Co-optimization of Performance and Power in a Superscalar Processor Design

Yongxin Zhu[1], Weng-Fai Wong[2], and Ştefan Andrei[2]

[1] School of Microelectronics
Shanghai Jiao Tong University
zhuyongxin@ic.sjtu.edu.cn
[2] School of Computing
National University of Singapore
{wongwf, andrei}@comp.nus.edu.sg

Abstract. As process technology scales down, power wall starts to hinder improvements in processor performance. Performance optimization has to proceed under a power constraint. The co-optimization requires exploration into a huge design space containing both performance and power factors, whose size is over costly for extensive traditional simulations. This paper describes a unified model covering both performance and power. The model consists of workload parameters, architectural parameters plus corresponding power parameters with a good degree of accuracy compared with physical processors and simulators. We apply the model to the problem of co-optimizing the power and performance. Concrete insights into the tradeoffs of designs for performance and power are obtained in the process of co-optimization.

1 Introduction

The tradeoffs between power and performance especially in embedded processors have attracted much attention. Although there have been many analytical models and simulators which address power or performance issue separately, there is still a need for a holistic model that provides insights into complex tradeoffs in an integrated manner. Because of the complexity, even integrated models tend to focus on a few processor components such as pipelines or the instruction queue without offering a system-wide view.

In order to arrive at a more realistic system-wide view of the power-performance trade-off, we proposed an integrated model based on a previous performance model of superscalar processors. In that model, nearly all major processor components including instruction classes, instruction dependencies, the cache, the branch unit, the decoder unit, the central instruction buffer, the functional units, the retirement buffer, the retirement unit, and instruction issue policy were modelled. Later, we extended the model to out-of-order-issue processors. We further extended this performance model by linking the performance metrics with the dynamic capacitance of each processor components, thereby deriving the power consumption for each of the processor components and finally the processor as a whole. The major component of static power, *leakage power* [1, 7] was also incorporated in the model.

We validated this model by comparing its predicted power consumptions with simulation over the same benchmarks using Sim-Wattch [4] and the results are on average within 10.9% accuracy. The average power consumption obtained by our model agrees with the measured result reported by Synopsys Power Compiler with a power library from Virginia Tech [12]. Our average result also agrees with analytical outcome of the Berkeley Advanced Chip Performance Calculator (BACPAC) [13].

We explain the definitions and results of the performance model in Section 3. We then present our power model in Section 5 and show how it is combined with the performance model. Section 6 describes the validation results. In Section 7, we interpret a co-optimization issue. Then we depict how the combined model handles the issue and other concrete tradeoffs for co-optimization. This is followed by a conclusion.

2 Related Work

The performance component of our model resembles that of Noonburg and Shen [9] in terms of the similar separable components. Part of our model, namely the modelling of the instruction window, is based on the work of Pyun et. al. [10]. We go beyond their work by proposing a comprehensive model that accounts for all the key components of a state-of-the-art superscalar processor.

In addition to many traditional issues such as performance, area, cost and reliability, power consumption has been recognized as a major concern of architects of portable and embedded computer processors. High level models have been proposed to identify areas of significant power density modelled by Cai [5]. The BACPAC calculator [13] also falls into the category. Bergamaschi and Wang [2] added power states and symbolic simulation into the calculation. These models are based on architectural complexity in terms of gate equivalents, activities in a circuit, instruction-level costs, behavior-level abstraction, or system-level power estimation. However, they did not consider power-performance tradeoffs in an integrated way.

Some unified approaches to address both the power and performance have been proposed recently. Brooks et. al. [3] introduced a measured metric called the *power-performance efficiency*. Conte et. al. [6] separated architectural and technology components of dynamic power, and used a near-optimal search to tailor a processor design to different benchmarks. While Conte's model used the trace-driven simulation to collect high level statistics about pipeline stages, our model dwells into greater details of each processor component. Their approach only considers a subset of the parameters accounted for in our integrated model. Most importantly, they do not account for the clock frequency. Some other unified approaches addressed part of parameters in our model. Srinivasan et. al. [11] focused on the pipeline optimization in terms of the best power-performance efficiency. Moreshet and Bahar [2] centered on the instruction issue queue.

