TLM Platforms Re-Define Hardware Virtualization

Jon McDonald
Why Build Transaction Level Platform?

- The higher you go toward system-level abstraction, the greater the leverage
  - Optimize system architecture for area, performance and power
  - Enable early software validation/debug against fast abstracted model
  - Reduce verification time
    - By quickly building and validating a transaction level model
    - By reusing the model for validating the implementation

Source: 3rd party ESL survey, Jan 2009
System Architecture Optimization

- Optimizing those **system architecture attributes** impacting area, timing and power
  - **10X power variation** resulted from system architecture tradeoffs in the following example

<table>
<thead>
<tr>
<th>Transmitter Design (IFFT Block)</th>
<th>Min. Freq. to Achieve Req. Rate</th>
<th>Average Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combinational</td>
<td>1.0 MHz</td>
<td>3.99</td>
</tr>
<tr>
<td>Pipelined</td>
<td>1.0 MHz</td>
<td>9.92</td>
</tr>
<tr>
<td>Folded (16 bfy4s)</td>
<td>1.0 MHz</td>
<td>~X10</td>
</tr>
<tr>
<td>Folded (8 bfy4s)</td>
<td>1.5 MHz</td>
<td>~140</td>
</tr>
<tr>
<td>Folded (4 bfy4s)</td>
<td>3.0 MHz</td>
<td>21.10</td>
</tr>
<tr>
<td>Folded (2 bfy4s)</td>
<td>6.0 MHz</td>
<td>34.6</td>
</tr>
<tr>
<td>Folded (1 bfy4)</td>
<td>12.0 MHz</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: IEEE 802.11a System Architecture

Only Transaction Level Platform Allows Users to Quickly Modify and Evaluate Various System Architectures

Source: Chip Design Magazine ESL Synthesis + Power Analysis By Holly Stump and George Harper
Hardware/Software Optimization

- Addressing **hardware/software architecture tradeoffs**
  - **2X power variation** resulted from hardware/software tradeoffs in the following example

Table 1: Results from power profiling of candidate architectures for the IEEE 802.11 MAC processor

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Design Power (mW)</th>
<th>Power Profiling (mW)</th>
<th>Relative Error (%)</th>
<th>Area variation (mW)</th>
<th>Area variation (mW)</th>
<th>Power Profiling (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>42 SW Single bus</td>
<td>389.7</td>
<td>389.8</td>
<td>0.05</td>
<td>1.8</td>
<td>0.4</td>
<td>0.21</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>454.9</td>
<td>454.8</td>
<td>0.80</td>
<td>4.1</td>
<td>2.2</td>
<td>0.35</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>543.0</td>
<td>543.9</td>
<td>0.18</td>
<td>24.0</td>
<td>4.1</td>
<td>0.29</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>592.9</td>
<td>592.9</td>
<td>0.55</td>
<td>13.0</td>
<td>3.4</td>
<td>0.42</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>614.0</td>
<td>614.0</td>
<td>0.18</td>
<td>45.0</td>
<td>7.5</td>
<td>0.23</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>621.5</td>
<td>621.5</td>
<td>0.45</td>
<td>41.8</td>
<td>6.2</td>
<td>0.37</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>512.2</td>
<td>512.2</td>
<td>0.97</td>
<td>19.0</td>
<td>3.2</td>
<td>0.22</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>587.4</td>
<td>587.4</td>
<td>0.22</td>
<td>4.4</td>
<td>2.2</td>
<td>0.37</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>672.2</td>
<td>672.2</td>
<td>0.00</td>
<td>28.8</td>
<td>3.5</td>
<td>0.28</td>
</tr>
<tr>
<td>42 SW Two buses</td>
<td>765.5</td>
<td>765.5</td>
<td>0.48</td>
<td>23.3</td>
<td>4.3</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Source: Fast System Level Power Profiling for Battery Efficient System Design Kanishka Lahiri Dept. of ECE UC San Diego, Anand Raghunath C&C Research Labs NEC USA

Only Transaction Level Platform Allows Users to Quickly Modify and Evaluate Various Hardware/Software Configurations
Software Compiler Dependent Power

