Consumption modelization of a DSP and automatic consmption/performance estimation of a C code executed by this DSP

GUITTON-OUHAMOU Patricia, BELLEUDY Cécile, AUGUIN Michel
Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis
Les Algorithmes-bat. Euclide
2000, route des Lucioles-BP 121, 06903 Sophia-Antipolis Cedex
guitton@i3s.unice.fr, belleudy@i3s.unice.fr, auguin@i3s.unice.fr

***Abstract**:*
The great development of numeric systems (mobile telephone, personal computers, multimedia terminal...) increases every day the power consumption. The development of mobile telephone and other embedded systems puts the question of the battery's life. Indeed, the life of the battery is a critical parameter, and batteries create pollution. It is necessary to reduce this consumption in this domain by every ways.
Some studies have shown that it is possible to reduce about 40% the consumed energy by rewriting some parts of the source code (software optimization) [1]. Optimization at system level can achieve a gain about 58% by reducing the supply voltage or the frequency (compromise speed/consumption) [2]. To optimize the code by rewriting some parts of the source code, it is necessary to detect those parts. So a power estimator coupling with a performance estimator would be an ideal tool. In this article, we show the integration of the consumption criteria in a performance estimator tool that already exists. So estimations of performance and of the consumed power can be realized in the same time to allow optimizations.

Key-words: *power estimation of C code, consumption model of a DSP, low-power code for DSP, performance/energy estimator for a C code, automatic code estimation.*

1•Introduction

The great development of numeric systems (mobile telephone, personal computers, multimedia terminal...) increase every day the power consumption. Indeed, the global consumption was about 160 MWatts in 1992 and was achieving about 9000 MWatts in 2001[3].It is necessary to reduce the consumption.Some studies have shown that it is possible to reduce about 40% the consumed energy by rewriting some parts of the source code (software optimization) [1]. Optimization at system level can achieve a gain about 58% by reducing the supply voltage or the frequency (compromise speed/ consumption) [2]. As the telecommunication applications always use a core of signal treatment processor, Digital Signal Processor, our study uses the consumption model of a DSP, DSP OAK^{TM} [4] of Phillips.
The telecommunication applications are written in C and compilers are ineffective for the DSP. That's why the laboratory and Phillips Semiconductors Sophia have built a tool which estimates the performance of the code, VESTIM [5]. It gives an estimation of an optimized code in performance, so in cycles. The goal is then to introduce the consumption criteria in this tool. It could estimate the performance and the consumption collectively. Indeed, to optimize the code by rewriting some parts of the source code to reduce the consumption, it is necessary to detect those parts. So a power estimator coupling with a performance estimator would be an ideal tool. To realize the integration of the consumption criteria in this performance estimator tool, it is necessary to have a consumption model.So, in first, a consumption model has been extracted. Our consumption model is very accurate, the worst error is about 8.8%. Some writing rules have been deduced to optimize the consumption, a gain of 14% is achieved for a FIR.
In this paper, the performance estimator tool, VESTIM is presented, then the consumption model of the DSP OAK^{TM} is briefly presented, and then its implementation in this tool. This last part constitutes the innovative part and is a preparatory study.
Indeed, we can mention some related works as Nathalie Julien[6]. They estimate the consumption of the C code for the TMS320C6201. A lot of parameters have to be computed with a lot of test sequences and is not easy to utilize.
Furthermore, there are works of Tiwari and al. [7] but their work is at assembly level.
The novelty of our work is to estimate in performance and in energy a C code without compiling it in assembly.

2•VESTIM: performance estimator for DSPs

VESTIM is a performance estimation tool: it estimates a C code from a compiler and gives an estimation of an optimized code.

a•Goal: To program a DSP is more and more difficult because of the complexification of the applications and of the architectures. It is necessary to build compilers able to supply an assembly code **since a C code**. The difficulty is the ineffectiveness of actually compilers because of the DSP specific and heterogeneous architectures. VESTIM allows to detect the parts of the code to optimize.

b•Estimation principles: VESTIM decomposes the program in some base blocs. A base bloc is a part of the program. The only entrance is the first instruction and the only out is the latest instruction.

Each base bloc presents an execution time supplies by the formula: Texe=xi ci, with ci, the cost in cycle number for each base bloc and xi is the number of executions of each bloc.