A recent work [16] briefed the model to integrate power and performance in an extended abstract. Another more recent work [17] focused on the co-optimization process without full details of proofs for the analytical model. In this paper, besides full details of the models, we provide for the first time, a generic solution to non-linear recurrences involved in the analytical model.

3 Performance Model

A *multiple-class multiple-resource* (MCMR) system is a queuing system where there are several classes of customers, each requiring a particular set of resources to service. To model a generic superscalar processor, we used a *network* of MCMR systems. Each stage of the pipelines contributes to the final results of the processor. The lowest throughput of all the pipeline stages is the bottleneck of the entire processor and determines the maximum possible throughput of the processor. We shall now recall the main results of the performance model.

The throughput of the processor Θ is the minimum of the service rates of *decoder unit* (μ_{dec}), *central window* (μ_{win}), and *retirement unit* (μ_{ret}):

$$\Theta = \min\{\mu_{dec}, \mu_{ret}, \mu_{win}\}. \tag{1}$$

Let W_{dec} denote the *decode width*, i.e. the maximum number of instructions that can be decoded in one cycle. Let $\overline{I}_{br}$ be the average number of (non-branch) instructions between two branch instructions (inclusive of one of the branches), T_{br} be the misprediction penalty time (the time taken to fetch and decode the correct instructions), $p_{ins,miss}$ be the instruction cache hit ratio, $t_{ins,pen}$ be the instruction cache miss penalty time, and $p_{br,prtd}$ be the probability of a correct branch prediction. If $\overline{I}_{br} < W_{dec}$, the average decoding rate without overflow in the central window, μ_{dec} is:

$$\mu_{dec} = \frac{C_1}{C_2 + C_3 \times t_{ins,pen} \times p_{ins,miss}} . \tag{2}$$

where C_1, C_2, and C_3 are linear functions of $\overline{I}_{br}$, T_{br}, W_{dec}, and $p_{br,prdt}$. The rest cases for $\overline{I}_{br}$ and W_{dec} relations are available in [15].

Let W_{ret} denote the *retire width*, i.e. the maximum number of instructions that can be retired in one cycle. Let $\overline{D}$ be the average dependence distance (inclusive of one of the instruction in the dependence) between two instructions that have a data dependence relation. Under an in-order retirement policy, the average retirement rate for $\overline{D} < W_{ret}$ is given below:

$$\mu_{ret} = (2 \times \overline{D})/(1 + T_{dep}) , \tag{3}$$

where the average time for an antecedent instruction to pass through functional units is:

$$T_{dep} = [\sum_{i}^{type} (t_i \times S_i)] \times (1 + \overline{P}_{dep}) . \tag{4}$$

where $type \in \{ieu, fpu, lsu, br\}$ is the set of types of functional units in the processor, namely the integer execution unit, the floating point unit, the load store unit and the branch unit. $S_i \in [0, 1]$ is the fraction of the total number of instructions that is executed on functional unit i for a given benchmark, and t_i is the average service time of each functional unit of type i. Typically, $t_{ieu} \in \{1, 2\}$, $t_{fpu} \in \{3, ..., 6\}$, $t_{lsu} = p_{d,prtd} + t_{dat,pen} \times (1 - p_{d,prtd})$ and $t_{br} = p_{i,prtd} + t_{ins,pen} \times (1 - p_{i,prtd})$. The parameters $p_{d,prtd}$, $p_{i,prtd} \in [0, 1]$ represent the probabilities of the data cache prediction and the instruction cache prediction, respectively. These parameters are determined by benchmarks. Thus, they vary from one benchmark to another one.

In the model, the central window works as the instruction buffer. Instructions stay in the central window after they are decoded until they are issued. For out-of-order processors, any independent and ready instruction in the instruction window may be dispatched to an available functional unit. Given $\rho_{k,t}(Z_{win})$ as the probability that k instructions of type t are issued from the window of size Z_{win}, then:

$$\mu_{win} = \sum_{t}^{type} \sum_{k=1}^{F_t} (\rho_{k,t}(Z_{win}) \times k) \ . \tag{5}$$

and $\rho_{k,t}(Z_{win}) = \mathcal{P}_{k,t}(Z_{win}) \times \phi_{pipe,t}(k)$, where $\mathcal{P}_{k,t}(Z_{win})$ is the probability that k independent instructions are extracted from Z_{win} instructions and $\phi_{pipe,t}(k)$ is the probability that at least k pipeline units of type t are available [10, 15].

