A scoring metric for the CBP-NG championship

Pierre Michaud

November 25, 2025

The classic branch prediction championship (aka CBP[1]) usually evaluates branch predictors along a single dimension, their prediction accuracy, with a single constraint, a limited storage budget. The CBP-NG championship evaluates conditional branch predictors (**CBP**) along several dimensions, including accuracy, throughput, latency, and energy, in an attempt to be closer to real constraints. CBP-NG does not constrain the amount of storage. Instead, contestants must use a C++ library, called HARCOM, allowing the functional simulation of microarchitectural algorithms while at the same time providing an estimate of their hardware complexity [7].

The CBP-NG simulator characterizes a CBP by three figures of merit (FoM):

- IPC_{cbp}: average prediction throughput in instructions per clock cycle;
- CPI_{cbp}: average number of cycles lost per correct-path instruction because of mispredictions detected at execution;
- EPI_{cbp}: average dynamic energy of the CBP per correct-path instruction.

These three FoM are combined via a mathematical formula to produce a single number called the **VFS**, where VFS stands for *voltage-frequency-scaled speedup*.

The document is organized as follows. Section 1 describes how the CBP-NG simulator works. Section 2 explains how the three FoM are obtained. Section 3 introduces the VFS and provides some intuition for it. Section 4 explains how the VFS formula should be used and how the reference predictor is defined. Section 5 provides a detailed derivation of the VFS formula.

1 The CBP-NG simulator

Like the classic championship, CBP-NG evaluates the conditional branch predictor (CBP) in isolation from the rest of the microarchitecture. Therefore, the CBP-NG simulation process is an approximation of how branch prediction works in reality.

The CBP-NG simulator is trace driven. A trace contains only correct-path instructions, in sequential order. The main simulation loop processes instructions one at a time, i.e., there is one loop iteration per instruction. For each instruction, the simulation loop asks the predictor to provide a first direction prediction (i.e., one bit), then a second direction prediction. This allows simulating an overriding predictor structure where a fast predictor drives a slower, more accurate predictor [5, 6]. The CBP encompasses the hardware for obtaining the first and second direction predictions, and the hardware for updating the CBP state.

[1] The abbreviation "CBP" was defined by the first branch prediction championship[14]. Throughout this document, it is used to mean *conditional branch predictor*, except when refering to a championship (CBP-NG or CBP2025).

The only information that is given to the predictor about the current instruction is its address. The instruction type is unknown to the predictor at that point. If the current instruction is not a conditional branch, the two predictions are ignored. If the instruction is an unconditional jump, it is assumed to be predicted perfectly.

If the first and second prediction for a conditional branch are different, this is a *short misprediction*. The second prediction is the final prediction. If it is incorrect, i.e., different from the actual branch direction, this is a *long misprediction*.

Just after a conditional branch gets its two predictions, the address of the branch is immediately provided to the predictor, along with the actual branch direction and the address of the next instruction.[2]

Before processing the next instruction, the simulation loop determines whether or not this is the end of the current *block*. A block is a set of instructions that are predicted simultaneously, i.e., in the same cycle. A block ends in any of the following cases:

- the current instruction is a jump (unconditional or conditional taken);
- the current instruction is a conditional branch undergoing a long misprediction;
- the predictor asked to end the block.

When a block ends, this information is communicated to the predictor, and a block counter N_{block} is incremented. If the block end coincides with a short misprediction[3], we increment a counter $N_{coincide}$.

The predictor inputs and outputs are HARCOM values. Each HARCOM value is timed. The timing of instruction addresses and that of corresponding predictions permits determining the prediction latency. For an instruction i, let t_i be the timing of its address and t'_i the timing of its prediction. Timings are in picoseconds. Instructions in the same block have the same t_i. After the end of the current block, t_i is incremented by τ, the clock cycle duration in picoseconds. The prediction latency, in cycles, is

$$L = \left\lceil \frac{\max(0, \max_i(t'_i - t_i))}{\tau} \right\rceil$$

where the inner max is computed over all instructions. The latency formula above is applied to the first and second predictions of the evaluated predictor. We denote L_1 and L_2 the latencies of the first and second predictions, respectively.

The predictor can ask for extra cycles in order to write in a RAM (a HARCOM RAM is limited to one access per cycle).

2 The three figures of merit (FoM)

2.1 Prediction throughput on the correct path

The maximum throughput is one block per cycle. Maximum throughput can only be attained if the first-prediction latency does not exceed one cycle ($L_1 \leq 1$), if there are no short mispredictions, and if the predictor did not ask for extra cycles. A short misprediction costs L_2 cycles.

