Introducing Google Axion Processors, our new Arm-based CPUs

April 9, 2024. Amin Vahdat. “…Today, we are thrilled to announce the latest incarnation of this work: Google Axion Processors, our first custom Arm®-based CPUs designed for the data center. Axion delivers industry-leading performance and energy efficiency and will be available to Google Cloud customers later this year.”
Announcements

• Homework 4 due this week
  • Quiz 4 next Tuesday, in class

• Project
  • Good preliminary design review on Tuesday
  • Pay attention to integration with other teams!
  • Final presentations: May 2, 9am-12pm

• Final exam: April 26, in class
Multiple Supplies
Multiple Supply Voltages

• Block-level supply assignment (“power domains” or “voltage islands”)
  • Higher throughput/lower latency functions are implemented in higher $V_{DD}$
  • Slower functions are implemented with lower $V_{DD}$
  • Often called “Voltage islands”
  • Separate supply grids, level conversion performed at block boundaries

• Multiple supplies inside a block
  • Non-critical paths moved to lower supply voltage
  • Level conversion within the block
  • Physical design challenging
  • (Not used in practice)
Utilize Unified Power Format (UPF) to capture design intent
Common Power Format (CPF) is similar
There are primarily 3 power domains –
- Logic inside aon_wrapper [but not inside aon_pgd_wrapper] is always-on.
- Logic inside pgd_wrapper can be power gated.
- Logic inside aon_pgd_wrapper can be power gated but won’t be power gated when pgd_wrapper is powered ON.

There are two voltage domains –
- The supply voltage to logic inside aon_wrapper [but not inside aon_pgd_wrapper] and logic inside pgd_wrapper is 0.9V.
- The supply voltage to logic inside aon_pgd_wrapper is 1.1V.

There are two registers – reg A and reg B. The state of reg A needs to be retained in power gated state.

There are six signals sig1-sig6 coming to and from different logic blocks.
Create Power Domains

- `create_power_domain pd_top -include_scope`
- `create_power_domain pd_aon -elements {aon_wrapper}`
- `create_power_domain pd_gated -elements {pgd_wrapper}`
- `create_power_domain pd_gated_aon -elements`
  ```
  {aon_wrapper/aon_pgd_wrapper}
  ```

There are primarily 3 power domains –

- Logic inside `aon_wrapper` [but not inside `aon_pgd_wrapper`] is always-on.
- Logic inside `pgd_wrapper` can be power gated.
- Logic inside `aon_pgd_wrapper` can be power gated but won’t be power gated when `pgd_wrapper` is powered ON.

https://vlsitutorials.com/upf-low-power-vlsi/
Unified Power Format

# Create Supply Ports
create_supply_port VCCL -direction in -domain pd_top
create_supply_port VCCH -direction in -domain pd_top
create_supply_port GND -direction in -domain pd_top

# Create Supply Nets
create_supply_net VCCL -domain pd_top
create_supply_net VCCH -domain pd_top
create_supply_net GND -domain pd_top
create_supply_net VCCL -domain pd_aon -reuse
create_supply_net GND -domain pd_aon -reuse
create_supply_net VCCH -domain pd_gated_aon -reuse
create_supply_net VCCH_gated -domain pd_gated_aon
create_supply_net GND -domain pd_gated_aon -reuse
create_supply_net VCCL -domain pd_gated -reuse
create_supply_net VCCL_gated -domain pd_gated
create_supply_net GND -domain pd_gated -reuse

There are two voltage domains –
- The supply voltage to logic inside aon_wrapper [but not inside aon_pgd_wrapper] and logic inside pgd_wrapper is 0.9V.
- The supply voltage to logic inside aon_pgd_wrapper is 1.1V.

https://vlsitutorials.com/upf-low-power-vlsi/
# Connect Supply Nets with corresponding Ports
connect_supply_net VCCL -ports VCCL
connect_supply_net VCCH -ports VCCH
connect_supply_net GND -ports GND