So, $$\mathcal{P}_{k,t}(Z_{win}) = \mathcal{P}_{k-1,t}(Z_{win}-1) \times p_t^{(Z_{win}-1)} + \\ +\mathcal{P}_{k,t}(Z_{win}-1) \times (1-p_t^{Z_{win}-1}) \ . \tag{6}$$

4 Solving the Recurrence

Let us solve the above challenging non-linear recurrences (6) by abstracting the type t from it. In other words, we shall consider the simpler but an equivalent description of the non-linear recurrences.
Initial cases (INI): $\mathcal{P}_1(1) = 1$ and $\mathcal{P}_i(j) = 0, \forall\, i > j$;
Recursive case (REC): $\mathcal{P}_k(Z_{win}) = \mathcal{P}_{k-1}(Z_{win}-1) \times p^{Z_{win}-1} + \mathcal{P}_k(Z_{win}-1) \times (1-p^{Z_{win}-1})$
where k and Z_{win} are natural numbers, and $p \in [0, 1]$. In practice, usually $k \in \{1, ..., 10\}$ and $Z_{win} \in \{4, ..., 20\}$.

First, we show that the above recurrence has a finite number of iterations by determining the degree of the polynomial $\mathcal{P}_k(Z_{win})$ in variable p for any parameter k and Z_{win}. Moreover, for some particular cases, we can even point out the analytical form of that polynomial. For the polynomial $P(X)$ given by $a_0+ a_1X+ ... +a_mX^m$, the following notations can be done:

- $deg(P) = m$ if $a_m \neq 0$, that is, m is the maximum exponent of P with a non-zero coefficient. In this case, a_m is called the *dominant coefficient*;
- $deg(P) = m$ if $a_m \neq 0$, that is, m is the maximum exponent of P with a non-zero coefficient. In this case, a_m is called the *dominant coefficient*;

Lemma 4.1. *The following relations hold for any* $k \geq 2$ *and* $Z_{win} \geq k$*:*
(a) $\mathcal{P}_2(Z_{win}) = \sum_{i=1}^{Z_{win}-1} p^i \times \prod_{j=Z_{win}-1}^{i+1} (1-p^j)$*;*
(b) $\mathcal{P}_k(Z_{win}) = \sum_{i=1}^{Z_{win}-k+1} \mathcal{P}_{k-1}(Z_{win}-i) \times p^{Z_{win}-i} \times \prod_{j=1}^{Z_{win}-i+1} (1-p^{Z_{win}-j})$.

Proof. *(a)* Considering the identity (REC) for $k = 2$, it follows that $\mathcal{P}_2(Z_{win}-i) = p^{Z_{win}-1-i} + \mathcal{P}_2(Z_{win}-1-i) \times (1-p^{Z_{win}-1-i})$, for any $i \in \{0, ..., Z_{win}-2\}$. According to (INI), the identity for $Z_{win}-2$ is $\mathcal{P}_2(2) = p$. By replacing $\mathcal{P}_2(2)$, ..., $\mathcal{P}_2(Z_{win}-1)$, in this order, in the previous identities, it results the identity *(a)*.

(b) The identity is still a recurrence, but it is depending only in terms $\mathcal{P}_{k-1}(Z_{win}-i)$, that is, both arguments are smaller than $\mathcal{P}_k(Z_{win})$. Considering the identity (REC) for Z_{win}, $Z_{win}-1$, ..., k, it follows that $\mathcal{P}_k(Z_{win}-i) = \mathcal{P}_{k-1}(Z_{win}-i-1) \times p^{Z_{win}-i-1} + \mathcal{P}_k(Z_{win}-i-1) \times (1-p^{Z_{win}-i-1})$, for any $i \in \{0, ..., Z_{win}-k\}$. According to (INI), the identity for $Z_{win}-k$ is $\mathcal{P}_k(k) = \mathcal{P}_{k-1}(k-1) \times p^{k-1}$. By replacing $\mathcal{P}_k(k)$, ..., $\mathcal{P}_k(Z_{win}-1)$, in this order, in the previous identities, it results the identity *(b)*. ■

The following result ensures the finiteness of (REC) by specifying deg, $minDeg$, as well as the dominant and subordinate coefficients for the polynomial $\mathcal{P}_k(Z_{win})$.