[2] This allows updating the CBP state with correct information immediately, which is not realistic and leads to overestimating prediction accuracy [11]. Nevertheless, causality is not violated.

[3] Conditional branch whose first and second predictions are not-taken and taken, respectively.

notation	meaning
IPC_{cbp}	prediction throughput in instructions per cycle
CPI_{cbp}	long-misprediction penalty in cycles per correct-path instruction
EPI_{cbp}	CBP energy per predicted instruction
IPC	(correct-path) instructions retired per cycle
EPI	total core energy per correct-path instruction
L_1	latency of the first prediction, in cycles
L_2	latency of the second prediction, in cycles
L_{pipe}	pipeline stages after the second prediction until branch execution
N_{cp}	total number of correct-path instructions
N_{wp}	total number of wrong-path instructions entering the pipeline
N_{block}	total number of blocks
N_{short}	total number of short mispredictions
N_{long}	total number of long mispredictions
$N_{coincide}$	total number of block ends coinciding with a short misprediction
T_{extra}	extra cycles
T_{cp}	total correct-path prediction cycles
T_{wp}	total wrong-path prediction cycles
E_{cbp}	total dynamic energy spent in the CBP by correct-path instructions
WPI	wrong-path instructions per correct-path instruction

Table 1: Notation

We define the total number T_{cp} of prediction cycles on the correct path as follows:

$$\begin{aligned} &\text{if } L_1 < L_2, \qquad T_{cp} = (N_{block} - N_{coincide}) \times \max(1, L_1) + N_{short} \times L_2 + T_{extra} \\ &\text{if } L_1 \geq L_2, \qquad T_{cp} = N_{block} \times \max(1, L_2) + T_{extra} \end{aligned}$$

where N_{block} is the total number of blocks, N_{short} the total number of short mispredictions and T_{extra} the total number of extra cycles asked by the predictor (see Table 1). The second case ($L_1 \geq L_2$) may happen, for instance, if a contestant chooses not to implement an overriding scheme (the first and second prediction are identical).

The average prediction throughput on the correct path is

$$\text{IPC}_{cbp} = \frac{N_{cp}}{T_{cp}} \tag{1}$$

where N_{cp} is the total number of correct-path instructions.

2.2 Penalty from long mispredictions

We assume a decoupled frontend where the branch predictor runs ahead of instruction fetching [10]. In such scheme, the fetching of the next block starts after the final predictions for the current block are known. So the latency of the second prediction, L_2, is part of the penalty of a long misprediction. The penalty of a long misprediction is $L_2 + L_{pipe}$ cycles, where L_{pipe} is the number of pipeline stages after the second prediction is known, until branch execution.

We assume that the performance of the core is bottlenecked by the CBP. In particular the frontend throughput is IPC_{cbp} and the backend can execute instructions as fast as the frontend produces them, with a sufficiently large instruction window. Under this assumption, the total number T_{wp} of prediction cycles on the wrong path of long mispredictions is

$$T_{wp} = N_{long} \times (L_2 + L_{pipe})$$

The average number of cycles lost because of long mispredictions, per correct-path instruction, is

$$\text{CPI}_{cbp} = \frac{T_{wp}}{N_{cp}} \tag{2}$$

2.3 CBP energy

The average dynamic energy spent in the CBP per correct-path instruction is

$$\text{EPI}_{cbp} = \frac{E_{cbp}}{N_{cp}} \tag{3}$$

where E_{cbp} is the total dynamic energy spent in the CBP by correct-path instructions.

3 The VFS formula

We evaluate the predictors with a higher-is-better metric called the VFS, for *voltage-frequency-scaled speedup*. The VFS combines the three FoM of the evaluated predictor (IPC_{cbp}, CPI_{cbp}, EPI_{cbp}) and those, fixed, of a reference predictor (IPC^*_{cbp}, CPI^*_{cbp}, EPI^*_{cbp}) to produce a single dimensionless number. This section provides the VFS formula without explaining how it is obtained.[4] The detailed derivation is provided in Section 5. The goal here is to provide some intuition.

