Speeding up Power Estimation of Embedded Software

Akshaye Sama
Philips Semiconductors
Prof Holstlaan 4,Eindhoven
The Netherlands
akshaye.sama@philips.com

M Balakrishnan
IIT Delhi
Hauz Khas, New Delhi
India
mbala@cse.iitd.ernet.in

J F M Theeuwen
Philips Research
Prof Holstlaan 4,Eindhoven
The Netherlands
frans.theeuwen@philips.com

ABSTRACT

Power is increasingly becoming a design constraint for embedded systems. A processor is responsible for energy consumption on account of the software component of the embedded system. The power estimation of this component is a major concern due to the rising complexities of processors and the slow estimation tools. This work attempts to estimate the energy dissipation of the PR1900 [1] processor based on instruction set model with improved accuracy. The model is integrated in a simulation framework and validated. Over 200 times speedup has been obtained with average 1.4% loss in accuracy over gate level estimation. Analysis of the energy dissipated by the instruction vis a vis the processor architecture has been carried out and a substantial reduction in the measurement effort to build the processor energy model has been achieved.

1. INTRODUCTION

The essential tradeoff in power estimation is between estimation accuracy and simulation time. Increasing efforts are being made to improve the simulation speed without appreciable loss in estimation accuracies. The circuit and the gate level techniques[11, 6] are found to be very slow when evaluating power consumption of software executing on complex microprocessor designs. A lot of useful effort has been directed to architecture level macromodeling technique which is faster and offers an estimate at higher level but offers low accuracy and is useful only while exploring drastic changes in architecture for power reductions [8, 7, 2, 1, 10, 9, 3]. In the case of fixed architecture processors, this effort proves to be redundant and better estimates can be arrived at using instruction level energy model which was proposed in [14]. [12] provides the result that data dependency might not be negligible in some processors and also proposes a bus transition based model to tackle this dimension. Two speedup techniques for instruction level estimation are proposed in [5], namely energy caching and power macromodeling.

[1]PR1900 is a embedded microprocessor core from Philips Semiconductors which confirms to MIPS II ISA.

This paper presents an energy estimator for the PR1900 microprocessor. Improvements in the instruction level model are reported and also a technique for drastic reductions in measurements for building the energy model is described and evaluated.

The energy model and a brief introduction to the processor architecture is described first in section 2. The measurement procedure is described in section 3. Analysis of the power dissipation with respect to the architecture to obtain a new model is described in section 4. The experimental results are presented in section 5 followed by the conclusion and future directions in section 6.

2. ENERGY MODEL

The complexities of processors is hidden behind a relatively simple interface - the instruction set. Each instruction on execution activates some specified modules on the processor chip. This circuit activity is characteristic of the instruction and contributes to the energy cost of the instruction. This hypothesis of measuring the instruction execution energy forms the basis of the instruction level model. The components of this model are described next in the four subsections. Though our model is based on Tiwari and Malik's work [14] who have also proposed the same components, we have made substantial improvements in the circuit state overhead and introduced operand modeling as a separate component which can be accurately modelled.

2.1 Base energy cost

This is the primary component of the power model. It can be associated with the basic processing needed to execute the instruction. Physically, this component can be measured by repeated executions of the particular instruction. Since this energy is a characteristic of each instruction, it has to be measured for each one individually. Programs have to be made which cause repeated instruction execution and then average power has to be measured during this time. This average power consumed and the execution time of the instruction gives the base energy cost of the instruction. Special care is taken so that the stall cycle energies are not included in this component.

2.2 Circuit state overhead

During the transition from one instruction to the other, there is some overhead energy dissipated. This energy is dissipated due to the change in the processor state from the execution of one instruction to the other. The major cause for this overhead being the switching resulting from instruction opcode differences between the consecutive instructions, the control state changes and the passing of data between the pipeline stages. This overhead depends on the instruction pair between which the transition is occurring and has to be measured for each such possible pair of instructions.

