms of power. It is expected that the relative power difference, which has been detected at the RTL, will remain after logic synthesis when the same synthesis flow and tools are used. Additionally, by using fixed RTL modules, the requirements of a design project not strongly dependent on time can be met, as possible only buffers will be added at the synthesis step and a few logic optimizations made [11].

Power estimation from LUT:

The table reference method in the table look-up step is also important for accurate power estimation. This section explains important metric *distance* to find out the proximal entry from a LUT. This section describes one of the effective distance calculation proposed in our previous work [12]. Let n_p, p^i_{in}, and p^i_{LUT} be the number of parameter types, *i*-th parameter extracted from input data, and *i*-th parameter in one entry in the LUT, respectively. The distance *dist* is calculated as follows:

$$dist = \sqrt{\frac{1}{n_p}\sum_{i=0}^{n_p-1}(dp_i)^2}$$

Where *dpi* is a ratio for *i*-th parameter, and defined as follows:

$$dp_i = 1 - \frac{p^i{}_{in}}{p^i{}_{LUT}}$$

The distance is defined with a ratio in each parameter and RMS to define the *dist*. In the table look-up step, the entry in the LUT which has the shortest distance with the extracted parameters is selected, and the corresponding power value is output as the estimated power [12].

RTL power estimation flow:

Depicts the typical estimated flow at the RTL and gate level. The starting point of the two flows is a design written in some HDL. After analysis and elaboration by the HDL compiler, the design translates into a technology-independent internal format that contains the four types of components mentioned in above: RTL modules (macros), gates, memory elements and MUXes [13]. The two flows start differentiating starting from this internal description. In true RTL estimation (Fig. 16), a forward annotation file is produced that contains the list of nets to be monitored during (RTL) simulation. RTL simulation takes this file, the HDL description and a test bench, to produce a backward annotation file, consisting of all the nets specified in the forward annotation file, this time annotated with switching activity and static probability values. An RTL power estimator takes the internal database produced in the first step and this activity information, and calculates a power estimate. This estimate is basically obtained by exercising specific power models for the objects of the internal database with the activity values derived from simulation [35] [40].

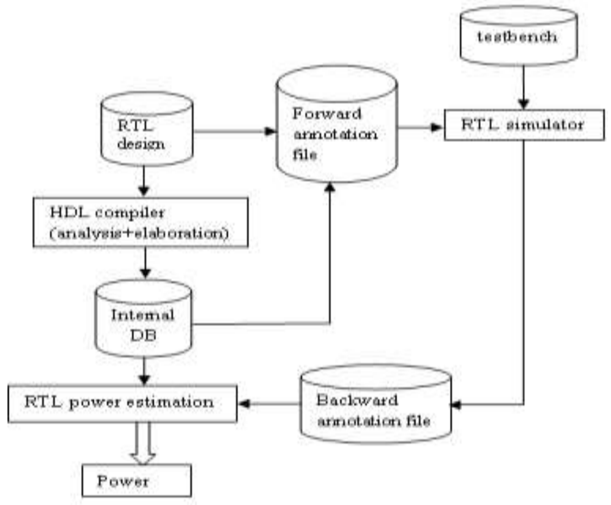


Fig 16: RTL power estimation flows

Behavioral level power estimation

Introduction:

Algorithmic level also known as behavioral level, describes the behavioral of the domain in terms of algorithms, flowcharts, processes and structures. The hardware modules that are used to represent the Behavioral domain, such as the control path and data path, are specified in the Structural domain. Clustering or partitioning of similar operations that might be described in the structural domain is described in the Physical domain. At the behavior level, not much information is available about the gate-level structure. Hence, abstract notions of physical capacitance and switching activity are used to predict power dissipation. These techniques can be classified into three broad categories: information theory based, complexity based, and synthesis based approaches [23].

Information theory based approaches:

Information theory based approach depends on information theoretic measures of activity (i.e., entropy) to estimate power dissipation. Entropy characterizes the randomness of a sequence of vector and hence is related to the switching activity. It shows in that, under the temporal independence assumption, switching activity of a bit is upper bounded by 1/2 of its entropy. The power dissipation in the circuit can be expressed as Power = $1/2V\ ^{2}fC_{tot}E_{avg}$, where C_{tot} is the total capacitance of the logic module and E_{avg} is the average of line activities, which is in turn approximated by 1/2 of the average entropy h_{avg}. The average line entropy h_{avg} is calculated by a closed-form expression parameterized by average bit-level entropies of circuit inputs/outputs (and number of inputs, outputs). Average input entropy can be derived from input sequences. Average output entropy is derived either by using an effective information scaling factor and number of logic level in the circuit if gate-level structure is given; or by a compositional technique based on pre characterization of library modules in terms of their entropy transmission coefficient if only functional/data-flow information is given. In, word-level entropy is used instead of bit-level entropy. A similar closed-form expression for h_{avg} is proposed using sectional (word-level) input/output entropy. The sectional entropies of circuit inputs and outputs may be obtained by monitoring input output signal values during a high-level simulation of the circuit. In practice, they are approximated as the summation of individual bit-level entropies. The total module capacitance can be calculated by summing up the entire gate loading and wire capacitance if gate-level structure is given. Otherwise, C_{tot} is estimated by a quick mapping (e.g., mapping onto universal gates) or by information theoretic models that relate the total capacitance to input and output entropies [23].