The execution number xi is achieved by executing the program on a host processor (pentium for instance, because its execution is faster). Test sequences of data are used. It is to avoid the parts of code depending on the values of data. For parts of test independent of data, a profiling of the source code gives xi. This avoids simulations. So the xi are known.

The cycle number of each base bloc ci is computed. An addition of the cost of each instruction in a base bloc is done.

The application representation in RTL (Register Transfer Language describes the processor) which is an intermediate code for compilers GNU (and gives the decomposition in base bloc) allows to build a CFG (Control Flow Graph, figure 1).

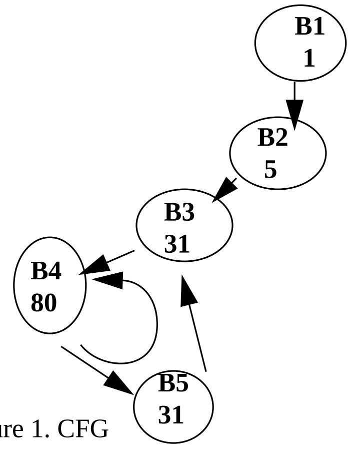


Figure 1. CFG

Each knot Bi represents a base bloc with annotations (priorities); each edge represents dependencies.

To estimate the number of cycles ci of each base bloc, a DAG (Data Acyclic Graph, figure 2) is built with the RTL representation.

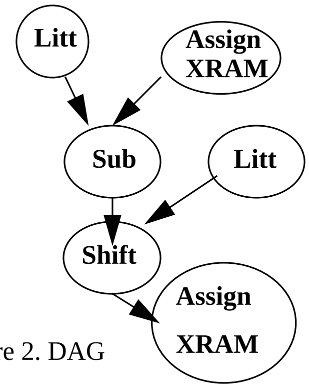


Figure 2. DAG

Each knot represents an operation; each edge represents precedencies and data dependencies. Names indicate characteristics. For example, Litt is an operation with a constant integer.

2•1•Overview of VESTIM

Let see a global diagram of VESTIM.

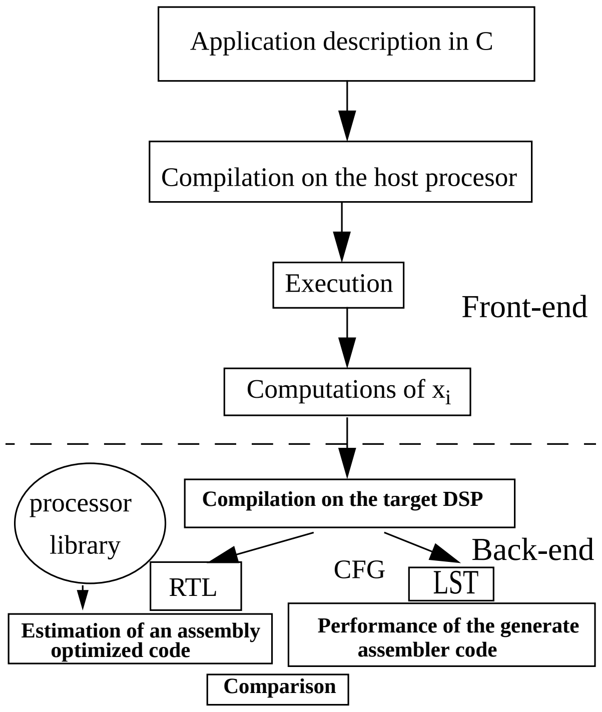

Application description in C
Compilation on the host procesor
Execution
Front-end
Computations of x_i
processor library
Compilation on the target DSP
Back-end
RTL
CFG
LST
Estimation of an assembly optimized code
Performance of the generate assembler code
Comparison

Figure 3. VESTIM

2•2•VESTIM, step by step

Let us explain the phases of VESTIM (Figure3):
Firstly, LST is an assembly code generates by a C compiler.

Let us give some details about the different parts of Figure3 which describes VESTIM.

- Front-end: Set of phases dependent on the source code are independent of the target processor. At this step a test sequence is executed on a host processor to obtain the xi.
- Back-end: Set of phases dependent on the target processor and on the intermediate language. The generated code and the optimized code are estimated.

Now, let us see libraries needed by VESTIM.