# Establish Connections
set_domain_supply_net pd_top -primary_power_net VCCL -primary_ground_net GND
set_domain_supply_net pd_aon -primary_power_net VCCL -primary_ground_net GND
set_domain_supply_net pd_gated_aon -primary_power_net VCCH_gated -primary_ground_net GND
set_domain_supply_net pd_gated -primary_power_net VCCL_gated -primary_ground_net GND

https://vlsitutorials.com/upf-low-power-vlsi/
# Shut-Down Logic for pgd_wrapper & aon_pgd_wrapper

```bash
create_power_switch sw_pgd_wrapper
-domain pd_gated
-input_supply_port "sw_VCCL VCCL"
-output_supply_port "sw_VCCL_gated VCCL_gated"
-control_port "sw_pgd_en aon_wrapper/pmu/pgd_en"
-on_state "SW_PGD_ON sw_VCCL {!sw_pgd_en}"

create_power_switch sw_aon_pgd_wrapper
-domain pd_gated_aon
-input_supply_port "sw_VCCH VCCH"
-output_supply_port "sw_VCCH_gated VCCH_gated"
-control_port "sw_aon_pgd_en aon_wrapper/pmu/aon_pgd_en"
-on_state "SW_AONPGD_ON sw_VCCH {!sw_aon_pgd_en}"
```

https://vlsitutorials.com/upf-low-power-vlsi/
# Isolation strategy

```plaintext
set_isolation isol_clamp1_sig_from_pgd
  -domain pd_gated
  -isolation_power_net VCCL
  -isolation_ground_net GND
  -clamp_value 1
  -elements {pgd_wrapper/sig2}

set_isolation_control isol_clamp1_sig_from_pgd
  -domain pd_gated
  -isolation_signal aon_wrapper/pmu/isol_pgd_en
  -isolation_sense low
  -location parent

set_isolation isol_clamp0_sig_from_pgd
  -domain pd_gated
  -isolation_power_net VCCL
  -isolation_ground_net GND
  -clamp_value 0
  -elements {pgd_wrapper/sig4}

set_isolation_control isol_clamp0_sig_from_pgd
  -domain pd_gated
  -isolation_signal aon_wrapper/pmu/isol_pgd_en
  -isolation_sense low
  -location parent
```

https://vlsitutorials.com/upf-low-power-vlsi/
# Level Shifter strategy

set_level_shifter LtoH_sig_to_aonpgd
- domain pd_gated_aon
- applies_to inputs
- rule low_to_high
- location self

set_level_shifter HtoL_sig_from_aonpgd
- domain pd_gated_aon
- applies_to outputs
- rule high_to_low
- location self

https://vlsitutorials.com/upf-low-power-vlsi/
There are two registers – reg A and reg B. The state of reg A needs to be retained in power gated state.

https://vlsitutorials.com/upf-low-power-vlsi/
# Create Power State Table

```plaintext
add_port_state VDDH 
-state {HighVoltage 1.1}
add_port_state VDDL 
-state {LowVoltage 0.9}
add_port_state sw_aon_pgd_wrapper/sw_VCCH_gated 
-state {HighVoltage 1.1} 
-state {aonpgd_off off}
add_port_state sw_pgd_wrapper/sw_VCCL_gated 
-state {LowVoltage 0.9} 
-state {pgd_off off}
create_pst pwr_state_table 
-supplies {VCCH VCCL VDDH_gated VDDL_gated}
add_pst_state PRE_BOOT 
-pst pwr_state_table 
-state { HighVoltage LowVoltage aonpgd_off pgd_off}
add_pst_state AONPGD_ON 
-pst pwr_state_table 
-state { HighVoltage LowVoltage aonpgd_off pgd_off}
add_pst_state PGD_ON 
-pst pwr_state_table 
-state { HighVoltage LowVoltage HighVoltage pgd_off}
add_pst_state ALL_ON 
-pst pwr_state_table 
-state { HighVoltage LowVoltage HighVoltage LowVoltage}
```
Practical Examples