Complexity-based approaches:

Complexity-based models relate the circuit power to the circuit complexity. Most of the proposed complexity-based models rely on the assumption that circuit complexity can be represented by the number of “equivalent gates”. Muller-Glaser et al. Proposed a *chip estimation system* that computes the average power of a logic module as $Power = fN\ (Energy_{gate}+0{:}\ 5V\ 2C_{load})\ E_{gate}$. Here, f is the clock frequency, N is the equivalent gate count for this module, $Energy_{gate}$ is the average internal energy dissipation for an equivalent gate, C_{load} is estimated capacitance based on the average fan-out in the circuit and the wire load model, and E_{gate} is an average output activity per clock cycle for an equivalent gate. E_{gate} is dependent on the functionality of the module. These data are pre-calculated and stored in a library and are independent of the implementation style and the circuit environment. In Nemani et al. presented a high-level estimation model for predicting the area of an optimized single-output Boolean function. The model is based on the assumption that the area complexity of a Boolean function is related to the distributions of the sizes of the onset and offset of the function. Area measure is used for total capacitance estimation and hence the high-level power estimation. This work has been extended to area estimation of multiple output functions [24]. Complexity-based power prediction for controller circuitry was proposed by Landman and Rabaey. Based on the knowledge of its target implementation style (i.e., precharged pseudo-NMOS or dynamic PLA), the number of inputs, outputs, input/output activities, etc., This technique can give a quick power estimation. The accuracy of the estimates depends on the empirical parameters (regression coefficients), which are derived from curve-fitting and least-square fit error analysis of low-level simulation of previous design.

Synthesis-based approaches:

Synthesis-based models assume an RT-level template and produce estimates based on that assumption. It requires the development of a quick synthesis capability that makes the relevant behavioral choices. Important behaviour choices include type of I/O, memory organization, pipeline issues, synchronization scheme, bus architecture, and controller design. After the RT-level structure is obtained, power consumption can be estimated by either simulation or static analysis of the circuit structure/functionality [24].In order to address the two questions, a comparison is made between the behaviour of gibbons in disturbed and undisturbed situations and the subsequent implications for monitoring are assessed. The behavioural changes can both affect the parameters needed for density estimation and violate the (critical) assumptions of the methods employed.

Material and methods:

Gibbons are territorial and live in monogamous family groups consisting of an adult pair within one to four offspring. Gibbons are completely arboreal, and are largely frugivorous. Paired groups give loud morning calls, which can be heard over several kilometres, whereas single individuals rarely call. The present study concerns data collected on Bornean gibbon H. muelleri in East Kalimantan (KayanMentarang National Park and adjacent areas in 1996 [115°51E, 2°50’N]) and Javan gibbon H. moloch on Java (Gede-Pangrango National Park and adjacent areas in 1994-1999 [107°00’E, 6°45’S], and Diengmountains proposed National Park and adjacent areas in 1995-1999 [109°35’E, 7°06’S]).Undisturbed and disturbed study sites were selected either in close proximity and were similar in climate, original vegetation type, altitude and topography (Gede-Pangrango and Kayan Mentarang), or a forest area was sampled before and during logging during the same months of the year. Given the close proximity and similarity of the forest areas, it is anticipated that the behaviour of the gibbons prior to the commencement of disturbance did not differ significantly. Sets of disturbed and undisturbed areas had mean densities differing less than 10%, which was established by a number of techniques (line-transects, range mapping, fixed point counts). For the present study, disturbance is taken in a rather broad term and may include hunting, encroachment, small scale logging, commercially (selective) logging, or a combination. Behavioural measurements were collected along line transects, on vantage points during fixed point counts, and ad libitum while surveying in the forest. Singing behaviour of at least eleven *H.moloch*groups was monitored in Dieng for 35 days in Sept-Oct 1998 (pre-logging) and for 25days in Sept-Oct 1999 (during logging). Some additional data on singing behaviour of Siamang H.syndactylus was collected in Way Kambas National Park, Sumatra [105°36'E, 4°50'S].For all analyses non-parametric statistics were used and Yates's correction for continuity was applied in the Chi-sq. tests where appropriate.

Behavioral-Level Power Estimation:

Typical approaches at the algorithmic- or behavioral-level assume to adopt some architectural styles or templates in order to obtain power estimates based on the exploration of a limited set of design solutions. Essentially, the behavioral approaches differ on the strategy adopted for the activity prediction: the behavioral methods can be classified as static and dynamic activity prediction techniques. The goal of the former techniques is the estimation of the access frequency of different HW resources, by statically analyzing the behavioral description of the functions to be implemented. The latter techniques are based on a dynamic profiling to determine the activation frequencies of various resources and the memory accesses. Developed a power estimation strategy based on a static profiling of the Control Data Flow Graph (CDFG) representing the design behavior. The analysis has been carried out in the context of the HYPER-LP high-level synthesis system targeting DSP-oriented applications. The power dissipated by some HW resources, such as data-path modules, has been analytically estimated from the CDFG. Conversely, for other modules, such as interconnects and controllers, for which the power information available at the behavioral-level is not sufficient, statistical models were built to estimate power based on a stochastic study on several ASICs. Basically, the power associated with a generic hardware resource has been estimated as [25]:

$$P = 1 / 2\ Na\ CaV2dd\ fs$$

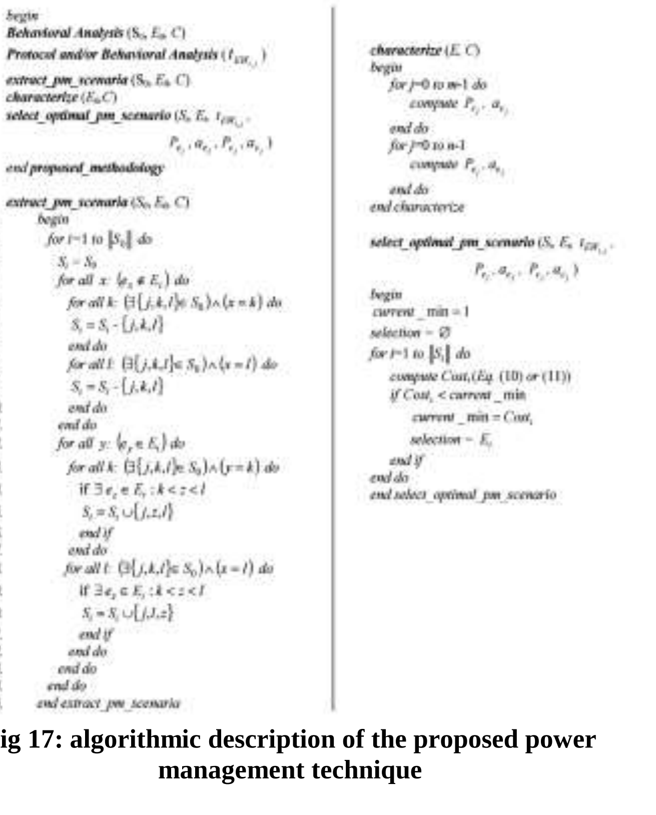
Where *Na* is the number of resource accesses over the computational period, *Ca* the average capacitance switched per access and *fs* the sampling frequency. The capacitance estimates have been obtained by the empirical characterization of fixed-activity models of the different HW resources. The numbers of resource accesses have been analytically calculated from the algorithm for the execution units, the registers and the memories, while they have been determined statistically from benchmarks for the interconnections and the control logic. Then, the estimation models have been included into an exploration tool that, given the CDFG description of an algorithm and a library of hardware modules, explores the space of the available solutions for different values of clock periods and supply voltages. The results have been compared with an architectural-level power estimator, called *Stochastical Power Analysis (SPA)* , on *23* different chips, showing an average error of approximately *20%*. Dynamic activity prediction of the behavioral-level is based on a dynamic profiling to determine the activation frequencies of various resources. During the simulation of a user supplied set of input patterns, the activities related to the frequency of various types of operations and memories accesses are gathered. These access frequencies are then plugged into a model similar to those used in the static approach. Examples of the dynamic approaches are the *Profile-Driven Synthesis System (PDSS),* that receives as input a behavioral subset of VHDL, and the *Power-Profiler* approach described in. The main advantage of dynamic versus static approaches are a higher accuracy, since data dependencies are taken into account, whereas the main disadvantages are related to the slower efficiency in terms of speed and the need of a set of user-supplied typical input patterns.

Proposed technique:

The proposed behavioral-level power management technique for digital receivers is described. For clarity reasons, some definitions are given first:

- Consider a behavioral level description partitioned to a number of behavioral clusters. We denote the behavioral clusters' set as C= {c_i|i=0,1,., m-1, m €N}, where N is the set of the physical numbers and m=||C|| is the total number of the behavioral clusters, where ||•|| denotes the cardinality of a set.
- The event is defined as an executing behavioral cluster.
- System period, T_{SYSTEM}, is the minimum fraction of time during which a sequence of events is not repeated.
- Event window, $EW_{i,\ j}$, is the fraction of system period that lies between the events e_i and e_j.

The proposed event-driven power-management technique is based on the fact that the unobservability of a circuit node at the behavioral level is introduced after the occurrence of an event. A system behavior is a result of a collection of interrelated functions. For instance, MPEG2 application requires, among others, the execution of vector quantization and Huffman coding functions. Their basic functions can be considered as behavioral clusters. Similarly, for the considered receiver's application, a behavioral cluster, for instance, can be a function that performs receiver's synchronization or a receiving symbol correction [46]. In almost all behavioral descriptions of a DECT receiver, there are behavioral clusters that their goal is to check whether an event occurs or not without modifying the output variables between the occurrences of two events. Such clusters are characterized by the unobservability for one or several event windows and their shutdown can lead to significant power savings. For example, a behavioral cluster responsible for synchronization does not change its outputs for a while, after the synchronization is achieved.



```
begin
Behavioral Analysis ($S_0$, $E_0$, $C$)
Protocol and/or Behavioral Analysis ($t_{EW_{i,j}}$)
extract_pm_scenaria ($S_0$, $E_0$, $C$)
characterize ($E_0$,$C$)
select_optimal_pm_scenario ($S_i$, $E_i$, $t_{EW_{i,j}}$,
        $P_{e_j}$, $a_{e_j}$, $P_{e_j}$, $a_{e_j}$)
end proposed_methodology

extract_pm_scenaria ($S_0$, $E_0$, $C$)
    begin
      for i=1 to $\|S_0\|$ do
        $S_i = S_0$
        for all $x$: $(e_x \notin E_i)$ do
          for all $k$: $(\exists\{j,k,l\} \in S_0) \wedge (x = k)$ do
            $S_i = S_i - \{j,k,l\}$
          end do
          for all $l$: $(\exists\{j,k,l\} \in S_0) \wedge (x = l)$ do
            $S_i = S_i - \{j,k,l\}$
          end do
        end do
        for all $y$: $(e_y \in E_i)$ do
          for all $k$: $(\exists\{j,k,l\} \in S_0) \wedge (y = k)$ do
            if $\exists e_z \in E_i : k < z < l$
              $S_i = S_i \cup \{j,z,l\}$
            end if
          end do
          for all $l$: $(\exists\{j,k,l\} \in S_0) \wedge (x = l)$ do
            if $\exists e_z \in E_i : k < z < l$
              $S_i = S_i \cup \{j,l,z\}$
            end if
          end do
        end do
      end do
    end extract_pm_scenaria

characterize ($E$, $C$)
begin
  for j=0 to m-1 do
    compute $P_{e_j}$, $a_{e_j}$
  end do
  for j=0 to n-1
    compute $P_{e_j}$, $a_{e_j}$
  end do
end characterize

select_optimal_pm_scenario ($S_i$, $E_i$, $t_{EW_{i,j}}$,
        $P_{e_j}$, $a_{e_j}$, $P_{e_j}$, $a_{e_j}$)
begin
 current_min = 1
 selection = ∅
 for i=1 to $\|S_i\|$ do
   compute $Cost_i$(Eq. (10) or (11))
   if $Cost_i$ < current_min
     current_min = $Cost_i$
     selection = $E_i$
   end if
 end do
end select_optimal_pm_scenario
```

Fig 17: algorithmic description of the proposed power management technique

The granularity of a behavioral cluster complexity is user specified. Depending on the features of an application, the designer can specify behavioral clusters with finer or coarser granularity of complexity. It is not always clear in an abstract behavioral description (e.g. CDFG) whether a cluster performs useful computations are not. Thus, a behavioral analysis is required to identify the clusters that can be shutdown and also the events that enable and disable these clusters. The fundamental steps of the proposed behavioral level event-driven management technique is described in Fig 17.