The IPC is

$$\text{IPC} = \frac{1}{\frac{1}{\text{IPC}_{cbp}} + \text{CPI}_{cbp}} \tag{4}$$

The number of wrong-path instructions per correct-path instruction is

$$\begin{aligned} \text{WPI}^* &= \text{IPC}^*_{cbp} \times \text{CPI}^*_{cbp} \\ \text{WPI} &= \text{IPC}_{cbp} \times \text{CPI}_{cbp} \end{aligned}$$

The IPC speedup is

$$\frac{\text{IPC}}{\text{IPC}^*} = \frac{\text{IPC}_{cbp}}{\text{IPC}^*_{cbp}} \times \frac{1 + \text{WPI}^*}{1 + \text{WPI}}$$

The normalized EPI is

$$\frac{\text{EPI}}{\text{EPI}^*} = \left[\lambda \times \left(\frac{\text{IPC}}{\text{IPC}^*} \right)^{\gamma} + \mu \times \frac{\text{EPI}_{cbp}}{\text{EPI}^*_{cbp}} \right] \times \left(1 + \frac{\text{WPI}}{2} \right) \tag{5}$$

[4] The VFS formula is based on some assumptions that are not necessarily accurate. It is not intended for academic studies.

where the constant factors μ and λ represent the fraction of core energy spent by correct-path instructions in the reference CBP and in the rest of the reference core, respectively ($0 < \mu \ll \lambda < 1$). Finally, the VFS formula is

$$\mathrm{VFS} = \frac{\mathrm{IPC}}{\mathrm{IPC}^*} \times \alpha \times \left(1 - \frac{2}{1 + \sqrt{1 + \beta \times \frac{\mathrm{EPI}^*}{\mathrm{EPI}} \times \frac{\mathrm{IPC}^*}{\mathrm{IPC}}}}\right) \tag{6}$$

where $\alpha = 1.625$ is a technological parameter, $\beta = \frac{4\alpha}{(\alpha-1)^2} = 16.64$ and $\gamma = \frac{2}{\alpha-1} = 3.2$.

The two main assumptions of the VFS metric are:

- the *average* IPC is bottlenecked by the CBP;
- the core power budget is fixed and equal to the power consumption of a *reference core*.

The first assumption implies that the core is not fixed but depends on the IPC: a higher IPC implies a more complex core consuming more energy. This corresponds to the $\lambda \times (\mathrm{IPC}/\mathrm{IPC}^*)^\gamma$ quantity in (5), which models core energy as a function of IPC. The $(1 + \mathrm{WPI}/2)$ factor in (5) represents the energy overhead of wrong-path instructions.

The second assumption has the following meaning: if the core power exceeds the power of the reference core, voltage and clock frequency are reduced to meet the power constraint; if the core power is under the power of the reference core, voltage and clock frequency are increased to use the power budget. The scaling factor $\alpha \times (1 - \frac{2}{1+\sqrt{\cdots}})$ in the VFS formula represents frequency scaling.

The VFS is maximum, equal to 1, when the predictor has the same FoM as the reference predictor. We set the reference FoM to values that we believe are difficult to attain, but are nevertheless nearly realistic.[5] Therefore, to increase the VFS, one must increase IPC_{cbp}, decrease CPI_{cbp} or decrease EPI_{cbp}.

The CBP-NG simulation infrastructure provides a script called `vfs.py` that takes the three FoM as input and gives the VFS value. This script is used to assess the predictor. It can also be used to test various scenarios, e.g., is it worth increasing prediction throughput by 10% if that costs 10% more mispredictions?

The VFS formula can be expressed in a slightly simpler (but equivalent) form by using the normalized versions of the three FoM:

$$x = \frac{\mathrm{IPC}_{cbp}}{\mathrm{IPC}^*_{cbp}} \qquad y = \frac{\mathrm{CPI}_{cbp}}{\mathrm{CPI}^*_{cbp}} \qquad z = \frac{\mathrm{EPI}_{cbp}}{\mathrm{EPI}^*_{cbp}}$$

We have

$$\begin{aligned} \frac{\mathrm{IPC}}{\mathrm{IPC}^*} &= \frac{1 + \mathrm{WPI}^*}{1/x + \mathrm{WPI}^* y} \\ \frac{\mathrm{EPI}}{\mathrm{EPI}^*} &= \left[\lambda \times \left(\frac{\mathrm{IPC}}{\mathrm{IPC}^*}\right)^\gamma + \mu z\right] \times \left(1 + \frac{\mathrm{WPI}^*}{2} xy\right) \end{aligned}$$

Figure 1 shows iso-VFS contours for variable x and y, when $z = 1$ and $z = 2$. To increase the VFS, starting from $x < 0.8$ and $y > 1$, it is most effective to increase x first. Reducing y is more important when x is close to 1. Increasing z, i.e., energy consumption, makes it more difficult to bring the VFS close to 1.

[5] The reference FoM are defined in the `vfs.py` script.