• Intel Skylake (ISSCC’16)
  • Four power planes indicated by colors
Practical Examples

- Intel 28-core Skylake-SP (ISSCC’18)

- 9 primary VCC domains are partitioned into 35 VCC planes
Leakage Issue

- Driving from $V_{DDL}$ to $V_{DDH}$

Level converter

\[ |V_{GS}| > V_{Th} \]

\[ V_{OH} < V_{DDH} \]

\[ I_{Leak} \]
Multiple Supplies Within A Block

• Downsizing, lowering the supply on the critical path will lower the operating frequency

• Downsize (lowering supply) non-critical paths
  • Narrows down the path delay distribution
  • Increases impact of variations
Multiple Supplies in a Block

Conventional Design

CVS Structure

Critical Path

Critical Path

Lower $V_{DD}$ portion is shaded

“Clustered voltage scaling”

M. Takahashi, ISSCC’98.
Multiple Supplies in a Block

CVS

Layout:

Usami'98
Level-Converting Flip-Flop
Dynamic Voltage-Frequency Scaling (DVFS)
# Power /Energy Optimization Space

<table>
<thead>
<tr>
<th>Energy</th>
<th>Constant Throughput/Latency</th>
<th>Variable Throughput/Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Design Time</td>
<td>Sleep Mode</td>
</tr>
<tr>
<td>Active</td>
<td>Logic design</td>
<td>Clock gating</td>
</tr>
<tr>
<td></td>
<td>Scaled $V_{DD}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Trans. sizing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Multi-$V_{DD}$</td>
<td></td>
</tr>
<tr>
<td>Leakage</td>
<td>Stack effects</td>
<td>Sleep T’s</td>
</tr>
<tr>
<td></td>
<td>Trans sizing</td>
<td>Multi-$V_{DD}$ Variable $V_{Th}$</td>
</tr>
<tr>
<td></td>
<td>Scaling $V_{DD}$</td>
<td>+ Input control</td>
</tr>
<tr>
<td></td>
<td>+ Multi-$V_{Th}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DVS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variable $V_{Th}$</td>
</tr>
</tbody>
</table>
Adaptive Supply Voltages

Exploit Data Dependent Computation Times To Vary the Supply

from [Nielsen94]

*(IEEE Transactions on VLSI Systems)*
Processors for Portable Devices

- Dynamic Voltage Scaling

• Eliminate performance ↔ energy trade-off

Performance (MIPS)

Processor Energy (Watt*sec)

PDAs

Pocket-PCs

Notebook Computers

EECS251B L23 DVS

Burd
ISSCC’00
Typical MPEG IDCT Histogram
Processor Usage Model

Desired Throughput

Compute-intensive and low-latency processes

Maximum Processor Speed

System Idle

Background and high-latency processes

System Optimizations:
- Maximize Peak Throughput
- Minimize Average Energy/operation

Burd
ISSCC'00
Common Design Approaches (Fixed VDD)

**Compute ASAP:**
- Excess throughput

**Clock Frequency Reduction:**
- $f_{CLK}$ Reduced
- Energy/operation remains unchanged ...
- while throughput scaled down with $f_{CLK}$

Delivered Throughput vs. time
Scale $V_{DD}$ with Clock Frequency

Constant supply voltage

$\sim10x$ Energy Reduction

Reduce $V_{DD}$, slow circuits down.

Throughput ($\propto f_{CLK}$)

Energy/operation

1.1V

3.3V

Burd
ISSCC'00
CMOS Circuits Track Over $V_{DD}$

Normalized max. $f_{CLK}$

Delay tracks within +/- 10%

Burd
ISSCC'00
Dynamic Voltage Scaling (DVS)

1. Vary $f_{CLK}, V_{DD}$
2. Dynamically adapt

- Dynamically scale energy/operation with throughput.
- Always minimize speed $\rightarrow$ minimize average energy/operation.
- Extend battery life up to 10x with the exact same hardware!