2•3•Processor library:

To take into account the consumption it is necessary to fill-up the processor library. Let us see how the processor is described.
A file contains the processor name (Proc_Name), the memories and registers resources (Res, Res...)operator units and addressing modes (RMi (indirect addressing mode), RMsd (short direct addressing mode))...
Example:
ADD:
CV, RMi; ALU; ACC
UT_Free = ALL - (MEM)
REG_Free = ALL - (RI)
Nb_Cycle = 1
Cond = No
In first, it is the operator name, ADD, in entrance, the register CV (Constant Value), RMI indicates that the addressing mode is indirect. In out, ACC means that out are in accumulators. The unit activated is the ALU. Constraints for memory use are UT_FREE=ALL-(MEM), so no constraints except that the memory is not available. For the registers, REG_FREE=ALL- - (RI) means that all the registers are free except the R registers. Then the number cycle is presented and no condition to execute the instruction are necessary: Cond= No.

2•4•Operations scheduling:

As the intermediate RTL description, more oriented RISC, an intermediate description oriented DSP is built by applying adapted writing rules. An operation scheduling is so built. An instruction can contain some operations.
Tasks are put in order of priority. Priorities are determined by deadlines and dependencies. When a task is scheduled, these successors are integrated in the list. This step takes into account the parallelism: instructions executing in parallel at the same cycle are put together.

2•5•OUTPUT of VESTIM: Estimation

A task (described in the CFG) is implemented when a corresponding operation in the target processor description is met or when it is met in the scheduling list. Then the conditional field is checking.
VESTIM supplies a scheduling table of operations on the architectures units.
For the previous DAG the reservation table is:

	cycle 1	**cycle 2**	**cycle 3**	**cycle 4**
ALU		Sub		
Barrel Shifter			Shift	
MEM		Ltd		X
CG (constant Generator)	X		X	
ACCU	X	X	X	
RI		X		
SV			X	X

3•DSP OAK+ (Phillips) description

The OAK DSP^{TM} processor [4] is a CMOS 16 bits fixed point DSP processor. It consists briefly of the MPU, the ALU, the barrel shifter, the exponent logic unit, the data address arithmetic unit, four 36-bit accumulators and six data/address registers r0...r5 (16 bits), two internal memories X and Y. Busses connecting the registers ri are preloaded to 1. Its description in VESTIM implies to simplify the architecture description and its resources description.

4•Implementation of the DSP Oak consumption model in VESTIM

4•1•The consumption models[8].

Models are the most possible by functional units as works VESTIM. Because of heterogeneous architectures, DSPs may have very complex behaviours in consumption. In addition, their many possibilities of datapath lead to developp some measures methods taking into account the units of the architecture. Furtheremore, the instruction set of a DSP is very near to the functional units, that means that parallelism instructions is obtain by activing paralell units. For example, we can mention the address generators, and computation unit in an multiply/accumulation instruction.

Comapred to generic processors, DSP processors are different. For example, registers are often dedicated and dependant on functional untis. They contain some operators and some specific adressing modes that speed up the execution of some treatment of the signal. Many DSPs are fixed and data width is adaped for some algorithms as audio algorithm.

For most of DSPs , it is preferable to dispose of a parallel memory rather than a speed hierarchic memory with caches.

As VESTIM exploits the operations schedulings at functional level, we have decided to modelize consumption according to datapath. For each VESTIM instruction, it is so possible to compute a consumption estimation by additionning contribution of each unit. So the corresponding consumption modelization of the OAK DSP could be at assembly level. Moreover, as the consumption behavior is very complex, it is preferable to modelize at the assembly level. So, let us describe the meaurement methods.

In order to compute the power consumption, we use the board-based measurement method developed in [9].The average power is given by $P=I*V_{dd}$ where I is the average current and Vdd the supply voltage. The energy E is given by: E=P*T where T is the execution time of the program. The basic idea is to measure the consumed current by inserting an ammeter between the power supply and the CPU (the power supply connection to the CPU must be isolated from the reminder of the system). Evaluation of the energy consumption of an assembly program requires to differentiate two types of consumptions:

- the base cost of an instruction provided by the consumption of the execution of one instruction,
- the inter-instruction (or overhead) cost due to the changes in the data path activated by two successive instructions.

So, the principe is to measure the current for each instruction, we create a test program that contains initialization of the system and a loop combining the instructions repeated 200 times. This number has been determined so that the loop instructions does not modify the measured current value, that means to obtain an average measure only due to the instruction itself, it is to neglect the effects due to instructions managing the loop. To measure the overhead, the test program contains 200 pairs of instructions.