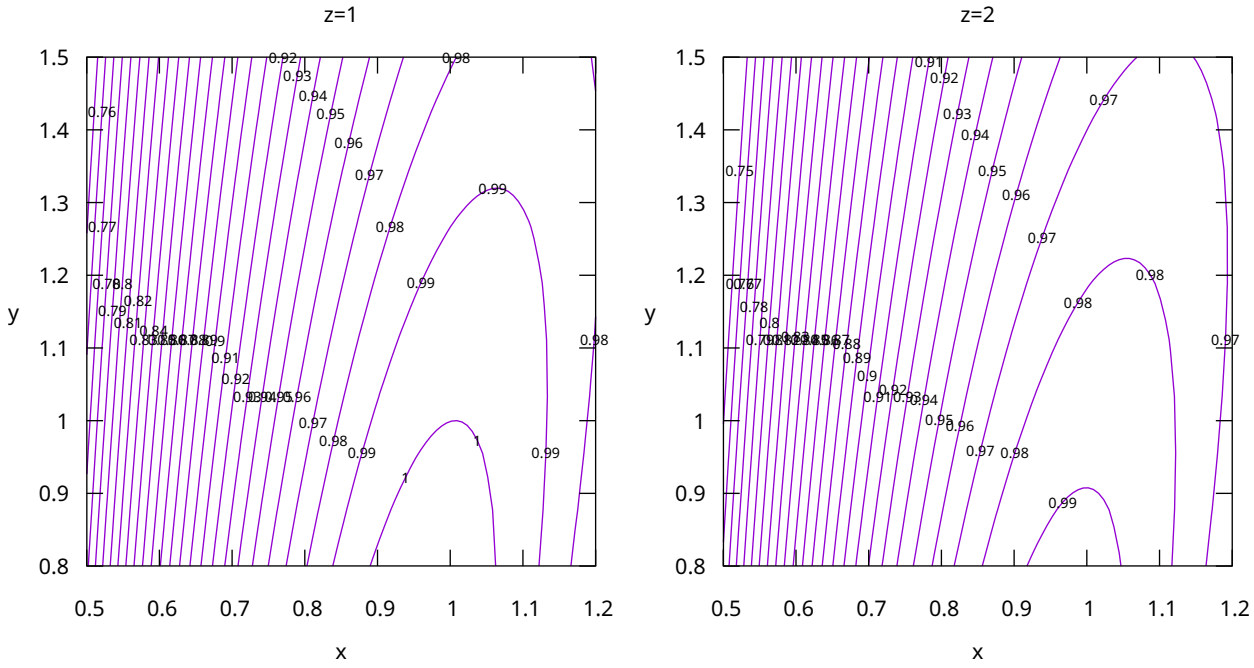

Figure 1: Iso-VFS contours for variable $x = \mathrm{IPC}_{cbp} / \mathrm{IPC}^*_{cbp}$ and $y = \mathrm{CPI}_{cbp} / \mathrm{CPI}^*_{cbp}$, and for fixed $z = \mathrm{EPI}_{cbp} / \mathrm{EPI}^*_{cbp} = 1$ (left) and $z = 2$ (right).

4 Averaging the FoM and defining a reference predictor

The VFS formula assumes that the *average* IPC is bottlenecked by the CBP. However, the CBP is evaluated on multiple traces with different characteristics. The $\mathrm{IPC}(\mathtt{t}_1)$ obtained with one trace $\mathtt{t}_1$ can be very different from the $\mathrm{IPC}(\mathtt{t}_2)$ obtained with another trace $\mathtt{t}_2$. However, the CBP is the same for all the traces, it is fixed, and so is the core. Therefore, it is not the ouput of the VFS formula that we must average, but its inputs. We average IPC_{cbp} over all traces with a harmonic mean, and we average CPI_{cbp} and EPI_{cbp} with an arithmetic mean [2]. That is, denoting $\mathbb{T}$ the set of traces used to rank the competing predictors, and $|\mathbb{T}|$ the number of traces,

$$
\begin{aligned}
\mathrm{IPC}_{cbp} &= \frac{|\mathbb{T}|}{\sum_{\mathtt{t}\in\mathbb{T}} \frac{1}{\mathrm{IPC}_{cbp}(\mathtt{t})}} \\
\mathrm{CPI}_{cbp} &= \frac{1}{|\mathbb{T}|} \sum_{\mathtt{t}\in\mathbb{T}} \mathrm{CPI}_{cbp}(\mathtt{t}) \\
\mathrm{EPI}_{cbp} &= \frac{1}{|\mathbb{T}|} \sum_{\mathtt{t}\in\mathbb{T}} \mathrm{EPI}_{cbp}(\mathtt{t})
\end{aligned}
$$

Note that the IPC formula (4) is consistent with this averaging, as applying the formula to individual traces $\mathtt{t}$ and averaging the resulting $\mathrm{IPC}(\mathtt{t})$ with a harmonic mean gives the same number as applying the formula directly to the average IPC_{cbp} and CPI_{cbp}.