Since the number of possible instruction pairs can be quite large even for small instruction sets, an approximation has to be used for the circuit state overhead. Previous approach has used the average state overhead value for this purpose. The state overhead forms a relatively small part of the total dissipation but, better approximations for this can increase the accuracy of the overall model.

In this work, we grouped the instructions based on their functionality and base costs. After which the intergroup and intragroup state overheads are measured. This approach was used because similar functionality and base costs usually indicate similar control state and opcode value. This is verified by measurements which show that average intragroup state overhead dissipation is 22% of the average intergroup state overhead.

2.3 Operand dependent dissipations

The energy consumed in the data-path during execution of an instruction also depends on the operands that flow through. The number of possibilities in the value of the operands does not permit us to measure energy for each. Hence, a model is required which can approximate this component adequately. Tiwari [14, 15] integrate this value in the base energy cost by using random operands during base cost measurements. A better approximation for this enhances the accuracy of the instruction level model.

The correlations in operands can be between the bits of the same operand and the bits of consecutive operands. The first one arises from the fact that in using 2's complement arithmetic, where the operand does not use the full range possible, the MSB's usually denote the sign of the number. The LSB's on the other hand change with even small values in the number and thus are better approximated as random. This correlation is captured using the Dual Bit Type (DBT) data model[8]. The second type of the correlation arises due to the reasoning that the operands of the same program do not show drastic changes in data. Hence, the switching in the operand decreases as we progress from the LSB's towards the MSB's. Using this, we can divide the operand into fields as shown in Fig-1. The bits in the same field assume the same value. The data words are generated manually so that the DBT conditions are obeyed and an activity decrease is achieved. These data words are then used in place of totally random operands for measurement of base instruction and state overhead energy costs.

2.4 Stall cycles

The types of stall cycles present in a processor depend on the architecture. Each type has to be identified and the corresponding energy has to be measured. All stall cycles such as cache misses, resource conflicts, buffer stalls etc are included in this component.

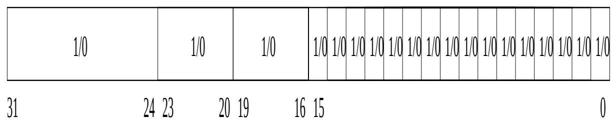

Figure 1: Data model

2.5 PR1900 processor architecture

PR1900[13] is a 32 bit MIPS-II[4] instruction set based processor. It consists of a 3 stage synchronous pipeline and is a fully static CMOS design. It is implemented on a 0.35 micron 5 metal layer process with an area of 6.5mm^2.

The memory hierarchy includes a 8KB 2-way set associative unified instruction and data cache. The write policy followed is write through with no write allocate. A single entry write buffer is provided to distribute bus load. The memory of the processor is divided into a cacheable and a non-cacheable type. The processor bypasses the cache in case of a non-cacheable memory access.

The possible stall cycles in this processor are cache miss cycles, cache hold cycles, write buffer stall cycles and load-store interlocks. Cache hold cycles are caused when the way-state has to be toggled due to a mismatch of the tags in the current way-state. Write buffer stalls are caused due to store type instructions when the write buffer is already full. In this case, the write buffer contents are first written to the memory before introducing the new contents in the write buffer. Load store interlock is essentially a RAW dependency on the memory which is caused by the write buffer. When the write buffer is full and a read miss or a non-cacheable load instruction is encountered, the contents of the buffer are first written to the memory before servicing the read request.

The cache hold cycles can be of two types depending on the gating of the integer unit. The integer clock is stopped for cache hold cycles in case a multicycle instruction is not being executed. Since the clock gating of the integer unit has a power lowering effect, the two types of hold cycles have to be catered to differently.