Theorem 4.1. *The following relations hold for any $k \geq 2$ and $Z_{win} \geq k$:*
(a) $deg(\mathcal{P}_k(Z_{win})) = \frac{Z_{win}(Z_{win}-1)}{2}$ and the dominant coefficient of $\mathcal{P}_k(Z_{win})$ is $(-1)^{k+n}\binom{Z_{win}-2}{k-2}$;
(b) $minDeg(\mathcal{P}_k(Z_{win})) = \frac{k(k-1)}{2}$ and the subordinate coefficient of $\mathcal{P}_k(Z_{win})$ is 1.

Proof. We proceed by induction on $k \geq 2$.

Base: $k = 2$. According to identity (a) of Lemma 4.1, the highest exponent of p in $\mathcal{P}_2(Z_{win})$ corresponds to $p\times\, p^2\times\, ...\, \times p^{Z_{win}-1}$, so $deg(\mathcal{P}_k(Z_{win})) = \frac{Z_{win}(Z_{win}-1)}{2}$. Moreover, the dominant coefficient is $(-1)^{\frac{Z_{win}(Z_{win}-1)}{2}}$. The subordinate coefficient, as well as $minDeg$, can be easily obtained by considering $i = 1$ in identity (a) of Lemma 4.1.

Inductive Step: We suppose that *(a)* and *(b)* hold for any $\mathcal{P}_{k'}(Z'_{win})$, where $k' < k$, $Z'_{win} < Z_{win}$. Considering the identity *(b)* of Lemma 4.1, the subordinate coefficient can be obtained by taking $i = Z_{win} - k + 1$, that is, according to the inductive hypothesis, $p^{\frac{(k-1)(k-2)}{2}} \times p^{k-1} = p^{\frac{k(k-1)}{2}}$.

To compute the dominant coefficient for $\mathcal{P}_k(Z_{win})$, we need to sum all the dominant terms for $i = 1$ to $Z_{win} - k + 1$. According to the inductive hypothesis, the dominant term of $\mathcal{P}_{k-1}(Z_{win}-i)$ is $(-1)^{k+Z_{win}-i-1}\times \binom{Z_{win}-i-2}{k-3} \times p^{\frac{(n-i)(n-i-1)}{2}}$, $\forall\, i \in \{1, ..., Z_{win}-k+1\}$. Applying the identity *(b)* of Lemma 4.1, it follows that the dominant term of $\mathcal{P}_k(Z_{win})$ is $(-1)^{k+Z_{win}} \times \sum_{i=Z_{win}-3}^{k-3} \binom{i}{k-3} \times p^{\frac{Z_{win}(Z_{win}-1)}{2}}$. Based on the obvious combinatorial identity $\binom{m}{j} = \binom{m-1}{j} + \binom{m-1}{j-1}$, it follows that $\binom{Z_{win}-2}{k-2} = \binom{Z_{win}-3}{k-3} + \binom{Z_{win}-3}{k-2} = ... = \sum_{i=Z_{win}-3}^{k-3} \binom{i}{k-3}$. Therefore, *(a)* and *(b)* hold for the general case. ■

The analytical form of the polynomial $\mathcal{P}_k(Z_{win})$ is very hard to be obtained. However, there are two general forms which allow that (Theorem 4.2).

Theorem 4.2. *For any $k \geq 2$, we have $\mathcal{P}_k(k) = p^{\frac{k(k-1)}{2}}$ and $\mathcal{P}_k(k+1) = (1-k)$ $p^{\frac{k(k+1)}{2}} + \sum_{i=\frac{k(k-1)}{2}}^{\frac{k(k+1)}{2}-1} p^i$.*

Proof. Taking $Z_{win} = k$ in (REC), it follows that $\mathcal{P}_k(k) = \mathcal{P}_{k-1}(k-1) \times p^{k-1}$. By iterating this relation for $k \geq 1$, it follows that $\mathcal{P}_k(k) = \mathcal{P}_1(1)\times\, p^{k-1}\times\, ... \,\times p^2 \times p = p^{\frac{k(k-1)}{2}}$.