In a first approach, these measurements must be realized for each instruction, each value and each addressing mode of the operands. Since this work is too exhaustive, classes of instructions are formed. Two instructions in a class activate the same data path in the DSP. For example, operations executed by the ALU and using the same addressing modes for the operands belong to the same class.

The evaluation of the power consumption of a program is calculated by [1,4]:

$$E = \sum_{i}(R_i \times N_i) + \sum_{i,j}(O_{(i,j)} \times N_{(i,j)}) + \sum_{ki}(Ek)$$

Where:

O(i,j): is the overhead cost due to execution of instruction i followed by the instruction j;

Ri: the base cost of instruction i;

Ni: number of cycles to execute the instruction i.

Ek represents the consumption due to lost cycles, for example, cache and pipeline stalls.

Finally, it is so consider the instruction set, different data and addressing modes.

To realize these measures, a test board is necessary. This test board has to allow to measure the consumption of the DSP core, of the output/input of the circuit and of the memories.

Considering the architectural characteristics, two sources of variation were identified:

a• the preload of the busses: the bits equal to 0 consume more than those equal to 1.

This preload allows a faster clock rate. Indeed, forcing busses to the logical zero level is faster than forcing them to the logical level 1. The preload of the busses is activated only when operands use the data/address registers of the DSP. To evaluate the effect of the preload of busses, the current values are measured for an instruction executed first with accumulators input operands and next with data/address register input operands. The difference between these two values represents the consumption due to the preloaded bus activity. We notice that in the worst case, the current can double when data/address register operand are used. The preload of busses is an important factor of consumption for this processor. In this study, we made vary the number of bits equals to 0 in the input operands and their position.

b•the switching activity:

Another part of the consumption is due to the transitions of the bits, varying at each cycle from 0 to 1 and conversely. This switching activity results from the charges and discharges of the capacities of transistors, that all put together corresponds to the capacity of the circuit, giving a power consumption proportional to: CV^2 where V is the supply voltage. Measurements lead to conclude that the consumed power of the DSP depends mainly on the switching activity of the signals. Let a be the average number of transitions by cycle. The power Pswitching consumed by a unit of the power with a

clock fclk is [10,11]: Pswitching= α *fclk*(C*V_{dd}^2).
The consumption of energy generated by the execution of an application for all the units of the processor[10,11] is:

$$P_{switching} = \sum_{i}^{n}\left(\alpha_i \cdot CiV^2 \cdot f_{clck}\right)$$ with:

- n, the number of units of the processor;
- α_i average number of bit transitions per cycle of the its unit;
- Ci, out capacity of the considered unit.

However the out capacity of each unit is unknown. By measuring the current, we can estimate the power induced by the switching activity by changing the number of switching bits in the same instruction and the same addressing mode. The influence of their position was also tested.
In the worst case, we notice that the current may be multiplied by 1.5 when all switching bits change with regard to no change of bit (not to be influenced by the preload of busses, operands are accumulators).
c•The consumption model
In order to take into account these consumption sources the preloaded busses and switching activity, the following instruction current model is:
$I = I_{cst} + I_{preloaded\ bus} + I_{switching\ activity}$ (1), where I_{cst} is the current measured with no switching activity. The two other components can double the value current.
We present here only the consumption model for the ALU:
$I_{preloaded\ bus} = N*I_{bus\ unit} + I_{extension\ of\ sign}$, where: N is the number of bits equal to 0 in the input data.
$I_{bus\ unit}$: elementary current (by bit) due the preload of the busses.
$I_{extension\ of\ sign:}$ if the 15th bit of the input operand is equal to 0, the extension of sign entails the following 20 bits to 0. If the 15th bit is equal to 1, I extension of sign = 0.
These basic currents are measured as follows:
-$I_{bus\ unit}$= (Iand r1=0,a0 - Iand r1 =7FFF,a0)/15.
second operand is equal to 0 so that there is no switching activity.
-$I_{extension\ of\ sign}$:
If the 15th bit is equal to 0, Iextension of sign = (Iand r1=7FFF,a0 - Iand r1 = FFF7,a0)
Else Iextension of sign =0.
- $I_{switching\ activity}$ = (Ixor a1=1,a0 - Ixor a1=0,a0.).
This model is applicable for arithmetic and logic instructions executed by the ALU such as add, sub, cmp, and, or, xor.... (instructions of the class untitled ALU).
Accuracy of our model:
For these instructions, the estimated values of the consumption are compared with the real values:
- for logic instructions (and, or, xor), the worst case reveals an error of 2.5%.
- for arithmetic instructions, the error is 6% in the worst case.
We detail no more, there are other models for the different units, with the same consumption sources, so it doesn't bring more [8].
If the switching activity is negligible, the consumption is a constant, else we have to take into account the values of data.