Behavioral analysis:

Behavioral analysis indicates the candidate clusters at the behavioral level for power management. This also involves the identification of the events that can trigger the shutdown of the behavioral clusters.

Definition 1. We define as events' set the ordered set E_0= {e_i|i=0,1,., n-1, n€N}, where e_i is an event that either introduces or ceases unobservability for a certain behavioral cluster, and n=||Eo || is the number of such events (||• ||denotes the cardinality of the set •). The set E_0 is ordered according to the time occurrence of the events e_i. Using mathematical notations, the behavioral analysis aims at defining the following set:

$$S0 =\{(j,k,l)|(C_j€C) \wedge (e_k,e_l € E0) \wedge (e_k \text{ introduces unobservability for } c_j) \wedge e_l \text{ caeses unobservability for } c_j)$$

The simplest way to perform behavioral analysis is simulation. Concerning that the design of a wireless system starts with a behavioral level description, using robust and mature automated tools, for instance Mat lab, the required behavioral analysis can be performed in an easy and accurate manner. Furthermore, in many cases the simulation is not always needed, since the behavioral analysis can also be performed manually by any designer familiar with the behavioral description of the design. In any case, the behavioral analysis can be visualized by the use of an event graph.

Behavioral Level of Macro Modeling:

The estimation of power at the behavioral level of design is much more complex as compared to the estimation of power at the RT level. First, the behavioral description is not HW oriented and looks much the same as any software program. Its mapping to the HW architecture may be ambiguous, different implementation strategies can be used. Second, here we cannot rely on the specific technological library components. At the behavioral level, computation of power must be approximated in order to account for the limited knowledge of the circuit. Therefore, a number of the high-level analysis techniques such as statistical analysis [36, stochastic methods, and macro modeling [37] are used. These techniques are usually based on the development of abstract power models, which are used for design space exploration to evaluate the relative impact of design decisions on the quality and characteristics of the final design. The estimated power consumption values provided by such models are neither absolute nor physically accurate, because at the highest level of abstraction the limited knowledge of the physical structure of the design does not allow to compute meaningful power estimates [38]. Such models can be built analytically by deriving a formula for each behavioral operation, which depends on a number of physical parameters such as switching or capacitance. Another way is to develop an empirical model or macro model, which is based on the approximation of the actual measured power dissipation values. The basic idea behind power macro modeling is to generate a mapping between the power dissipation of a circuit and certain statistics of its input signals. Such macro models can be used during modeling instead of detailed hardware models resulting in modeling speedup

SYSTEM-LEVEL ESTIMATION:

Introduction: This chapter presents a survey of the most important methodologies for system level estimation and design found in the literature, and compares them to Fun time. Three categories of tools are identified: simulation-based tools, analytical tools and tools that are a combination of multiple approaches. Due to their importance and to the lack of a standard approach, both system level estimation and system-level design in general are a hot research topic today and the focus of a high number of research groups. The most significant estimation tools for SLD/SLE found in the literature. In doing so, the surveys presented in [17] and [18] are partially taken as a reference and adapted to the purposes of this work. At the same time, a comparison is presented between these approaches and Funtime, which emphasizes key similarities and differences. System-level estimation tools can roughly be classified into three broad categories: simulation-based tools, analytical tools, and tools that are a combination of different approaches. Although the following subsections review each category, a large space is dedicated to the simulation-based approaches and, in particular, to describing System C and Transaction Level Modeling (TLM). The reason is that this simulation-based approach has lately gained consensus and has become quite popular in both the industrial and academic community. This is why SystemC/TLMis also used in this work as the reference system-level approach when validating Funtime for estimation speed [23].

System-Level Estimation:

There have been many attempts to estimate the energy used in a particular system design at all levels of abstraction. At the lowest levels the estimates are quite accurate, but these methods can be used only when a design is complete and the application is well documented. At the gate level of abstraction, each gate is precharacterized for power and the total power is then calculated on the basis of switching activity of nodes in the design, which is obtained by simulation or in a probabilistic manner. Power estimation at the register transfer level is similar to that used at the gate level; the primary difference is the complexity of pre-characterizing each component for power. Several methods have been tried, including characterization through extensive simulation and the use of lookup tables or analytical functions to summarize results [44]. Recent years have seen significant research interest in system-level power estimation. Most of this research has focused on power modeling techniques for individual system components (*e.g.*, processors, memories, on-chip buses, peripherals, user-defined logic, *etc.*). These power models can be integrated into system-level simulation frameworks to provide power estimation capabilities. Power models within a system-level simulation environment to achieve a superior trade-off between overall power estimation accuracy and efficiency. A power estimation framework that integrates heterogeneous component power models using a network of "power monitors". The monitor-based framework provides an intelligent interface, facilitating the seamless integration of component simulation models on one hand, and a variety of heterogeneous power models on the other. Power monitors enable each component model to be associated with multiple (distinct) power models of differing accuracy and efficiency, or with configurable power models that can be tuned to different accuracy/efficiency levels. The power monitor exercises fine-grained control over the different power models through dynamic selection and configuration of power models based on information gathered during simulation [39].