2.6 Overall energy model

The overall energy dissipated is the weighted sum of the various components. For the PR1900 case, it sums up to..

$$E_{program}=\Sigma_i(B_i * N_i)+\Sigma_{i,j}(O_{i,j} *N_{i,j})+N_{C.M}*E_{C.M}$$
$$+N_{W.B}*E_{W.B}+N_{C.H}*E_{C.H}+N_{L/S}*E_{L/S}$$

B_i:Base energy cost of instruction i

N_i:Number of times i is executed

$O_{i,j}$:Circuit state overhead energy for grp(i)->grp(j) transition

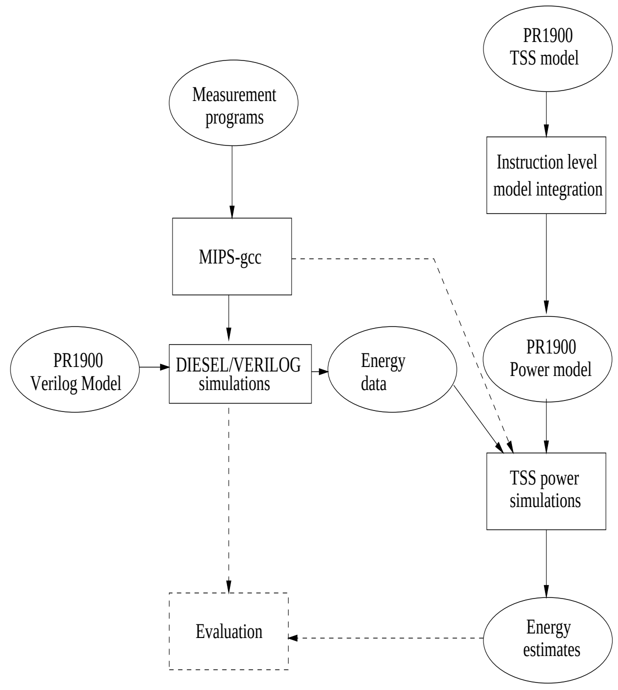

Figure 2: Model characterization

$N_{i,j}$:Number of times grp(i)->grp(j) takes place

$N_{C.M}$:Number of cache miss cycles introduced

$E_{C.M}$:Cache miss energy/cycle

$N_{W.B}$:Number of write buffer stall cycles introduced

$E_{W.B}$:Write buffer stall energy/cycle

$N_{C.H}$:Number of cache hold cycles

$E_{C.H}$:Cache hold cycle energy

$N_{L/S}$:Number of load store interlocks

$E_{L/S}$:Load store interlock energy

3. INSTRUCTION POWER MEASUREMENT

3.1 Gate level measurements

The energy numbers were calculated using DIESEL-verilog[2] gate level cosimulation. The design is described as a gate level netlist. With each component, a power view is stored during the library characterization process. The output load of each cell is first calculated from the fanout and the wire capacitance extracted from the layout. Using this information, the energy dissipated due to the transitions reported by the logic simulator is calculated.

3.2 Instruction energy measurement

The energy numbers have been derived from DIESEL gate level simulations as depicted in Fig-2. Programs were made in assembly and assembled using gcc(for MIPS II ISA). Verilog-XL logic simulator was used to report the transitions to DIESEL.

The base energy cost of the instructions is measured using a loop consisting of repetitions of the same instruction. The loop is executed twice. The energy dissipated during the second execution of the loop is considered for calculating the base cost since the first execution also includes the energy for the cache misses. The effect of operand variations is also captured in the base cost by providing the individual instructions with different operands generated as described earlier in section 2.3.

The circuit state overhead is measured using a similar technique but with the loop consisting of an alternating sequence of the two instructions. The base cost values are then subtracted appropriately to give the state overhead. The instructions were divided into 15 groups on the basis of functionality and base cost values. This overhead was then measured for four possible combinations for each possible group transition. An average value is then calculated for each transition.

