CS211 Advanced Computer Architecture Paper Reading 1

Anonymous

1 Paper Information

J. Sartori, B. Ahrens and R. Kumar, "Power balanced pipelines," IEEE International Symposium on High-Performance Comp Architecture, 2012, pp. 1-12, doi: 10.1109/HPCA.2012.6169032.

2 Paper Intended Solve Problem

Traditionally, processor use Balancing Processor Pipelines For Delays policy to maximize instruction throughput, but it will cause significant energy ineffciency, due to the fact that each microarchitectural pipline stage share the same time to complete regardless of its size or complexity, which the ratio between the power consumption of the least power stage and the highest power stage can be as high as 1-2 orders of magnitude.

The author implemete a new policy that Balancing Piplines Power, which are supposed to find an optimized solution that re-distributed cycle time from low-power stages to high power stages, enabling power reduction in the high power stages at the expense of power increase in the low power stages, while preserving processor throughput and minimizing total process power.

3 Solution:

The author propose Power Balanced Microarchitectural Pipelines, as deliberately unbalancing delays of different piplines stage to reduce the disparity in the power consumption of them can significantly reduce the total power of a processor without affecting processor throughput.

3.1 Theoretical Calculation:

We assumpt that:

$$Lower\ Power\ Stage : S_{lo}\ with\ power\ P_{lo},\ increased\ by\ \Delta V_{lo}$$

$$Higher\ Power\ Stage : S_{hi}\ with\ power\ P_{hi},\ decresed\ by\ \Delta V_{hi}$$

$$\Delta V_{lo} + \Delta V_{hi} < 0$$

We have already known the equation:

$$P + \Delta P = (\frac{V + \Delta V}{V})^n P$$

so:

$$\Delta P = ((\frac{V + \Delta V}{V})^n - 1)P$$

We get the leak power and the dynamic power:

$$\Delta P_{leak} = \Delta P(n = 1) = P_{leak}(\frac{\Delta V}{V})$$

$$\Delta P_{dyn} = \Delta P(n = 2) = P_{dyn}(\frac{\Delta V}{V})(2 + \frac{\Delta V}{V})$$

We get the total changed power:

$$\Delta P_{total} = P_{leak} + P_{dyn} = (\frac{\Delta V}{V})(P_{leak} + P_{dyn}(2 + \frac{\Delta V}{V}))$$

Take ths equation back to the inequality describing a trade that reduce total power:

$$\left|\frac{\frac{\Delta V_{hi}}{V_{hi}}}{\frac{\Delta V_{lo}}{V_{lo}}}\right| > \frac{P_{lo,leak} + P_{lo,dyn}(2 + \frac{\Delta V_{lo}}{V})}{P_{hi,leak} + P_{hi,dyn}(2 + \frac{\Delta V_{hi}}{V})}$$

Since the initial voltages of each stage is equal:

$$\left|\frac{\Delta V_{hi}}{\Delta V_{lo}}\right| > \frac{P_{lo,leak} + P_{lo,dyn}(2 + \frac{\Delta V_{lo}}{V})}{P_{hi,leak} + P_{hi,dyn}(2 + \frac{\Delta V_{hi}}{V})}$$

We assume that $|\Delta V_{hi}| \approx |\Delta V_{lo}|$

$$P_{hi,leak} + P_{hi,dyn}(2 + \frac{\Delta V}{V}) > P_{lo,leak} + P_{lo,leak}(2 + \frac{\Delta V}{V})$$

$$P_{hi,total} + \kappa P_{hi,dyn} > P_{lo,total} + \kappa P_{lo.dyn}$$

So we get the theoretical calculation support: the power balancing typically results in processor power savings for the same performance when the power of the time stealing pipeline stage is greater than the power of the time donating stage. The greater the power differential between the two pipeline stages, the more power is reduced when cycle time is redistributed from the low-power stage to the high-power stage.