For obtaining $\mathcal{P}_k(k+1)$, we proceed by induction on k. The case $k = 2$ holds obviously. According to (REC), we have $\mathcal{P}_k(k+1) = \mathcal{P}_{k-1}(k)\times\, p^k +\, \mathcal{P}_k(k)\times\, (1 - p^k) = \mathcal{P}_{k-1}(k)\times\, p^k +\, p^{\frac{k(k-1)}{2}} \times (1-p^k)$. According to the inductive hypothesis, this can be continued by $p^{\frac{(k-1)(k-2)}{2}+k} +\, p^{\frac{(k-1)(k-2)}{2}+1+k} +\, ... \,+ p^{\frac{k(k-1)}{2}-1+k} -\, (k-2)\times p^{\frac{(k-1)(k-2)}{2}+k} +\, p^{\frac{(k-1)(k-2)}{2}} -\, p^{\frac{(k-1)(k-2)}{2}}$, which is equivalent to what was needed to be proved. ∎

5 Power Model

The power consumption of a resource consists of a dynamic and a static component, i.e., $\pi_{tot,res} = \pi_{static,res} + \pi_{dyn,res}$. The static portion is given by $\pi_{static,res} = I_{static,res} \times V_{dd}$. The leakage current $I_{static,res}$ is an exponential function of threshold voltage V_t (in mV) by Sylvester and Keutzer [14]:

$$I_{static,res} = 10 \times \omega \times 10^{-V_t/95} . \tag{7}$$

where ω is the device width in micro meter. According to the formula, the static power increases with the downsizing process technologies. For any technology node, the static power takes a usually stable portion of the total power. Khouri and Jha [7] summarized the ratios of the static power over the total power based on 6 different circuits, which are listed in Table 1.

For the dynamic power component, which is dependent on workloads, we used a model that is similar to that of several recent studies [2], [8]. We model dynamic power as a function of *dynamic capacitance* ($\mathcal{C}_{res}$), the *supply voltage* (V_{dd}) and the *clock frequency* (Ω):

$$\pi_{dyn,res} = \mathcal{C}_{res} \times V_{dd}^2 \times \Omega \, . \tag{8}$$

For each component of the processor, the capacitance is obtained by either using the same empirical formulas used by Sim-Wattch or by means of summing up the bit stream changes. With total dynamic capacitance and number of accesses of a resource, we can obtain the *dynamic capacitance per access to the resource* ($\mathcal{C}_{a,res}$)

Table 1. The Proportions of Leakage Power in Total Power

Tech.	**stat. pwr. /tot. pwr. without leakage opti.**	**stat. pwr./tot. pwr. with leakage opti.**	V_{dd}
0.35μm	9.8%	6.6%	3.3
0.18μm	22.6%	11.7%	1.8
0.13μm	43.4%	26.9%	1.5
0.10μm	48.1%	25.5%	1.2
0.07μm	56.2%	25.1%	0.9

for each benchmark. The values of the various $\mathcal{C}_{a,res}$ used by the model are shown in Table 3.

The total power of a processor is the sum of the power consumption by each resource/component.

$$\pi_{dyn,tot} = \pi_{win} + \pi_{ret} + \pi_{dec} + \pi_{ieu} + \pi_{fpu} + \pi_{lsu} + \pi_{br} + \\ + \pi_{icache} + \pi_{dcache} \ . \tag{9}$$

6 Validation of Models

The earlier version of the performance model was for in-order-issue processors, and it was validated against the results measured physically on an UltraSPARC processor with an average error of 5.1%. This performance model was extended to out-of-order issue processors, and validated with SimpleScalar out-of-order issue simulated processor with a small average error of 5.9%.

As further validations, we also compared our results with those of other power models. The BACPAC [13] calculator shows that the typical power consumption is 24.03 watts for a 5-million-transistor processor running at 600MHz and V_{dd} of 2.5V. The power consumption is close to the averaged analytical power of 27.38 watts. Using the same V_{dd}, clock frequency and a 0.25μm technology based power library by Sulistyo and Ha [12], we also obtained a total power of 32.1 watts reported by the Synopsys Power Compiler. for a similar RISC processor design in the scale. More details of the validations are available in [17].