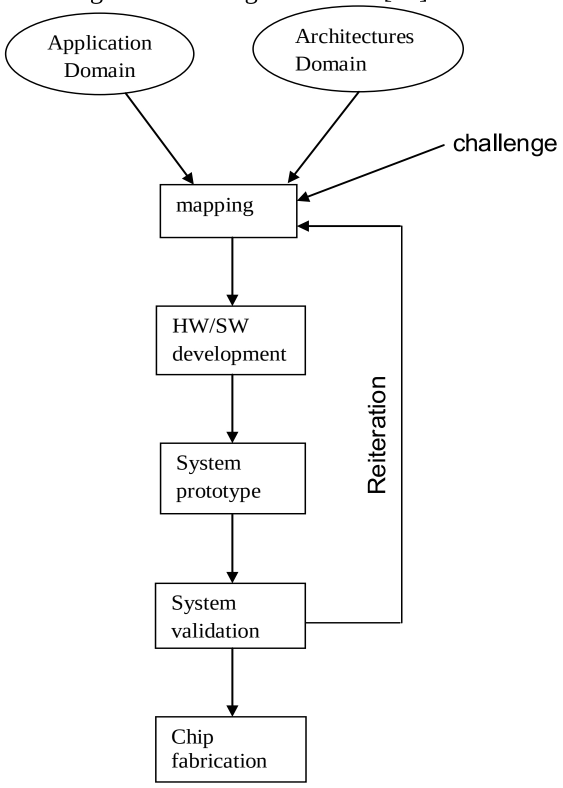


Fig 18: System-level design challenge: the mapping phase

Shifting towards higher levels of abstraction has proved to be a winning strategy for dealing with increasing complexity. Indeed, by abstracting away the low-level details, implementation is faster, which means lower engineering effort, lower cost and lower time to market, as well as higher productivity. Decisions made at the system level have a very strong impact on the quality of the final product, since the degree of achievable optimization is normally proportional to the abstraction level and, indirectly, to the point in the design flow where decisions are taken: the earlier the better. At the system level, the question that system architects have to answer to is the following: given a set of applications and a set of possible architectures, what is the best architecture on which to map this set of applications. The expression best architecture refers to the properties of architecture in terms of metrics like performance, power consumption and silicon area, for a given set of applications. For example, what is the power and the performance impact of using a voltage-frequency scaling scheme rather than a fixed frequency? What is the power and the performance impact of varying the number of levels in the memory hierarchy? What is the best interconnect to use: a bus or a NoC? What is the advantage/disadvantage of implementing part or the whole set of applications in hardware rather than software? These are just examples of the hardest choices a designer has to make. Since they are so important, taking the right system-level decisions from the beginning is crucial, especially when complexity grows: any error at this early stage would lead to annoying design reiterations, as shown in Figure 17, with a consequent high loss of time, money and, probably, a sub-optimal final implementation [19].However, although very important, decisions at system level are very hard to take and this is for two main reasons: the first is that, at the system level, the design space to consider is extremely broad as a consequence of the limited amount of implementation details available.

Figure 19 shows the relation between the design space width and abstraction level. The second reason is that the impact of the decisions taken at system level is not known until a very late stage of the design process, which can take months of work. From the second reason mentioned above, it can be concluded that the lack of a quick and accurate System-Level Estimation (SLE) approach is one of the main obstacles for successful system-level design today. In fact, if an efficient system-level methodology for energy and performance estimation was available, it would be possible to carry out a reasonably comprehensive Design Space Exploration (DSE), and thus judge from the beginning of the design flow which architecture is the most suitable for a certain application domain, in terms of performance and power consumption. In addition, estimation at any abstraction level is a requirement for the implementation of automatic synthesis tools, since it is only after estimation that the tool can judge what the best solution is. Efficient estimation of lower abstraction levels has allowed us to have quite mature automatic synthesis tools today. Estimation at the physical level requires accounting for the individual capacitance and resistance contributions coming from each transistor and interconnecting wire. Estimation at this level is extremely accurate, but also very slow. Simulation at the physical level is also very slow and is thus feasible for only very small designs and for a very short design execution time. At the gate level, estimation is simplified by the fact that standard cells are used, whose physical properties are pre-characterized. Only the impact of cell-to-cell connecting wires has to be estimated separately, which is done using so called wire load models. Estimation at this level is less accurate, although faster, and bigger design sizes can be simulated. At the RT level, Hardware Description Language (HDL) languages is used to describe in words what RTL synthesis translates into logic gates. The simulation is very common at RT level and reasonably fast for medium size designs running very short chunks of application. However, estimation made at this level loses accuracy due to the lack of enough physical details. In general, the increase of the abstraction level is directly proportional to an increase of the estimated speed and inversely proportional to the estimation accuracy [23] [45]. When it comes to system level, the lack of an efficient estimation methodology has been an obstacle to having mature automatic system-level synthesis tools available today. In fact, the operation of mapping the system-level functional description of the actual architecture is still largely done manually. The decision-making approach used by system designers has been mostly relying on their acquired experience, on the comparison with previous designs and on rules of thumb. However, while this approach can still work with small/medium-size systems, its application to today's more and more complex systems has become unrealistic and the need for a more systematic and accurate approach has become a necessity. TLM has appeared at the beginning of the last decade as a simulation-based approach raising the abstraction level above RTL and as a starting point for synthesis. In essence, TLM abstracts away the RTL details and models functionality and communication among the system modules. Communication is seen as an exchange of transactions between architectural resources.

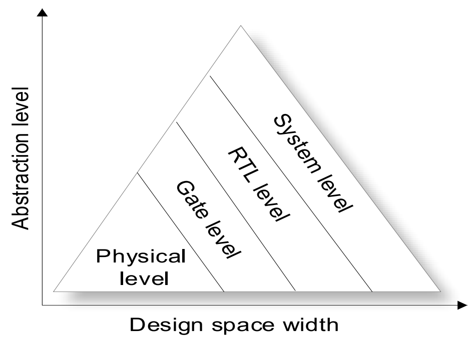


Fig 19: Design space width versus abstraction level

As a result, TLM has proved to be much faster than RTL [17]. In spite of that, even TLM could be too slow to allow proper simulation of future complex systems. In addition, the problem remains of how to obtain for example accurate power estimation at the system level, since TLM does not provide intrinsic support for power estimation, and waiting until reaching the gate-level design phase is not an option.

The System-Level Power Model:

The proposed power estimation model [15] is composed of *three* main cooperating sub-modules: (i) the *memory hierarchy*, (ii) the *bus encoder,* and (iii) the address/data *stream generator*, which have been integrated into an analysis tool, written in C++.