3.2 Methodology:

The author use FP benchmarks primarily to evaluate dynamic power balancing, since the FabScalar architecture does not contain a FPU.
The author compare power balanced pipelines against two different baselines.

- First baseline is a conventional design that has been leakage optimized by CAD flow for minimum power.
- Second baseline that takes the original synthesized, placed, and routed design and performs cycle stealing to maximize the frequency of the processor. The evaluate power balancing at the highest frequency achievable by the cycle stealing performance-maximized baseline compare with the second baseline.

The author evaluate the power and performance of designs at different voltages between 1.2V and 0.4V, at 0.01V intervals.

3.3 Theoratical Implemetation:

The author split the implemetation into two parts, static implemetation and dynamic implemetation. While static power balancing can be performed either at design time(Pre-Silicon), or after manufacturing(Post-Silicon).

3.3.1 Design-Level Power Balacing

Be incorporated at the design-level implemetation of processor, take in hardware description(RTL) of pipline, operating frequency(f), and the number of allowable voltage domains(N_v), chooses the implementation and voltage for each microarchitectural pipeline stage such that processor power is minimized and the throughput target is met, and performs synthesis, placement, and routing (SPR) to implement the power balanced pipeline.

- First implements each processor pipeline stage for a range of possible timing constraints, and selects the minimumpower implementation of each stage that meets timing (1/f).
- Then, the heuristic performs cycle stealing between the stages to reduce power by selecting lower power versions of high-power stages and selecting higher power versions of low-power stages to satisfy cycle stealing constraints.
- Evaluate all possible design choices and select the pipeline implementation with minimum power.

Result:

- The power savings afforded by balancing pipeline power rather than delay can be significant, even when only a single voltage domain is allowed.
- Power savings increase for higher clock periods.
- Power savings increase as more voltage domains are allowed, but reducing the number of voltage domains does not significantly hinder power savings.
- The power-optimized baseline is more power efficient than a cycle time stealing-based performance maximized baseline for Fabscalar design.

Advantage:

- A single voltage domain design both can void design overheads for additional voltage domains and appreciable hardware overheads.
- Design-level power balancing does not significantly affect processor area.
- Power savings increase as more voltage domains are allowed, but reducing the number of voltage domains does not significantly hinder power savings.
- Power Design level balancing can potentially save power even when cycle stealing cannot increase performance.

3.3.2 Post-Silicon Static Voltage Assignment

To determine the most efficient power balancing configuration for a chip, the power and delay of each stage is characterized over the range of possible voltages during testing. An optimization similar to the one described in Design-Level Power Balancing is performed to select the cycle stealing and voltage assignment strategy that minimizes total power. The time required to find a suitable power balancing strategy can be reduced to negligible levels by using an optimization heuristic.

- First, all stages are set to the maximum voltage, such that delay is minimized.
- Then, for each stage, follow the direction of steepest descent by reducing the voltage by a small amount (vstep=0.01V) on the stage that has the highest potential power savings, which continues until no stage can reduce its voltage without breaking a loop constraint.

Result:

- The power savings achieved by the Post-Silicon is not as efficiency as the Design-Level.
- Same as Design-Level, power savings increase for higher clock periods and voltage domains.
- The power-optimized baseline is more power efficient than a cycle time stealing-based performance maximized baseline for Fabscalar design.

Advantage:

- Do not need use a new design flow to create a power balanced pipeline, which can designed normally.
- Can be used to overcome inefficiencies caused by process variations.
- Gradient descent can avoid choosing stages that present significant savings in the short run but consume too much delay in the process.
- Compared with Design-Level, Post-Silicon reduced design time and the potential to achieve additional benefits by adapting to process variations, if they are significant.

3.3.3 Dynamic Power Balancing

The author model an OS-based power balancing mechanism that counts the number of FP instructions committed in an OS timeslice (5 ms), and decides whether power should be re-balanced. The exact mechanism involves using the FP instruction count to reference into a lookup table that stores the voltage and delay assignments for each stage in each configuration. When rebalancing is needed, the OS assigns the stage voltages and delays loaded from the table.