Burd
ISSCC'00
Operating System Sets Processor Speed

- DVS requires a voltage scheduler (VS).
- VS predicts workload to estimate CPU cycles.
- Applications supply completion deadlines.

\[
\frac{\text{CPU cycles}}{\Delta \text{time}} = F_{\text{DESIRED}}
\]
Converter Loop Sets $V_{DD}$, $f_{CLK}$

- Feedback loop sets $V_{DD}$ so that $F_{ERR} \rightarrow 0$.
- Ring oscillator delay-matched to CPU critical paths.
- Custom loop implementation $\rightarrow$ Can optimize $C_{DD}$.

Burd
ISSCC'00
Power Estimator Example for HW-Based DVFS Control

- Receives ongoing counts from each CPU of “significant” events
  - e.g., how many vector instructions executed per cycle, aggregates these to a per-CPU instantaneous power estimate
- Sends per-CPU throttle requests as necessary.
- The number of “power events” over some period of time is tracked
  - If this is too high, the fact is reported to the PMU and PMGR, and the frequency/voltage settings for the offending core(s) are reduced; and similarly if a core has been operating within spec for some period of time, its frequency/voltage is allowed to rise.

<table>
<thead>
<tr>
<th>DVFM State</th>
<th>Volt.</th>
<th>Freq.</th>
<th>Limit3</th>
<th>Limit4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>V0</td>
<td>F0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>V1</td>
<td>F1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>V2</td>
<td>F2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>V3</td>
<td>F3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3rdCoreMax</td>
<td>V3rd</td>
<td>F3rd</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4thCoreMax</td>
<td>V4th</td>
<td>F4th</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

DPE (Digital Power Estimator)
APSC (Automatic Power-State Controller)
DVFS Scheduling

• First stage – establish performance goals (achieve state X by time Y)
  • Use closed-loop performance controller to adjust DVFS to meet these performance goals.
  • E.g. performance described as a monotonic map from n-cores running at max frequency down to 1 core running at minimum frequency.
  • Move up/down this performance map depending on how the target is being met

• Second stage - energy and performance issues.
  • Track short-term power draw and temperature to back-off high-performance goals if needed
  • Track an energy per instruction metric and try to optimize this while not impairing the performance goals
DVFS as a Systems Problem for the OS

• Based on thread groups and matching target metrics to measured metrics, a map of the best performance schedule for the thread group is created (i.e. each thread gets mapped onto a P vs E core at certain voltage/frequency)

• These maps are then read by the thread scheduler (Thread Director on Adler Lake CPU, for example) which puts each thread into an E vs P queue, with the appropriate DVFS info attached to it. The queues are adjusted across different thread groups to ensure maximal occupancy of every core.
Circuit design constraints. (Functional verification)

Circuit delay variation. (Timing verification)

Noise margin reduction. (Power grid, coupling)

Delay sensitivity. (Local power distribution)

Design verification complexity similar to high-performance processor design @ fixed $V_{DD}$
• Cannot use NMOS pass gates – fails for $V_{DD} < 2V_T$.
• Functional verification only needed at one $V_{DD}$ value.
Relative Delay Variation

Delay relative to ring oscillator

Four extreme cases of critical paths:

All vary monotonically with $V_{DD}$.

Dominated by:
- Gate
- Interconnect
- Diffusion
- Series

• Timing verification only needed at min. & max. $V_{DD}$.