System Bus Hierarchy:

Modern system-on-a-chip embedded media systems include many components: a high-speed processor core, hardware accelerators, a rich set of peripherals, direct memory access (DMA), on-chip cache and off-chip memory. The system architecture considered in the study includes a single-core microprocessor, several peripherals, and off-chip SDRAM memory, and is similar to many current embedded platforms. Without losing generality, the system architecture definitions can then be used to conduct the majority of the experiments. For multimedia applications, data throughput requirements are increasing higher than what they were ten years ago. Today, for a D1 (720x480 pixel resolution) video codec (encoder/decoder) media node, it needs to be able to process 10 million pixels per second. This workload requires a multimedia-specialized processor for computation, peripheral devices to support high speed media streaming and data conversion via a parallel peripheral interface (PPI), and a synchronous serial port (SPORT) for interfacing to high speed telecom interfaces. The high data throughput requirements associated with this platform make it impossible to store all the data in an on-chip memory or cache. Therefore, a typical multimedia embedded system usually provides a high-speed system-on-a-chip microprocessor and a very large off-chip memory. The Analog Devices Black fin family processors [4], the Texas Instrument OMAP, and the Sigma Design EM8400 series are all examples of low-power embedded media chip-sets which share many similarities in system design and bus structure. Another key component in the architecture model is the system bus and external memory. Memory bandwidth is a great challenge for systems to process streaming data in real-time. To insure sufficient bandwidth, hardware designers usually provide multiple buses in the system, each having different bus speeds and different protocols. An external bus is used to interface to the largest off-chip memory system and other asynchronous memory-mapped devices. The external bus has a much longer physical length than other buses, and thus typically has much higher bus capacitance and greater power dissipation. The goal of the architectural model is to accurately model power dissipation in a complete system power model so that new power-efficient design.

The bus encoder model can be inserted either on the interface from the processor to the first level of the memory hierarchy or between any adjacent levels of the hierarchy to evaluate the bus encoding effects on power consumption. The model implements the main power oriented bus encoding techniques, namely Gray, Bus- Invert [12], T0, T0_BI, Dual_T0 and Dual_T0_BI. The encoding schemes can be applied to both data and address buses. The generator outputs are tightly dependent on the processor architecture. The current version of the stream generator model includes generic load/store RISC architecture. For our analysis, we considered a sub-set of a generic RISC instruction set, which is composed of three basic classes of instructions: Conditional Branch Instructions (B); Arithmetic-Logic or Data Processing Instructions (DP); Load/Store or Data Transfer Instructions (DT). The memory address spaces for data and instructions are separated. Basically, the sequence of memory addresses is generated by assigning the percentage of the different classes of instructions with respect to the total number of generating addresses. The address sequence is generated by the processor by varying: the format and the execution frequency for each instruction class; the possible addressing modes for each instruction and the related execution frequency; the frequency to satisfy a conditional branch. All these parameters contribute to modify the spatial and temporal locality of memory references [16]. The address bus from the processor to the memory subsystem contains a memory address corresponding to a datum or an instruction. The address stream characteristics can be assigned depending on the desired level of the spatial and temporal locality. The bidirectional data bus can carry two different types of information: instructions and data. The type of instruction contained at a given memory address depends on the parameters set for the address bus model. Meanwhile, the datum contained in the memory address can be generated either probabilistically or pseudo-random. In the first case, the model is based on a medium average model of the first order, MA (1), to take into consideration the correlation between two consecutive data words, responsible for the switching activity on the system-bus.

The power estimation unit:

A power estimation framework improved from is proposed. The power estimation framework is divided into two individual processes. One is the systemc simulation environment, and the other is the power estimation unit. These two processes can be operated in parallel, while communicating with each other by a system FIFO. Therefore, the power consumption can be calculated by the power estimation unit during the systemc simulation. There are differences between different hardware components, just like each power. Model in the hardware component. To adapt different hardware consumptions, the power models are built into the power estimation unit. These units are defined by users, which makes the power estimation unit more flexible. The power estimation unit will collect the needed information by the power models to calculate the power consumptions. In a systemc simulation environment, it has many components which are included in a common embedded SOC system. For instance, CPU, BUS, the main memory, and the Application Specific Integrated Circuit (ASIC), etc. Also each component in the embedded SOC system has different power consumption factors. The power models have to be different to adapt to different hardware component. The power estimation unit has two different functions power information collector and power consumption calculator. The hardware's power consumption change when the input or the process has changed. The power estimation unit collects the needed information with the mapped power model during systemc simulation. The information is then transferred to the power calculation unit. When the power estimation units are separated with the systemc simulation to achieve two advantages. The simulation and power estimation are separated, which means that the simulation will not be significantly delayed while collecting the power information with the power estimation unit.

Power estimation of a CPU:

The CPU power estimator calculates ARM7TDMI power consumption for each instruction, based on the trace from *Trace Converter*. In the ARM7 programming model, the power variation is dependent on instruction-level *energy-sensitive factors* such as instruction fetch addresses, opcodes (operations), register encoding, data fetch addresses, immediate operands, and so on .

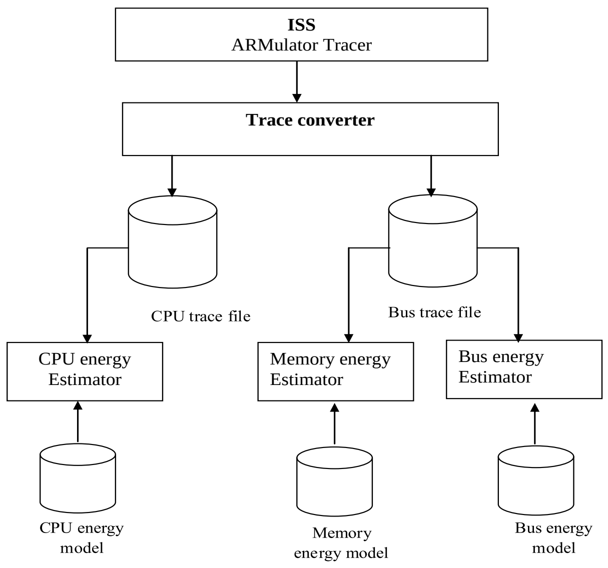


Fig 20: Architecture of the power estimator

More specifically, Hamming distance between two adjacent instructions and the number of one's in the encoding of an instruction are the two major basis of power consumption [3]. Lectures deliver theoretical background of the ARM7TDMI power consumption model. We guide the students to implement the instruction-level estimator by adding the power consumption of each pipeline stage. The power coefficients of the energy-sensitive factors are supplied in the course materials. All the coefficients are the measurable results of an ARM7TDMI test chip, with a cycle-accurate energy measurement technique. Students may also capture the coefficients of the ARM7TDMI by themselves using SEE.