For the cache miss, the energy and cycle difference between the two executions of the loop is calculated. Since the only difference between the two executions are the cache misses, these differences are contributed by it. The measurements for other stall cycles was done by manually isolating the interval in which these occurred and then triggering the energy simulations for it. In all these cases, appropriate averages were taken to get a better approximation.

4. ANALYSIS OF INSTRUCTION POWER DISSIPATION

The instruction level model is then related to the architecture of the processor so that prediction of some base energy costs is possible. The objective in doing this is to speed up the process of building up the instruction energy model as this would reduce the number of instructions for which measurements have to be carried out. The prediction is done by first deriving an analytical energy model for the architecture. This approach is based on the fact that each instruction execution consists of a number of sub tasks such as memory access and module activations. But, in most of the cases, the instruction execution takes similar flow through the pipeline. In a few cases, extra module activations are required or some module activations are supressed. The approach described here, isolates the energy associated with the instruction flow through the pipeline as a few stage based components. Also, some components are isolated for additional module activations. These component values are calculated from the instruction base energy consumption of a few instructions. Using these extracted values, base costs of some other instructions have been predicted and shown to be quite close to the measured values.

4.1 Model components

The model consists of stage based activation components and module activation components. Since no clock gating is

[2]DIESEL is a gate level power measurement tool used within Philips.

implemented within the integer unit and most of the instructions pass through the same stages of the pipeline, two stage based components are found to be enough. A usual execution goes through IF(Instruction Fetch), EX (execute) and WB(write Back) stages therefore, a component IF_EX_WB is isolated to account for this flow. Another component is isolated for the extension cycles that are added for multicycle instructions. This component IFI_EX_WBI is attributed to the idle IF and WB stages and the active EX stage. In addition to these two, 13 extra module activation energies are isolated. Examples include IMM(S) for signed extension for the immediate and Cache_L_W for cache and cache controller activation for loading a word. These 15 components are characterized using one instruction each.

4.2 Reduction in measurements

Using these, the base costs of some other instructions can be predicted by just adding the stage based and the module activations for that particular instruction. For example, a LW(load word- 2 cycle) instruction consists of both the stage based components along with IMM(S) and Cache_L_W module activations. Similarly, architectural information is used to decide which components form a part of the instruction execution and the corresponding components are then added to predict the base cost. The components are derived here for the case of a PR1900 processor but similar models can be derived easily for other processor architectures also.

5. EXPERIMENTAL RESULTS

5.1 Integration in simulation environment

TSS(Tool for System Simulation)[3] is a cycle level 'C' language based simulation environment. The energy estimator was integrated into it so that the energy of programs could be estimated along with the usual simulation.

The instruction level model estimates are first compared with the gate level simulation in Table-1. These comparisons are done for the same programs to study the gain in simulation speed and the corresponding loss in accuracy. Five programs were used, Bub_sort is a sorting program, Op_loop is a loop of a few arithmetic operations, IDCT is an inverse discrete cosine transform program, FFT is Fast fourier transform and Mat_mult is a matrix multiplication program. The program charachteristics are tabulated in the first few rows which show that the comparison programs cover a wide range in static(74 to 433) and dynamic(878 to 3018) instruction counts. Also, the constituent instructions covered the whole instruction set appropriately. The next three rows show that the simulation speed has increased between 240 and 380 times in all the cases and is expected to be much more for larger programs. Next important consideration is the average accuracy loss which is shown to be a low 1.4% with only one significant error of 5.7%.

In Table-1, the instruction type abbreviations are as- L-Load, S-Store, B-Branch, A-Add, Su-Subtract, Se-Set, M-Multiply and D-Divide.

5.2 Effect of instruction grouping

[3]TSS is a cycle accurate C language based simulation framework used within Philips.