Writing rules deduced:
Addressing modes have a significant impact on the current consumed by an instruction. There for, it is preferable to use the following resources, we examine only the ALU, but it's the same way for all the units:
Addressing modes have a significant impact on the current consumed by an instruction. There for, it is preferable to use the following resources, we examine only the ALU, but it's the same way for all the units:
-For the ALU
The source operands and addressing modes that lead to reduce the power consumption are:
- accumulators and registers a0, a1, p,
-short direct, and indirect addressing mode.

Using the differences of the overheads:
In a general approach, higher power savings are obtained for successions of instructions belonging to the same class then instructions belonging to different classes but with the same addressing mode.
Particularly:
- some pairs of instructions have a low power consumption, as for example the pair mov-mpy or mov-add.
-In contrast, it would be preferable to avoid mpy-add

d•Implementation
Addressing mode and the working unit are mainly to determine the base cost of an instruction. Indeed, Icst is the consumption without switching activity, it is determined by the addressing mode and the instruction itself. The scheduling table, output of VESTIM, supplies the addressing mode and the working unit, so a first consumption estimation is obtained: Icst.
So it is necessary to integrate in the processor library the consumption of each instruction and the overhead.
For the moment, we have modified VESTIM to take into account the consumption of each instruction and of an average overhead.

Reading the scheduling table, the algorithm computes the consumption with the library. So implementation gives us some average values of consumption for a C code.

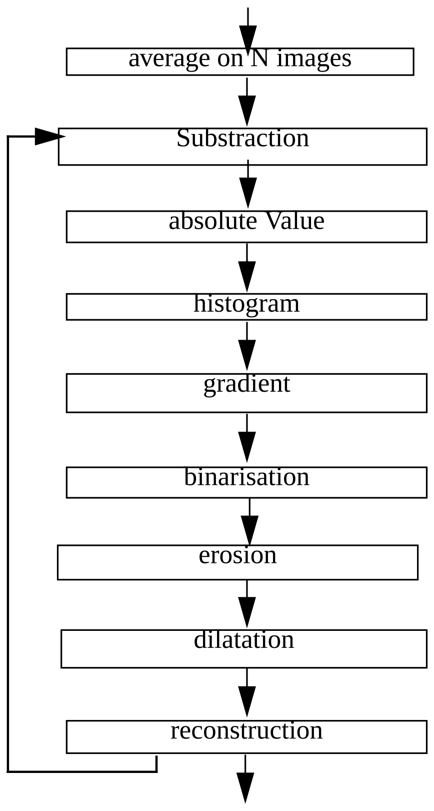
5•Results:

We present here some results.
We have considered an embedded video application. This application treats images for an embedded camera to detect motions on a fixed picture. The time constraint is to treat 25 images per second. All given results are fictive values because Philips Semiconductors does not allow us to give real values. First, let us describe the tested application.
The application considers several pictures. The first one is the background picture. The next pictures are utilized to compare at the first picture to detect the motion. Let us see the algorithm step by step:

- a parametrically average; its goal is to limit the impact of noise signal;
- a substraction of images to distinguish motion parts of fixed parts. In fact, the algorithm realized the difference pixel to pixel between two pictures. Fixed parts are in black and other in grey.
- a binarise operation to isolate moving objects,
- a treatment called "morphological" that filters picture to erase isolated points. The algorithm transforms a white pixel in a black pixel if all neighboring pixels are black.This treatment eliminates noise. Therefore it is necessary to realized previously some steps as
- erosion,
- dilatation and
- reconstruction.
  Erosion eliminates noise and some details of moving objects. Then it is necessary to operate a dilatation and a reconstruction to avoid too much information losses.