System-Level Power-Aware Design for Real-Time Systems:

We assume familiarity with common concepts in real-time systems; for detailed information, the reader is encouraged to consult [20]. System-level power-aware design in real-time systems is a relatively new research area. Low-power had become an important parameter at the higher layers of system design by the mid - 1990's. Most of the new system level low-power techniques initially targeted general-purpose computing systems. However, it soon became apparent that real-time systems present unique challenges and opportunities for system-level low-power design as demonstrated next [26].

❖Real-time systems are usually severely power constrained. In particular, space borne and multimedia systems are typically battery-operated and therefore have a limited *energy budget*. Real-time systems are also relatively more time-constrained compared to general-purpose systems. Therefore, the challenge is to save power while satisfying temporal guarantees.

❖Some real-time applications such as avionics, robotics and deep space missions require systems with small form factors, which in turn mandate low heat dissipation. Since heat is a byproduct of power dissipation, low-power system-design ensures a more reliable system by limiting he heat produced.

❖Real-time systems are typically over-designed to ensure that the temporal deadline guarantees are still met even if all tasks take up their worst-case execution time (WCET) to finish. Since, in the average case, tasks do not run until their WCET, this is very energy inefficient. System-level techniques can decrease this power dissipation through the use of power-aware task scheduling algorithms while preserving the temporal guarantees.

❖Real-time systems are designed to be fault-tolerant. Fault tolerance ensures reliability through replication of software/hardware resources. However, brute replication in turn, causes high power dissipation. System-level low power techniques manage replication resources judiciously to reduce the required power.

System-level power-aware research in real-time systems is still in its infancy. While there is intense activity in the area, most initial research is concentrated in adding power awareness as a second-tier design goal which complements the more traditional real-time design goals. According to this approach, the system is first optimized subject to traditional real-time design constraints like timing and reliability. More often than not, an additional optimization step subject to power-aware design constraints is then piggybacked to this design. We believe that power-awareness should be one of the primary design goals for real-time systems, integrated in the design process at all levels, simultaneously coexisting with the traditional real-time design objectives. This requires a radical rethinking of the design methods as well as the definition of new metrics, a vision that is already becoming more ingrained in the research community [22].

References:

[1]Madhu Saravana Sibi Govindan, Stephen Keckler, Sani Nassif∗, Emrah Acar∗, "A Temperature-Aware Power Estimation Methodology", September 2, 2008

[2]Fuming Sun, Haiyang Wang, Fei Fu, and XiaoyingLi "Survey of FPGA Low Power Design**",** International Conference on Intelligent Control and Information Processing August 13-15, 2010 - Dalian, China

[3] Bikash Chandra Rout, Dr. Kamala KantaMahapatra, "Multilevel Power Estimation Of VLSI Circuits Using Efficient Algorithms", Master Thesis Dept. Elec. Eng., National Central University Rourkela, 2011.

[4]DimitriKagaris&ThemistoklisHaniotakis Transistor-Level Synthesis for Low-Power Applications" Department of Electrical and Computer Engineering Proceedings of the 8th International Symposium on 2007

[5]AdrielZiesemer, CristianoLazzari, RicardoReis "Transistor-Level automatic layout generator for non complementary CMOS cells", 2007

[6] Mostafa E. A. Ibrahim, [1, 2, Markus] Rupp (EURASIP Member), [2] and Hossam A. H. Fahmy[3], "A Precise High-Level Power Consumption Model for Embedded Systems Software", Accepted 11 August 2010

[7] A. Davoodi and A. Srivastava, "Probabilistic Dual-Vth Leakage Optimization under Variability," in Proc. International Symp. Low Power Electronics and Design, 2005, pp. 143–168

[8] D. Soudris, G. Theodoridis, K. Katis, A. Thanailakis, and C.E.Goutis "Structure and Techniques of the Low-Power Design Flow", 2000

[9]Toru Fujimura and Shigetoshi Nakatake School of Environmental Engineering, University of Kitakyushu "Transistor-Level programmable MOS analog IC with body biasing", 2008.

[10] SalarAlipour, BabakHidaji and Amir Sabbagh Pour "Circuit level, Static Power, and Logic Level Power Analyses" Dept. of Computer Science and Engineering Chalmers University of Technology,2010

[11] S.Theoharis, G.Theodoridis, P.Merakos and C.Goutis "Accurate data path models for fast RT-level power estimation" IEEE proc.-Compute. Digit.Tech, Vol.147, No.4, July 2000

[12] Hirofumi Kawauchi, Ittetsu Taniguchi, and Masahiro Fukui "A New Approach for Accurate RTL Power Macro-Modeling" journal of semiconductor technology and science, vol.10, NO.1, MARCH, 2010

[13] M. Bruno, A. Macii and M. Poncino "RTL power estimation in an HDL-based design flow" IEE Proc.-Compute. Digit. Tech., Vol. 152, No. 6, November 2005.

[14] MassoudPedram and Afshin Abdollahi "Low Power RT-Level Synthesis Techniques" Dept. of Electrical Engineering University of Southern California

[15]Yiwei Zhang, GeZhang, "Fast Gate-level Simulation and Power Analysis For High Performance Microprocessor" Department of Information and Technology University of International Relations Beijing, China, International Conference 2009

[16] William Fornaciari, Donatella Sciuto, CristinaSilvano,"Power Estimation of System-Level Buses for Microprocessor-Based Architectures: A Case Study"1999.

[17] Sandro Penolazzi Doctoral "A System-Level Framework for Energy and Performance Estimation of System-on-Chip Architectures" Electronic and Computer Systems Stockholm, Sweden 2011.