The second main assumption of the VFS is that the power budget is fixed. This power budget is implicitly defined by the FoM of the reference predictor, IPC^*_{cbp}, CPI^*_{cbp}, and EPI^*_{cbp}. The reference FoM are defined by the organizers. They are fixed and have been calibrated for the set of traces $\mathbb{T}$ used to rank the competing

predictors.

There is an absolute upper bound for prediction throughput, as the CBP-NG simulator assumes that a block cannot continue past a jump. However, the actual IPC_{cbp} is significantly lower than this natural upper bound because of short mispredictions and extra cycles needed to write single-ported RAMs. We set IPC^*_{cbp} to a value that we believe is hard to attain when averaging an actual IPC_{cbp} on $\mathbb{T}$, but which is nevertheless close to a realistic value. We set CPI^*_{cbp} equal to that of the TAGE-SC predictor that was ranked second[6] at the CBP2025 championship [12], assuming an unrealistic zero latency for the second prediction ($L_2 = 0$). As for EPI^*_{cbp}, we set it approximately equal to that of the example TAGE predictor provided with the CBP-NG simulation infrastructure.

Set $\mathbb{T}$ contains some traces that are given to contestants to train their predictors. This is the public subset $\mathbb{T}_p$. The other traces $\mathbb{T} - \mathbb{T}_p$ are secret and are there to prevent overfitting. The public subset is representative of the full set: the reference predictor is consistent both with the full set $\mathbb{T}$ and with the public subset $\mathbb{T}_p$. In particular, the CPI_{cbp} of TAGE-SC is approximately the same when averaged over $\mathbb{T}$ or $\mathbb{T}_p$, and IPC^*_{cbp} is hard to attain both on $\mathbb{T}$ and $\mathbb{T}_p$.

In summary, the inputs of the VFS formula must be the FoM averaged over $\mathbb{T}$ or $\mathbb{T}_p$. **Do not apply the VFS formula on a single trace or on truncated traces**. If you need to truncate traces to reduce simulation time (e.g., for exploring the design space), just look at the three FoM.

5 Derivation of the VFS formula

The IPC, that is, the number of instructions retired per cycle, is

$$\mathrm{IPC} = \frac{N_{cp}}{T_{cp} + T_{wp}} = \frac{1}{\frac{1}{\mathrm{IPC}_{cbp}} + \mathrm{CPI}_{cbp}} \tag{7}$$

where we have used (1) and (2). Equation (7) combines two of the three FoM. It does not take into account energy though. When looking for a scoring metric, we briefly considered the possibility of using (7) with a fixed energy buddget for the CBP, i.e., a maximum EPI_{cbp} that contestants would have to stay under. However, we decided to take a different approach, for the following reasons:

- It is not clear what the energy budget should be for the CBP. Reducing the number of mispredictions at the cost of higher CBP energy may actually reduce the core energy.
- The classic championship imposes a fixed storage budget. It is easy for contestants to adjust storage to meet a storage constraint. Adjusting energy consumption, however, is more difficult.
- Equation (7) is based on the assumption that the IPC is bottlenecked by the CBP. This makes the core's hardware complexity depend on the CBP's performance. This is not consistent with a fixed energy budget for the CBP, as a more complex core can spend more energy in its CBP. So a model of core energy is needed.

Instead, we decided to use a metric combining IPC and core energy. An example of such metric is the well-known energy-delay product or, more generally, $E \times D^n$ metric [8].

[6] The winning predictor of CBP2025 exploits correlations between branch directions and register values. The CBP-NG simulator does not provide register values to the predictor.

5.1 Why not $E \times D^n$?

Several parameters are available to change the delay D and energy consumption E of a circuit: supply voltage, threshold voltage, transistor sizing, logic design, microarchitecture, etc. In general, there is an anticorrelation between delay and energy: to reduce one quantity (delay or energy), one must increase the other. On the Pareto-optimal curve, all the parameters entail equal marginal costs [15, 16, 1], where the marginal cost[7] κ_x of a parameter x is defined as

$$\kappa_x = -\frac{D}{E} \times \frac{\partial E/\partial x}{\partial D/\partial x}$$

which is a dimensionless positive quantity. Among these parameters, supply voltage v can be varied easily, is effective, and its κ_v is relatively easy to estimate [15, 1]. This is what gives meaning to the popular $E \times D^n$ metrics [8]. Consider a point (D_0, E_0) distant from the Pareto-optimal curve, and a region around this point sufficiently small for the marginal cost κ_v of supply voltage to be constant. This means that varying the supply voltage keeps the quantity $E \times D^{\kappa_v}$ constant in the region. If we can reduce $E \times D^{\kappa_v}$ by varying another parameter (e.g., microarchitecture), then we can adjust supply voltage to reach a point (D_0, E) with $E < E_0$ or, equivalently, a point (D, E_0) with $D < D_0$.