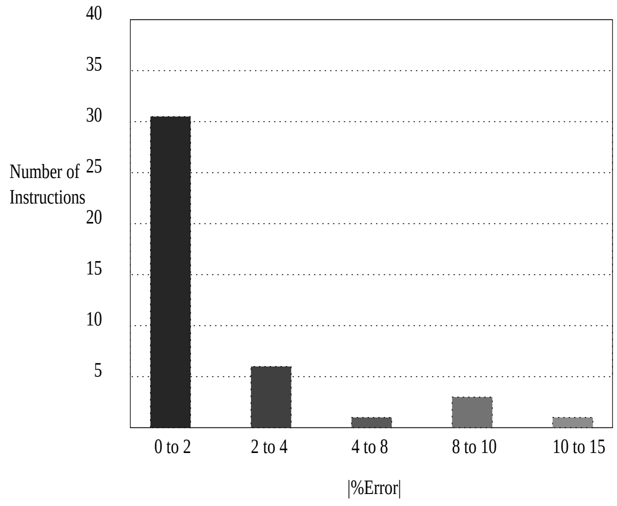

Figure 3: Error variations

The improvement obtained through instruction grouping for state overhead is tabulated in Table-2. The use of grouping approach instead of the average approximation lead to an accuracy gain of approximately 1.3%. The results also show that the grouping approximation estimate is nearer to the gate level estimate than the average approximation in all the example cases.

5.3 Prediction of instruction energy

Since the measurements are done at a much lower level(gate level), the efforts and time for the instruction level model characterization process is large. The method of predicting instruction base costs can reduce this effort. In this work, the method was applied on a subset of the total instruction set and 27 instruction base costs were predicted from 15 measured values. A measurement set reduction of more than 60%. The price to pay for this reduction is tabulated in Table-3 which is a small accuracy loss calculated to be less than 1% for all the example cases. The error in predicted values of instruction base costs for the instruction subset (42 instructions) is shown in Fig-3. This is a bar chart which groups the instructions according to the error range in which there base cost predictions fall. This also shows that 15 instructions are predicted with error within 2%.

6. CONCLUSIONS AND FUTURE WORK

The results show an increase in simulation speed of more than 240 times at the cost of an average accuracy loss of 1.4% with respect to gate level simulations. New accuracy enhancing measures of data modeling and grouping approach for state overhead are also tried and found to give estimates which are closer by approximately 1.3%.

A reduction of more than 60% in the measurement set for characterization process is obtained through the new architecture based model. This is achieved by relating the instruction power dissipation with the processor modules. Since the measurement process is done on a much lower level, it requires a lot of time and efforts. Also for the case of large instruction set processors, this reduction is important.

Program	Bub_sort	Op_loop	IDCT	FFT	Mat_mult
Instruction count(static)	116	74	318	433	142
Instruction count(dynamic)	3018	878	318	2268	1241
Execution cycles	5800	2990	1391	6173	2896
Majority instructions	L,S,Se,B,A	L,S,M,D,B	L,S,A,Su	L,S,A,B	L,S,A,B
TSS simulation time(sec)	<3	<3	<2	<3	<3
DIESEL simulation time(sec)	1131	726	610	792	726
DIESEL sim time/TSS sim time	377	242	305	264	242
DIESEL energy result(uJ)	17.64	6.75	2.80	15.36	7.48
TSS energy result(uJ)	18.64	6.78	2.80	15.22	7.49
Energy Error(%)	5.7%	0.4%	0.0%	-0.9%	0.1%
Energy Error(nJ/ cycle)	0.17	0.01	0.00	-0.02	0.00
Energy Error(nJ/ instruction)	0.33	0.03	0.00	-0.06	0.01

Table 1: Comparing gate level with instruction level estimates

Program	Bub_sort	Op_loop	IDCT	FFT	mat_mult
DIESEL result(uJ)	17.64	6.75	2.80	15.36	7.48
Result using grouping(uJ)	18.64	6.78	2.80	15.22	7.49
Error using grouping%	5.7%	0.4%	0.0%	-0.9%	0.1%
Result using average(uJ)	18.95	6.59	2.80	14.98	7.39
Error using average%	7.4%	-2.4%	0.0%	-2.5%	-1.2%