The inputs to the performance model are given in Table 2. The capacitance parameters from Table 3 are inputs to our power model. Our architectural analysis yields values of $N_{a,req,res}$: $N_{a,req,win} = 6$, $N_{a,req,regfile} = 2$ and $N_{a,req,dec} = N_{a,req,ieu} = N_{a,req,fpu} = N_{a,req,lsu} = N_{a,req,br} = N_{a,req,icache} = N_{a,req,dcache} = 1$. We assume the service rate of register file equals the one of retirement unit, that is $\mu_{ret} = \mu_{regfile}$.

Table 2. Benchmark Characteristics for the performance model

Bench.	**bzip2**	**equake**	**mcf**	**mesa**	**vpr**
S_{ieu}	45.7%	26.3 %	39.4 %	42.2 %	43.6 %
S_{fpu}	0.0%	15.3%	0.0 %	7.0 %	5.6 %
S_{br}	15.9%	6.1%	2.7 %	1.0 %	1.0 %
S_{lsu}	28.5%	41.5%	48.2%	53.4 %	53.4 %
$\overline{D}$	1.996	1.955	2.016	1.873	1.911
$\overline{I}_{br}$	7.26	6.69	3.65	4.18	4.74
$\overline{P}_{dep}$	0.562	0.504	0.620	0.425	0.589
T_{dep}	1.972	2.403	2.085	2.248	2.187
p	0.438	0.4962	0.3802	0.5755	0.5178
q	0.991	0.975	0.995	0.95	0.983
$p_{ins,miss}$	0.0110	0.0343	0.0038	0.0296	0.0067
$p_{dat,miss}$	0.0227	0.0552	0.1589	0.0221	0.0820

Table 3. Capacitance (in 10^{-10} farad) Primitives for Our Power Model

Bench.	**bzip2**	**equake**	**mcf**	**mesa**	**vpr**
$\mathcal{C}_{a,win}$	0.631	0.898	1.004	0.762	0.769
$\mathcal{C}_{a,regfile}$	2.665	3.806	4.527	3.330	3.590
$\mathcal{C}_{a,dec}$	0.421	0.603	0.614	0.485	0.501
$\mathcal{C}_{a,ieu}$	16.32	24.09	26.18	19.56	20.33
$\mathcal{C}_{a,fpu}$	16.32	24.09	26.18	19.56	20.33
$\mathcal{C}_{a,lsu}$	2.527	3.981	4.087	3.912	3.035
$\mathcal{C}_{a,br}$	38.90	53.14	37.39	28.84	43.83
$\mathcal{C}_{a,icache}$	2.751	3.911	3.846	3.152	3.194
$\mathcal{C}_{a,dcache}$	17.09	27.02	27.72	27.35	24.33

7 Co-optimization Applications of the Models

We shall now show by examples how the model can be used to explore the design space to reach a co-optimized solution.

A Co-optimization Issue: To co-optimize power and performance, we need to minimize $\pi_{dyn,tot}$ in (10), while maximizing the throughput in terms of number of instructions per second, i.e. $\Theta \times \Omega$. Firstly, we let the user set an upper limit, π_U say, i.e. $\pi_{dyn,tot} \leq \pi_U$. Within this constraint, we seek to maximize Θ in (1) along with varying Ω. In short, this approach is to maximize the throughout under a power budget.

In order to obtain the configuration with the least energy consumption for a computation, we look for the minimal total energy to finish the task whose number of instructions is n_i. Let $\pi_{u,x}$ be the upper power limit for the x-th optimization case, the constraint $\pi_{dyn,tot} \leq \pi_{u,x} \leq \pi_U$ should hold when seeking for the maximum performance $\theta \times \Omega$. If such a case x exists, then the time to execute the application is $n_i/(\theta \times \Omega)$. Consequently, this will also yield the minimal total energy, at the x-th case where the power is $\pi_{dyn,tot}$:

$$E_x = n_i \times \pi_{dyn,tot}/(\theta \times \Omega) \tag{2}$$

Impact of Clock Frequency: We will now use `bzip2` as an example to show how co-optimization is achieved. To begin, we set an upper bound on the dynamic power, $\pi_{dyn,U} = 25$ watts. The co-optimized solution is obtained by the following search procedure:
1. Read the performance values of `256.bzip2` from Table 2: $\{S_{ieu} = 0.457$, $S_{fpu} = 0.0$, $S_{lsu} = 0.285$, $S_{br} = 0.159$, $\overline{D} = 1.9960$, $\overline{I}_{br} = 7.26$, $\overline{p}_{dep} = 0.562$, $p_{ins,miss} = 0.0110$, $p_{dat,miss} = 0.0227$, $p_{d,prtd} = 1-p_{dat,miss}$, $p_{i,prtd} = 1-p_{ins,miss}$, $p_{br,prtd} = 1 - p_{ins,miss}\}$. These benchmark specific parameters along with architectural parameters $\{t_{ieu} = 1$, $t_{fpu} = 3$, $t_{dat,pen} = 3$, $t_{ins,pen} = 2$, $Z_{win} = 8$, $type = 4$, $W_{dec} = 4\}$ are fed into (2), (3) and (5) to obtain μ_{dec}, μ_{ret}, and μ_{win} then $\mu_{lsu} = \mu_{dec} \times S_{lsu}$, μ_{icache} and μ_{dcache}.

2. For the power constraint on the dynamic power, $\pi_{u,x}$ from 25 watts down to 1 watt in steps of -1 watt do:

2.1. For each clock frequency Ω from 100 to 600 MHz at a step of 100 MHz, we repeat the following steps to obtain the maximum performance $\theta \times \Omega$ under the power constraint of 25 watts.

2.1.1. With the above performance service ratios of resources and Ω, we obtain π_{res} in (8), where $\mathcal{C}_{a,res}$ is obtained from Table 3.

2.1.2. Sum up π_{res} for all components. If the total $\pi_{dyn,tot}$ is less than π_u, then we have found a configuration within the constraints. We also note down the performance $\theta \times \Omega$ and $\pi_{dyn,tot}$.

3. Find the maximum of $\theta \times \Omega_i$ and its associated $\pi_{dyn,tot}$ and Ω_i.

Impact of Leakage Power: Using our model, we can study the impact of leakage power on the maximum clock frequencies and dynamic power consumptions If we vary the clock frequencies while keeping the rest parameters fixed, we can obtain clear changes in both leakage power and dynamic power. We find the leakage power without optimization grows consistently along with the reduction of feature size. For the technology node of 0.07μm, the leakage power overtakes the dynamic power as the dominant power factor. This trend hinders the increase of clock frequencies which ranges from 400 Mhz for 0.35μm technology to 3 Ghz for 0.07μm technology. With optimizations [7] on leakage power, the total power budget can be more effectively spent on the dynamic power consumption. The leakage power will be kept lower than the dynamic power. The maximum clock frequency for 0.07μm technology can be improved to 5.2 Ghz.

Projection of Minimum Dynamic Power: We also apply our model to gauge the minimum dynamic power for different benchmarks. We keep the processor configuration fixed, and seek for a possible low for a certain workload. In practice, we use the service rates of resources μ_{res} and the capacitance primitives of resources $\mathcal{C}_{a,res}$ in Table 3 to obtain the dynamic capacitances of resources $\mathcal{C}_{res}$. The dynamic power $\pi_{dyn,res}$ is obtained by feeding $\mathcal{C}_{res}$, V_{dd} and Ω into Equ. (8).

For example, the minimum $\mathcal{C}_{a,win}$ for the instruction window is 0.631 (10^{-10} farad) in Table 3, and the minimum μ_{win} for the instruction window is 1.548. Along with the average number of access to the instruction window per request, $N_{a,req,win} = 6$, we obtain the minimum $\mathcal{C}_{a,win}$ as $\mathcal{C}_{a,win} = 0.631 \times 10^{-10} \times 1.548 \times 6 \approx 5.861 \times 10^{-10}$. Then the minimum $\pi_{dyn,win} = 5.861 \times 10^{-10} \times 2.5^2 \times 600 \times 10^6 \approx 2.198$ watts. The minimum total dynamic power of 15.81 watts is found by repeating the above process for all the resources. This bound implies that the processor dynamic power can be reduced to lower than the bound with proper scheduling and choice of workloads.