Result:

- The pipeline power breakdown does not vary significantly within or between INT benchmarks, even for a processor with a FPU. The difference in the pipeline power breakdown between INT and FP benchmarks can be significant due to the change in FPU power consumption. Thus, dynamic power balancing may achieve benefits by identifying and adapting to FP and non-FP workloads.

Advantage:

- For processors in which the relative power breakdown between pipeline stages may change due to changes in the workload, dynamic power balancing may afford additional power reduction over static power balancing.
- The potential benefit of dynamic adaptation is mainly in adapting to the differences between INT and FP benchmarks, but not the differences between phases within a FP benchmark.

3.4 Practical Consideration

3.4.1 Cycle Time Stealing

The author use cycle time stealing as the core mechanism to perform power balancing. Cycle time stealing allows a pipeline stage to donate a fraction of its evaluation period (cycle time) to another stage, without affecting the operating frequency of the pipeline. Cycle time stealing re-distributes cycle time from a donating stage (SD) to a receiving stage (SR) by delaying the clock signal at the input flip-flop (FF) of SD (allowing less time to evaluate) and the output FF of SR (allowing more time to evaluate) by the same amount (δ).

For design-level power balancing, the author implement cycle time stealing statically in the clock network during clock tree synthesis. For techniques that require post-silicon adaptation, the author assume the use of tunable delay buffers.

To avoid throughput implications, the total latency of each loop ($s * Tcp$ for an s-stage loop) must remain constant before and after cycle time stealing.

Additional practical issues were considered when donating a large amount of cycle time to a single stage.

- Voltage cannot be decreased indefinitely.
- Voltages of donating stages cannot be increased indefinitely, due to aging considerations, since circuit aging is accelerated at higher voltages.
- All paths in a stage must satisfy a short path or hold time constraint ($Dmin\delta f - \delta i + Thold$). The constraint on the minimum path delay allowable in a stage (Dmin) depends on the amount by which the evaluation time of the stage has been extended ($\delta f - \delta i$) and the FF hold time (T_{hold}).

3.4.2 Local Voltage Scaling

- Routing a unique voltage to each stage can be costly. Rather than a single voltage network feeding the pipeline, a separate network is needed for each stage. In practice, the overhead of creating separate voltage domains can be limited to acceptable levels by restricting the number of allowable voltage domains.

- Voltage level conversion between stages may also be a concern. When a low-voltage stage feeds into a highvoltage stage, the signal from the low-voltage stage may not completely turn off an input transistor in the high-voltage stage, potentially creating a short circuit path. Thus, during optimization, avoid larger voltage differentials between adjacent stages to avoid excessive leakage.

3.4.3 Voltage-Delay Relationship

Power balancing heuristics avoid these regions(where not linear) and choose voltages in the "linear" region of the delay vs. voltage curve.

4 Personal Opinion

4.1 Dynamic Power Balancing Implemetation

For the implementation of Dynamic Power Balancing, using OS-based policy might be inconvenient, since it break the encapsulation of processor architecture, force OS developer to consider the number of FP instruction and modificate voltages of pipline stages. I think it should be solved at hardware(architecture) level which addding a counter of hardware to count FP instructions' number in the processor and a ROM to store configurations. When the number of counter per time slice reach some threshold value search in the configurations and change volage of stages.

Pros

- Package implementation in architecture level, avoid affecting existing operating system, be friendly to programmer.

Cons

- Need to re-design current processors, might increase the area of processors.

4.2 More Flexible Power Balancing

As FPU consumes a large amount of power, and some pipline stages are constitude by several functional units, I think that we can apply power balancing policy into more lower levels. For example, we can individual control the power of the FPU units instead of the whole ALU pipline stage. While other stages like IF, DE, WB in RISCV piplines are not easy to be controlled, or with extra ineffciency hardware overhead. It might be more efficiency to apply power balancing policy on some key pipline stages.