Burd
ISSCC’00
Multiple Path Tracking

A. Drake, ISSCC’07

Path delay variation at nominal voltage

<table>
<thead>
<tr>
<th>ps</th>
<th>adder</th>
<th>nor</th>
<th>nand</th>
<th>pass-gate</th>
<th>wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>core 0: σ</td>
<td>3.78</td>
<td>3.45</td>
<td>3.91</td>
<td>3.28</td>
<td>2.82</td>
</tr>
<tr>
<td>core 1: σ</td>
<td>3.55</td>
<td>3.16</td>
<td>2.25</td>
<td>3.55</td>
<td>4.21</td>
</tr>
<tr>
<td>chip: σ</td>
<td>4.09</td>
<td>4.03</td>
<td>3.90</td>
<td>4.80</td>
<td>4.10</td>
</tr>
<tr>
<td>core 0: Δ</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>core 0: Δ</td>
<td>12</td>
<td>9</td>
<td>6</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>core 0: Δ</td>
<td>13</td>
<td>11</td>
<td>14</td>
<td>16</td>
<td>15</td>
</tr>
</tbody>
</table>
Multiple Path Tracking

Tunable replica circuit

Path type  Tunable delay  Polarity

selectable INV/NAND/NOR/INV delay

tunable INV delay


Cho, ISSCC'16
Tracking with SRAM in Critical Path

Mismatch between logic and SRAM

\[ \Delta V_{BL} = \frac{V_{DD}}{2 \cdot n} \]

\[ V_{DD} = 0.6V \]

\[ V_{DD} = 1.1V \]

\[ V_{OS} = 0.15V \]

SRAM multiplicative replica

Niki, JSSC’11
Alternative: Error Detection

Bull, ISSCC’2010
Design for Dynamically Varying VDD

• Static CMOS logic.

• Ring oscillator.

• Dynamic logic (& tri-state busses).

• Sense amp (& memory cell).

Max. allowed $|dV_{DD}/dt|$ $\rightarrow$ Min. $C_{DD} = 100\text{nF (0.6mm)}$

Circuits continue to properly operate as $V_{DD}$ changes
Static CMOS Logic

- Static CMOS robustly operates with varying $V_{DD}$.

**Equation:**
$V_{in} = 0 \quad V_{out} = V_{DD}$

**Diagram:**
- $r_{ds\mid PMOS}$
- $C_L$
- $V_{out}$

**Constraints:**
- $0.6\text{mm CMOS: } |dV_{DD}/dt| < 200\text{V/ms}$

**Graph:**
- The graph shows a circuit diagram with $V_{DD}$, $V_{in}$, $V_{out}$, and $C_L$ components, along with the condition $r_{ds\mid PMOS}$ and $\tau = 4\text{ns}$.
Ring Oscillator

- Output $f_{CLK}$ instantaneously adapts to new $V_{DD}$. 

Simulated with $dV_{DD}/dt = 20V/\mu s$
Dynamic Logic

0.6mm CMOS: $|dV_{DD}/dt| < 20$V/ms

- Cannot gate clock in evaluation state.
- Tri-state busses fail similarly → Use hold circuit.

Errors

- False logic low: $DV_{DD} > V_{TP}$
- Latch-up: $DV_{DD} > V_{be}$

\[ \Delta V_{DD} = V_{DD} - V_{out} \]

\[ -\Delta V_{DD} \]
Dynamic operation can increase energy efficiency > 10x.
**V\textsubscript{DD}-Hopping**

**Normalized power**

- 0
- 0.2
- 0.4
- 0.6
- 0.8
- 1

**# of frequency levels**

- Transition time between $f$ levels $= 200\mu s$

**Time**

- n-th slice finished here
- Next milestone

**Application slicing and software feedback guarantee real-time operation.**

**MPEG-4 encoding**

- Two hopping levels are sufficient.
Dithering Between Supply Levels

- Done with switched-capacitor DC-DC converters which efficiently work only at discrete levels

Keller et al, ESSCIRC’16
Dithering Between Supply Levels

- Dithering fills in between fixed DC-DC modes

Keller et al, ESSCIRC’16
Summary

- Multiple supply voltages
- Dynamic voltage scaling
Next Lecture

• Low-power design
  • Clock gating
  • Leakage management