8 Conclusion

In this paper, we raise an approach to power and performance co-optimization using our unified model accounting for both issues. Validation against an established power simulator using large SPEC2000 benchmarks indicates the accuracy of the model. The results are also in agreement with previous analytical studies and experimental results.

In the process of co-optimization, we showed the impact of leakage power on the performance improvements for different technology nodes. We also obtained a bound of the minimum dynamic power. In addition, we found that the clock frequency is the dominant factor compared to the cache, instruction window and functional units in improving performance under dynamic power constraints. These results illustrate our model is a useful tool for designers to make power-aware decisions at early stages of co-optimization.

References

1. F. A. Aloul, S. Hassoun, K. A. Sakallah, and D. Blaauw. Robust sat-based search algorithm for leakage power reduction. In *Proc. of the 12th Int'l Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation*, pages 167–177. Springer-Verlag, 2002.
2. R. N. Bergamaschi and Y. W. Wang. State-based power analysis for systems-on-chip. In *Proc. of the 40th Int'l Design Automation Conference (DAC-40)*, pages 638–641, June 2003.
3. D. Brooks, P. Bose, S. Schuster, H. Jacobson, P. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, and P. Cook. Power-aware microachitecture: Design and modeling challenges for next-generation microprocessors. *Micro, IEEE*, 20(6):26–44, November-December 2000.
4. D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. of 27th Ann. Int'l Symp. Computer Architecture (ISCA)*, pages 83–94. IEEE Computer Society Process, Los Alamitos, California, USA, 2000.
5. G. Cai and C. Lim. Architectural level power/performance optimization and dynamic power estimation. In *Cool Chips Tutorial Colocated with the 32nd Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO-32)*, November 1999.
6. T. Conte, K. Menezes, S. Sathaye, and M. Toburen. System-level power consumption modeling and tradeoff analysis techniques for superscalar processor design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(2):129–137, 2000.
7. K. Khouri and N. Jha. Leakage power analysis and reduction during behavioral synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(6):876–885, December 2002.
8. T. Moreshet and R. I. Bahar. Power-aware issue queue design for speculative instructions. In *Proc. of the 40th Int'l Design Automation Conference (DAC-40)*, pages 634–637, June 2003.
9. D. Noonburg and J. Shen. Theoretical modeling of superscalar processor performance. In *Proc. of 27th Annual IEEE/ACM Int'l Symposium on Microarchitecture (MICRO-27)*, pages 53–62, 1994.
10. Y. Pyun, C. Park, and S. Choi. The effect of instruction window on the performance of superscalar processors. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E81A(6):1036–1044, June 1998.
11. V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. N. Strenski, and P. G. E. Sylvester. Optimizing pipelines for power and performance. In *Proc. of MICRO-35*, pages 333–344, November 2002.
12. J. Sulstyo and D. S. Ha. A new characterization method for delay and power dissipation of standard library cells. *VLSI Design*, 15(3):667–678, 2002.
13. D. Sylvester, W. Jiang, and K. Keutzer. Bacpac - Berkeley Advanced Chip Performance Calculator. In *Available online, http://www.eecs.umich.edu/ dennis/bacpac/*, 1998.
14. D. Sylvester and K. Keutzer. Getting to the bottom of deep submicron. In *Proc. of Int'l Conference on CAD*, pages 203–211, November 1998.

15. Y. Zhu and W. Wong. Sensitivity analysis of a superscalar processor model. In *Proc. of the 7th Asia-Pacific Computer Systems Architectures Conference, Melbourne, Australia*, pages 109–118, January 2002.
16. Y. Zhu, W. Wong, and S. Andrei. An integrated performance and power model for superscalar processor designs. In *Proc. of IEEE Asia South Pacific Design Automation Conference (ASP-DAC) 2005, Shanghai, China*, pages 948–951, 2005.
17. Y. Zhu, W. Wong, and C. Koh. A performance and power co-optimization approach for modern processors. In *Proc. of 5th Int'l Conference on Computer and Information Technology (CIT) 2005, Shanghai, China*, pages 822–828, 2005.