- Image Recognition Package running on an ARM
  - O0 - No Optimizations
  - O3 - Rename registers, inline functions
- Different compiler options may result in different profiles (cache access)

Source: POWER ESTIMATION AND POWER OPTIMIZATION POLICIES FOR PROCESSOR-BASED SYSTEMS José L. Ayala Rodrigo Universidad Politécnica de Madrid
Example Architecture: Latency

[Graph showing latency analysis with time on the x-axis and latency (ns) on the y-axis, with data points indicating minimal, maximal, and average latency values.]
Example Architecture: Power
Example Architecture: Cache, DDR
Vista™ Main Capabilities

- Modeling
- Assembly
- Debug
- Analysis

- Virtual Platform
- Performance (latency, utilization, bandwidth)
- Power

- Policies
- Communication Layer
- Transaction Level Platform
- On Chip Bus
- Processor
- TLM

- Vista Model Builder

- RTL IP

- Software
Vista Scalable TLM Models

- Based on SystemC and the **OSCI TLM 2.0 standard**
- Scalability is accomplished by
  - Modeling the core **Function**
  - Providing the **Communication Layer**
  - Adding a separate **Timing/Power** model
Vista Scalable TLM Timing Model

- Single transaction level timing model supporting
  - LT (loosely time)
  - AT (approximated time)
- GUI based timing definition policies
- Policy types:
  - Delay, Sequential, Split, Pipeline
- Timing values:
  - Wait states, Latency, Data Delay
Vista Scalable TLM Power Model

- Transaction-level modeling of all power types
  - Static (leakage) power
  - Clock tree power
  - Dynamic power (per transaction)

- Dynamic power is assigned to each transaction type

- Vista TLM power model is
  - Reactive to incoming traffic and inner states
  - Supports voltage and frequency scaling

### Power Policies

<table>
<thead>
<tr>
<th>Dynamic Power</th>
<th>Transaction</th>
<th>Power</th>
<th>Time Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB.READ</td>
<td>0.16mw</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>USB.WRITE</td>
<td>0.44mw</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Clock Tree Power</td>
<td>0.5mw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static (Leakage) Power</td>
<td>0.2mw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Architectural Power Optimization Flow

- Model TLM Timing and Power via policies
- Assemble the transaction level reference platform
- Analyze Timing/Power in a system context
- Modify Timing/Power policies or the platform architecture
- Quickly iterate optimizing for timing and power

**Dynamic Power**

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Power</th>
<th>Time Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB.READ</td>
<td>0.16mw</td>
<td>200</td>
</tr>
<tr>
<td>Clock Tree Power</td>
<td>0.5mw</td>
<td></td>
</tr>
<tr>
<td>Static (Leakage) Power</td>
<td>0.2mw</td>
<td></td>
</tr>
</tbody>
</table>

Quick Optimization of Power and Performance before RTL
Lisa,
Please highlight the message at the bottom
Shabtay, 7/22/2009
Vista Wide Range of Analysis Toolsets

- **Functional Analysis**
  - Data ID and Tracing

- **Timing/Performance Analysis**
  - Throughput
  - Latencies
  - Bus Utilization
  - State Distribution

- **Power Analysis**
  - Dynamic, Static and Clock Power
  - Instance Power
  - Mean and Peak values
  - System and software power profiles
  - Hot Spot analysis
  - Voltage and frequency scaling (DVFS)
  - Power domain management
Vista - Optimization Philosophy

✅ Use policies to model power/timing of new IP

✅ Use accurate power/timing models for legacy IP

✅ Analyze power/timing profiles
  - typical system scenarios
  - running software application

✅ Explore various power/timing domain management strategies and voltage/frequency scaling techniques
Vista enables HW/SW Co-Development

- Vista produces SystemC virtual reference platform
  - Available for early software development and validation
  - Accurate enough for tuning SW for power and performance
Vista: Power Design and Optimization at the ESL

- Architectural analysis, exploration and optimization for power/timing
  - Unique power/timing modeling capabilities
  - Wide range of analysis toolsets

- Improved model accuracy enables
  - Analysis of system architecture changes
  - HW/SW trade-off
  - Tuning the application software
  - Correlating performance with system workload