However, κ_v varies significantly with supply voltage [15, 1]. Therefore, the $E \times D^{\kappa_v}$ metric makes sense only for tiny delay and energy variations. We prefer to use a different metric for the championship. Instead of assuming a constant κ_v, we model the relation between supply voltage and clock frequency, and consider IPC speedup when core power is made equal to that of a reference core by adjusting voltage and frequency.

5.2 Clock frequency vs. voltage

We base our model of frequency vs. voltage on the assumption that the transconductance g_m of transistors is approximately constant, independent of voltage:

$$g_m = \frac{dI}{dv}$$

where I is the transistor saturation current and v is the supply voltage. Assuming a constant g_m is an approximation, but it seems to be accurate over a relatively large voltage range (Figure 14b in [13]). A constant g_m implies

$$I = I_0 + g_m(v - v_0)$$

where v_0 is a reference voltage and I_0 is the saturation current at the reference voltage. We further assume that the gate delay is proportional to v/I and that the clock frequency is inversely proportional to the gate delay. Therefore,

$$\frac{f}{f_0} = \frac{I}{v} \times \frac{v_0}{I_0} = \alpha - (\alpha - 1)\frac{v_0}{v} \tag{8}$$

where f_0 is the clock frequency at the reference voltage and α is defined as

$$\alpha = \frac{g_m v_0}{I_0}$$

Assuming a "5 nm" process [4, 13], $v_0 = 0.75\,\mathrm{V}$, $I_0 = 60\,\mu\mathrm{A}$ and $g_m = 130\,\mu\mathrm{A\,V^{-1}}$, we obtain $\alpha = 1.625$.

[7] Marginal cost is a term borrowed from economics. It was used by Azizi et al. [1]. Zyuban and Strenski call it *intensity*, e.g., *hardware intensity* and *voltage intensity* [15, 16].

5.3 Voltage-frequency-scaled speedup

Let C^* be a reference core at the reference voltage v_0 and frequency f_0. It executes IPC^* instructions per cycle, with energy per instruction EPI^*. Its power consumption is $P^* = \text{EPI}^* \times \text{IPC}^* \times f_0$.

Let C be the core whose energy efficiency we seek to determine. It executes IPC instructions per cycle with energy per instruction EPI at the reference voltage v_0. Its power consumption at frequency f_0 and voltage v_0 is

$$P = \text{EPI} \times \text{IPC} \times f_0 \tag{9}$$

We assume that the IPC does not vary with clock frequency.[8] We define the *voltage-frequency-scaled speedup* (VFS) of core C relative to C^* as

$$\text{VFS}_{C^*}(C) = \frac{f \times \text{IPC}}{f_0 \times \text{IPC}^*}$$

where clock frequency f and voltage v have been adjusted so that the power consumption of core C is equal to that of core C^*:

$$\text{EPI} \times \left(\frac{v}{v_0}\right)^2 \times \text{IPC} \times f = P^*$$

Here, we neglect static power and assume that dynamic energy is proportional to voltage squared. The above equation is equivalent to

$$\frac{P^*}{P} = \frac{f}{f_0} \times \left(\frac{v}{v_0}\right)^2$$

Combining the above equation with (8) gives an equation in v,

$$\alpha \left(\frac{v}{v_0}\right)^2 - (\alpha - 1)\frac{v}{v_0} = \frac{P^*}{P}$$

whose solution is

$$\frac{v}{v_0} = \frac{\alpha - 1}{2\alpha} \times \left(1 + \sqrt{1 + \frac{4\alpha}{(\alpha-1)^2} \times \frac{P^*}{P}}\right)$$

Reinjecting the above formula in Equation (8) yields

$$\text{VFS}_{C^*}(C) = \frac{\text{IPC}}{\text{IPC}^*} \times \alpha \times \left(1 - \frac{2}{1 + \sqrt{1 + \beta \times \frac{P^*}{P}}}\right) \tag{10}$$

with β defined as

$$\beta = \frac{4\alpha}{(\alpha-1)^2}$$

Assuming $\alpha = 1.625$ gives $\beta = 16.64$.

To increase the VFS, one must increase the IPC or decrease power consumption. Note that the VFS is equal to the plain IPC speedup $\text{IPC} / \text{IPC}^*$ when $P = P^*$. In particular, $\text{VFS}_{C^*}(C^*) = 1$.

[8] This is approximately true for workloads whose instructions and data fit in the on-chip caches.