In summary, let us see the figure4:

Figure4.video application



We have estimated the consumption for each step of this application. Values are not real because Philips Semiconductors does not allow us to give real values. As shows the followed table, we obtain the number of cycles and the power for each task.

tasks	cycles	power (mW)
parametrically average/substraction/absolute	130082	111000000
histogram	178886	115000000
gradient	4	4000
binarisation	308874	171000000
erosion	564344	560000000
dilatation	1802333	758000000
reconstruction	196621	90000000

It is so possible to program and to know in the same time the cost of the code.

6•Optimization

To optimize the consumption by modifying the code, the method consists in using the writing rules presented previously. The successions of the instructions can be cheap in energy or not. These values are put in a library and are taken into account to build the scheduling list. So the CFG is now built with two priorities: the priority for the performance and the consumption priority. Consumption priority is computed according to the base cost.

The overhead cost must be considered carefully because it is very important. The overhead must be taken into account as annotation on the bow that is between two base blocs. It is not satisfying because some overheads between instructions are here neglected. Indeed base blocs regroup some instructions. How to take into account overheads between instructions? We have to modify the reservation table obtained since the DAG. For example, the table can be reviewed in considering the overheads between each successive instruction, in taking into account the performance.
So finally, an algorithm can be done that reviews the scheduling table and put other orders to optimize the total costs and compare with the performance.

A compromise has to be found because priorities are not obligatory compatibles, that means that the optimized solution for the consumption is not necessary the optimized performance solution. The compromise will be determined by tolerances.
To optimize solutions, a genetic algorithm can be implemented to propose the best solutions between the compromises.

7•Conclusion:

We have presented a consumption model of a DSP. Our method is reliable: the error is about 6% for the ALU. Then we present the step to implement the consumption model in a performance estimator tool, VESTIM. The same work has to be realized with the new generations of DSP processors. Libraries have to be updated, that represents an important work.

8•References:

[1] Tutorial: High level Power modeling, estimation and optimization; DAC 1997 Chair: Massoud Pedram; Organizers: Giovanni De Micheli, Massoud Pedram; Presenters: Enrico Macii, Massoud Pedram, Fabio Somenzi

[2] A. Pegatoquet, M. Auguin, L. Bianco, E. Gresset, “Rapid Development of optimized DSP Code from a High Level Description through Software Estimations”, 36th Design Automation Conference, June 19-24, New Orleans, Louisiana, USA, 1999.

[3] Vivek Tiwari, Deo Singh, “Power Challenge in the internet world”, Intel Corp., Tutorial of the 32 Annual International Symposium on Microarchitectures, November 1999.

[4] VVF3500 DSP Core User Manual, VLSI technology, Inc.

[5] Thesis, A. Pegatoquet, “Méthodes d’estimation de performance logicielle: application au développement rapide de code optimisé pour une classe de processeur DSP", Université de Nice Sophia-Antipolis (FRANCE), Octobre 1999.

[6] J. Laurent, N.Julien, E.Martin, «High Level Power Estimation for a DSP», in proceedings of Sophia Antipolis forum on MicroElectronics2000, October, 25-26 2000, p. 112-116.

[7]V. Tiwari, S. Malik and A. Wolfe, «Instruction level optimisation software», Journal of VLSI Signal Processing Systems, , p.139-154, April 1996.

[8] Guitton-Ouhamou Patricia, Belleudy Cécile, Auguin Michel “Power consumption Model for the DSP OAK Processor”, in SOC Design Methodologies edited by Michel Robert, Christian Piguet, Marie-Lise Flottes, Kluwer Academics Publishers, p217-228

[9]Vivek Tiwari, Sharad Malk and Andrew Wolfe, Mike Tien-chien Lee “Instruction level power analysis and optimization of software”, Journal of VLSI Signal Processing Systems, Vol. 13, No. 2, August 1996.

[10] Chingwei Yeh, Min-Cheng Chang, Shih-Chieh Chang, Wen-Bone Jone, “Gate-level Design exploiting Dual supply voltages for power-driven applications”, pp68-71 DAC 1999, New Orleans, LA, USA Newsletter, pp59-64, January 2001.

[11] Renu Mehra and Jan Rabaey “Behavioral Level Power Estimation and Exploration” Proceedings 1994 International Workshop on Low Power Design, Napa, CA, pp. 197-202, April 24-27, 1994.