Table 2: Validation of grouping approach

Program	Bubble sort	Operation loop	1D-IDCT	FFT	mat_mult
DIESEL result(uJ)	17.64	6.75	2.80	15.36	7.48
TSS result using exp. data(uJ)	18.64	6.78	2.80	15.22	7.49
Error using exp. data %	5.7%	0.4%	0.0%	-0.9%	0.1%
TSS result using derived data(uJ)	18.48	6.73	2.79	15.15	7.49
Error using derived data%	4.8%	-0.3%	-0.4%	-1.4%	0.1%

Table 3: Evaluation of predicted data

If the link between instruction level and architecture level is established, the effects of small changes in architecture can be propagated to the instruction level without total re-measurements. If the constituent instructions of a program are known, the effect of such changes can be predicted. This can provide us methods for fine tuning the architecture (with respect to power) for the application programs that use it.

Such an approach can be effectively integrated into the design space exploration for custom processor synthesis. An instruction model taking the pipeline structure into account can be used to quickly predict the effect of changes in the stage power, and other architectural modifications on the energy dissipation of the application programs. This can prove useful in the design of low power application specific processors.

7. REFERENCES

[1] G. Bernacchia and M. C. Papaefthymiou. Analytical macromodeling for high level power estimation. In *International Conference on CAD*, 1999.

[2] R. Y. Chen, R. M. Owens, M. J. Irwin, and R. S. Bajwa. Validation of an architectural level power analysis technique. In *DAC'98, San Francisco, USA*, 1998.

[3] S. Gupta and F. N. Najm. Power macromodeling for high level power estimation. In *DAC'97, Anaheim, California, USA*, 1997.

[4] G. Kane and J. Hienrich. *MIPS RISC Architecture.* Prentice Hall, 1991.

[5] M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno, and A. Sangiovanni-Vincentelli. Efficient power estimation techniques for hw/sw systems. In *IEEE Alessandro Volta Memorial Workshop on Low-Power Design, Como, Italy*, March 1999.

[6] P. L. Landman and J. M. Rabaey. Power estimation for high level synthesis. In *Proc. of EDAC-EUROASIC'93, Paris*, pages 361–366, February 1993.

[7] P. L. Landman and J. M. Rabaey. Black box capacitance models for architectural power analysis. In *Proceeding of international workshop on low power design*, pages 165–170, April 1994.

[8] P. L. Landman and J. M. Rabaey. Architectural power analysis:the dual bit type method. *IEEE Trans on VLSI systems*, 3(2):173–187, June 1995.

[9] J. Y. Lin, W. Z. Shen, and J. Y. Jou. A power modeling and characterization method for macrocells using struction information. In *International Conference on CAD*, 1997.

[10] H. Mehta, R. M. Owens, and M. J. Irwin. Energy characterization based on clustering. In *DAC'96, Las Vegas , NV, USA*, 1996.

[11] F. N. Najm. Survey of power estimation techniques in vlsi circuits. In *IEEE Trans. on VLSI*, December 1994.

[12] D. Sarta, D. Trifone, and G. Ascia. A data dependent approach to instruction level power estimation. In *IEEE Alessandro Volta Memorial Workshop on Low-Power Design, Como, Italy*, March 1999.

[13] Philips Semiconductors. *PR1900 microprocessor RISC core user manual,v1.6.* Philips Semiconductors, 1998.

[14] V. Tiwari, S. Malik, A.Wolfe, and M. C. Lee. Instruction level power analysis and optimization of software. In *Journal of VLSI Signal Processing Kluwer Academic Publishers*, August/Sept 1996.

[15] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software:a first step towards software power minimization. In *IEEE Trans. on VLSI systems*, pages 437–445, December 1994.