5.4 VFS-optimal curve

Equation (7) is based on the assumption that the IPC is bottlenecked by the CBP. This implies that the core's complexity and power is no fixed but depends on the CBP. We need a model of core power as a function of IPC. To this end, we introduce the concept of *VFS-optimal* curve.

The iso-VFS contour $\mathscr{V}_C$ of core C is the set of points $C' = (\text{IPC}', P')$ such that $\text{VFS}_C(C') = 1$. $\mathscr{V}_C$ corresponds to the following equation:

$$\frac{P'}{P} = \frac{\text{IPC}'}{\text{IPC}} \times \left(\frac{\alpha - \frac{\text{IPC}}{\text{IPC}'}}{\alpha - 1} \right)^2 \qquad \text{with } \text{IPC}' \geq \frac{\text{IPC}}{\alpha}$$

A curve $P = h(\text{IPC})$ is *VFS-optimal* if, for every core $C = (\text{IPC}, P)$ on the curve, the iso-VFS contour $\mathscr{V}_C$ is tangent to the curve at C. A VFS-optimal curve defines a set of cores with a common voltage v_0 and frequency f_0 such that each core is optimal for its power budget *among cores on the curve*. That is, $\text{VFS}_C(C') < 1$ for any two distinct cores C and C' on the curve. The slope of $\mathscr{V}_C$ at C is

$$\frac{dP}{d\,\text{IPC}} = \frac{P}{\text{IPC}} \times (\gamma + 1) \tag{11}$$

where γ is defined as

$$\gamma = \frac{2}{\alpha - 1}$$

We have $\gamma = 3.2$ for $\alpha = 1.625$. The ordinary differential equation (11) defines the shape of VFS-optimal curves. Solving this equation yields the VFS-optimal curve containing the reference core $C^* = (\text{IPC}^*, P^*)$:

$$\frac{P}{P^*} = \left(\frac{\text{IPC}}{\text{IPC}^*} \right)^{\gamma+1}$$

The above equation is equivalent to

$$\frac{\text{EPI}}{\text{EPI}^*} = \left(\frac{\text{IPC}}{\text{IPC}^*} \right)^{\gamma} \tag{12}$$

where we have used (9).

We use Equation (12) to model the core energy as a function of IPC. Equation (12) is not guaranteed to be realistic.[9] It is chosen for its mathematical property that $\text{VFS}_{C^*}(C)$ is maximum when $C = C^*$. The VFS drives the competing predictors towards the characteristics of the reference predictor. This is why the reference predictor should not be a predictor that we, the organizers, know how to make, but one that we would like to have.

5.5 Core EPI model and VFS formula

Our core EPI model is based on several assumptions. A wrong-path instructions consumes, on average, less energy than a correct-path instruction, as wrong-path instructions are in intermediate pipeline stages between branch prediction and execution when a branch misprediction recovery occurs. A first assumption of our EPI model is that the average core energy e_{wp} of a wrong-path instruction is half the average core energy e_{cp} of a correct correct-path instruction:

$$e_{wp} = \frac{e_{cp}}{2}$$

[9] It is qualitatively similar to Pollack's rule[9, 3], but with a larger exponent.

The energy of a correct-path instruction is the sum of two contributions, the energy EPI_{cbp} spent in the CBP and the energy e_{core} spent elsewhere:

$$e_{cp} = e_{core} + \mathrm{EPI}_{cbp}$$

A second assumption is that the average prediction throughput on the wrong path equals the average prediction throughput IPC_{cbp} on the correct path. The total number N_{wp} of wrong-path instructions that have entered the instruction pipeline is

$$N_{wp} = T_{wp} \times \mathrm{IPC}_{cbp}$$

The number WPI of wrong-path instructions per correct-path instruction is

$$\mathrm{WPI} = \frac{N_{wp}}{N_{cp}} = \mathrm{IPC}_{cbp} \times \mathrm{CPI}_{cbp} \tag{13}$$

The total energy per correct-path instruction is

$$\mathrm{EPI} = \frac{N_{cp} \times e_{cp} + N_{wp} \times e_{wp}}{N_{cp}} = (e_{core} + \mathrm{EPI}_{cbp}) \times \left(1 + \frac{\mathrm{WPI}}{2}\right) \tag{14}$$

Applying Equation (14) to the reference core gives

$$\lambda + \mu = \frac{1}{1 + \frac{\mathrm{WPI}^*}{2}} \tag{15}$$

where we define λ and μ as

$$\lambda = \frac{e^*_{core}}{\mathrm{EPI}^*} \qquad\qquad \mu = \frac{\mathrm{EPI}^*_{cbp}}{\mathrm{EPI}^*}$$