[18]Matthias Gries "Methods for evaluating and covering the design space during early design development" CAD-Group, Electronics Research Laboratory, University of California at Berkeley the VLSI journal 38 (2004)

[19] Saumya Chandra, Kanishka Lahiri, Anand Raghunathan and Sujit Dey, "Variation-Tolerant Dynamic Power Management at the System-Level" Dept. of ECE, University of California, San Diego NEC Laboratories America, Princeton, NJ

[20] Naehyuck Chang, Hyeonmin Lim, Kyungsoo Lee, Youngjin Cho, HyungGyu Lee and Hojun Shim "Graduate Class For System Level Low Power Design" School of CSE, Seoul National University, Korea, June 2005

[21] KeNing "System-level memory power and performance optimization for system-on-a-chip embedded systems", The Department of Electrical and Computer Engineering, January 01, 2008

[22] Osman S. Unsal *Member, IEEE,* and Israel Koren, *Fellow, IEEE* "System-Level Power-Aware Design Techniques in Real-Time Systems" Proceedings of the IEEE, VOL. 91, NO.7, JULY 2003

[23] Ankur Agarwal, Eduardo Fernandez, "System Level Power Management for Embedded RTOS: An Object Oriented Approach" Department of Computer Science and Engineering Florida Atlantic University Boca Raton, FL 33431, USA

[24] Auburn, Alabama "Process-Variation-Resistant Dynamic Power Optimization for VLSI Circuits" May 11, 2006

[25]*Vincent Nijman* "Effect of behavioural changes due to habitat disturbance on density estimation of rain forest vertebrates, as illustrated by gibbons (primates: hylobatidae)"

[26] Marcello Lajolo, Anand Raghunathan, Sujit Dey "Efficient Power Co-Estimation Techniques for System-on-Chip Design", In Proc. Design Automation and Test Europe (DATE),March 2000

[27] C. Brandolese, A code sign approach to software power estimation for embedded systems, Ph.D. disseration, Politecnico di Milano, Institute of Electronics and Information, 2000.

[28]Q. Wang and S. B. K. Vrudhula, "On Short Circuit Power Estimation of CMOS Inverters," Proc. IEEE International Conference on Computer Design, pp. 70–75, Oct. 1998

[29] Julien Lamoureux and Wayne Luk "An Overview of Low-Power Techniques for Field-Programmable Gate Arrays", NASA/ESA Conference on Adaptive Hardware and Systems IEEE 2008.

[30] Y.A. Durrani, T. Riesgo, "Architectural Power Analysis for Intellectal Property-Baesd Digital Systen", Journal of Low Power Electronics, pp. 271-279(9), Vol.3, No.3, 2007.

[31] Joel Coburn, Srivaths Ravi, and Anand Raghunathan "Hardware Accelerated Power Estimation", 2005 IEEE

[32] Y. A. Durrani, T.Riesgo, F.Machado "Statistical Power Estimation for Register Transfer Level", International Conference Department of Microelectronics & Computer Science 2006

[33] Y.A. Durrani, T. Riesgo, "LUT-Based Power Macromodeling Technique for DSP Architectures", In Proceedings for IEEE International conference on Electronics, Circuits and System, 2007.

[34] Dirk Rabe Wolfgang Nebel "New Approach in Gate-Level Glitch Modeling", 1996 IEEE

[35]S. Ravi, A. Raghunathan, and S. Chakradhar, "Efficient RTL power estimation for large designs," in Proc. Int. Conf. VLSI Design, Jan. 2003

[36] B.Arts, N.Eng, M.J.M.Heijligers, etal., "Statistical power estimation of behavioral descriptions," in Proceedings of the 13th International Work shop on Integrated Circuit and System Design, Power and Timing Modeling, Optimization and Simulation (PAT MOS ' 03), vol. 2799 of Lecture Notes in Computer Science , pp. 197–207, Springer, Torino, Italy, September 2003.

[37] R. Zafalon, M. Rossello, E. Macii, and M. Poncino, "Power macro modeling for a high quality RT-level power estimation," in Proceedings of the 1st IEEE International Symposium on Quality of Electronic Design (ISQED '00), pp. 59–63, San Jose, Calif, USA, March 2000

[38] Robertas Dama sevicius and Vytautas Stuikys, "Estimation of Power Consumption at Behavioral Modeling Level Using SystemC", Department of Software Engineering, Faculty of Informatics, Kaunas University of Technology, Accepted 6 May 2007

[39] Nikhil Bansal Kanishka Lahiri Anand Raghunathan Srimat T. Chakradhar, “Power Monitors: A Framework for System-Level Power Estimation Using Heterogeneous Power Models”, Proceedings of the 18th International Conference on VLSI Design in 2005

[40] L.Zhong, S. Ravi, A. Raghunathan, and N. K. Jha, “Power Estimation for Cycle-Accurate Functional Descriptions of Hardware,” in *Proc. Int. Conf. Computer-Aided Design*, 2004.

[41] S. Gupta and F. N. Najm, “Power modeling for high level power estimation,” *IEEE Transactions on VLSI Systems*,vol. 8, no. 1, pp. 18–29, Feb. 2000. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/tvlsi00-gupta.pdf

[42] Kavel M. Buyuksahin, Farid N. Najm, *Fellow* ,“Early Power Estimation for VLSI Circuits”,vol 24,pg 1076-1088 in 2005

[43] S. Gupta and F. N. Najm, “Analytical models for RTL power estimation of combinational and sequential circuits,” *IEEE Transactions on Computer-Aided Design*, vol. 19, no. 7, pp. 808–814, July 2000. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/tcad00-gupta.pdf

[44] J. A. Darringer,R. A. Bergamaschi, S. Bhattacharya, D. Brand, A. Herkersdorf, J. K. Morrell,I. I. Nair,P. Sagmeister,Y. Shin, “Early analysis tools for system-on-achip design”, IBM J. RES. & DEV. VOL. 46 NO. 6 NOVEMBER 2002

[45] R. Damasevicius, “Estimation of design characteristics at RTL modeling level using systemC,” *Information Technology and Control*, vol. 35, no. 2, pp. 117–123, 2006

[46] N.D. Zervas[a],*, G. Theodoridis[b], D. Soudris[c], “Behavioral-level event-driven power management for DECT digital receivers”, Microelectronics Journal 36 (2005) 163–172.