A third assumption is that, except for the CBP, the core is identical to the core lying on the VFS-optimal curve (12) and having the IPC defined by Equation (7). A fourth assumption is that, for a core on the VFS-optimal curve, e_{core} represents a fixed fraction λ of the EPI. Combining these two assumptions gives

$$e_{core} = \lambda \times \mathrm{EPI}^* \times \left(\frac{\mathrm{IPC}}{\mathrm{IPC}^*}\right)^{\gamma}$$

Reinjecting the above in (14) gives

$$\frac{\mathrm{EPI}}{\mathrm{EPI}^*} = \left[\lambda \times \left(\frac{\mathrm{IPC}}{\mathrm{IPC}^*}\right)^{\gamma} + \mu \times \frac{\mathrm{EPI}_{cbp}}{\mathrm{EPI}^*_{cbp}}\right] \times \left(1 + \frac{\mathrm{WPI}}{2}\right) \tag{16}$$

Finally, the normalized EPI (16) is entered into Equation (10), using (9):

$$\mathrm{VFS} = \frac{\mathrm{IPC}}{\mathrm{IPC}^*} \times \alpha \times \left(1 - \frac{2}{1 + \sqrt{1 + \beta \times \frac{\mathrm{EPI}^*}{\mathrm{EPI}} \times \frac{\mathrm{IPC}^*}{\mathrm{IPC}}}}\right)$$

The IPC speedup is obtained from (7):

$$\frac{\mathrm{IPC}}{\mathrm{IPC}^*} = \frac{\mathrm{IPC}_{cbp}}{\mathrm{IPC}^*_{cbp}} \times \frac{1 + \mathrm{WPI}^*}{1 + \mathrm{WPI}}$$

WPI and WPI^* are obtained from (13), μ is given (e.g., $\mu = 0.05$) and λ is obtained from (15).

References

[1] O. Azizi, A. Mahesri, B. C. Lee, S. J. Patel, and M. Horowitz. Energy-performance tradeoffs in processor architecture and circuit design: a marginal cost analysis. In *International Symposium on Computer Architecture (ISCA)*, 2010.

[2] L. Eeckhout. Use equal-work or equal-time speedup, not geomean speedup. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2025.

[3] E. Grochowski and M. Annavaram. Energy per instruction trends in Intel microprocessors. *Technology@ Intel Magazine*, March 2006.

[4] IRDS. International Roadmap for Devices and Systems - More Moore. https://irds.ieee.org/editions/2021, 2021.

[5] D. A. Jimenéz, S. W. Keckler, and C. Lin. The impact of delay on the design of branch predictors. In *International Symposium on Microarchitecture (MICRO)*, 2000.

[6] G. H. Loh. Revisiting the performance impact of branch predictor latencies. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2006.

[7] P. Michaud. *HARCOM: hardware complexity model for microarchitecture exploration*. https://gitlab.inria.fr/pmichaud/harcom.

[8] P. I. Pénzes and A. J. Martin. Energy-delay efficiency of VLSI computations. In *Great Lakes Symposium on VLSI (GLSVLSI)*, 2002.

[9] F. Pollack. New microarchitecture challenges in the coming generations of CMOS process technologies. In *International Symposium on Microarchitecture (MICRO)*, 1999. Keynote presentation.

[10] G. Reinman, B. Calder, and T. Austin. Fetch directed instruction prefetching. In *International Symposium on Microarchitecture (MICRO)*, 1999.

[11] A. Seznec. A new case for the TAGE predictor. In *International Symposium on Microarchitecture (MICRO)*, 2011.

[12] A. Seznec. TAGE-SC for CBP2025. In *6th Championship Branch Prediction (CBP2025)*, 2025. https://ericrotenberg.wordpress.ncsu.edu/cbp2025/.

[13] V. Vashishtha and L. T. Clark. ASAP5: a predictive PDK for the 5 nm node. *Microelectronics Journal*, 126, August 2022.

[14] C. Wilkerson and J. Stark. Introduction to the JILP's special edition for finalists of the Championship Branch Prediction (CBP1) competition. *Journal of Instruction-Level Parallelism*, 7, April 2005. http://www.jilp.org/vol7/v7paper5.pdf.

[15] V. Zyuban and P. Strenski. Unified methodology for resolving power-performance tradeoffs at the microarchitectural and circuit level. In *International Symposium on Low Power Electronics and Design (ISLPED)*, 2002.

[16] V. Zyuban and P. Strenski. Power performance tradeoffs in design for SoCs. Research Report RC23026, IBM, December 2003. https://dominoweb.draco.res.ibm.com.