Lab-STICC : Power Consumption Modelling with AADL

Dominique BLOUIN
Skander Turki
Eric SENN
Saâdia Dhouib
11/06/2009
Outline

- Context
  - Review of power estimation methodologies and tools
    - Functional Level Power Analysis (FLPA)

- Power consumption AADL modeling approaches
  - Common property sets for power consumption analysis

- Modeling examples
  - Threads and Processors
  - IPC (sockets and Ethernet devices)
  - ASIC buses

- Consumption Analysis Toolbox (CAT) demo

- Interoperability with CAT (MARTE to AADL model transformation)
### Power Estimation Methodologies and Tools

<table>
<thead>
<tr>
<th>System level power estimation</th>
<th>Multi-Level Power Analysis</th>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithmic level power estimation</td>
<td>Functional Level Power Analysis</td>
<td>CAT / FLPA</td>
</tr>
<tr>
<td>Micro-Architectural level power estimation</td>
<td>Instruction level</td>
<td>SoftExplorer / FLPA</td>
</tr>
<tr>
<td>Circuit level power estimation</td>
<td>Cycle level</td>
<td>Jouletrack / ILPA</td>
</tr>
<tr>
<td></td>
<td>Gate level</td>
<td>PowerTimer</td>
</tr>
<tr>
<td></td>
<td>Transistor level</td>
<td>Wattch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SimplePower</td>
</tr>
</tbody>
</table>

**Tools /Methodologies**
- **CAT / FLPA**
- **SoftExplorer / FLPA**
- **Jouletrack / ILPA**
- **PowerTimer**
- **Wattch**
- **SimplePower**
- **QuickPower**
- **SPICE**

**ILPA: I(prog) = Σ I(instr) + Σ ΔI**

**FLPA: parameterized activity ⇒ Parameterized Models**

*Dominique BLOUIN, Skander TURKI, Eric SENN, Saâdia DHOUIB*  
*Lab-STICC*  
*Brest, 11/06/2009*
**Context**

**Functional Level Power Analysis**

- **FLPA**
  - 3 steps
  - Complex architectures
  - Simples models
  - Precise estimations
  - Processors Models
    - DSP & GPP
    - *SoftExplorer* Tool
  - FPGA Models
  - Flash Memory Models
  - System ...

---

1. **Functional Analysis**

   - Algorithmic parameters
   - Architectural parameters

   - Block 1
   - Block 2
   - Block 3

   - Processor

   

2. **Characterization**

   - Scenario:
     \[ \alpha = 0 \ldots 1 \]

   - Architecture:
     \[ F = 20 \ldots 200 \text{ MHz} \]

   - Block 1 stimulated

   - Block 1
   - Block 2
   - Block 3

   - Processor

---

3. **Model determination**

   - Algorithmic parameters
   - Architectural parameters

   - Processor Model

   - \[ \text{P} = f(\text{parameters}) \]

---

Dominique BLOUIN, Skander TURKI, Eric SENN, Saâdia DHOUIB

*Lab-STICC*

Brest, 11/06/2009
### Context

**Estimation Model Parameters**

- Models for different architectures: DSP, RISC, VLIW, ...
- Uncertainty < 10% from C code, < 4% from asm code

<table>
<thead>
<tr>
<th>Processors</th>
<th>TI C67</th>
<th>TI C64</th>
<th>TI C62</th>
<th>TI C55</th>
<th>ARM7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
<td>floating point VLIW</td>
<td>fixed point VLIW + L1 &amp; L2 cache</td>
<td>fixed point VLIW</td>
<td>low power</td>
<td>RISC</td>
</tr>
<tr>
<td><strong>Architectural</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Clock frequency</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MM</td>
<td>Memory mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DM</td>
<td>Data mapping</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>W</td>
<td>DMA data width</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PM</td>
<td>Power management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Algorithmic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>Parallelism rate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>β</td>
<td>Processing rate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ε</td>
<td>DMA activity rate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>γ</td>
<td>Instruction cache miss rate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>τ</td>
<td>External data access rate</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>μ</td>
<td>Data cache miss rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>σ</td>
<td>Data read rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δ</td>
<td>Data write rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSR</td>
<td>Pipeline Stall rate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Only the high level methodologies are practical to explore the design space.

High level methodologies are component-instance specific.
  - A power estimation model per component is needed.
  - Would be nice if it were shipped with the component, i.e.: provided by manufacturer.

Need for component libraries.
  - Library modeled with the help of AADL packages and property sets.
  - The properties are often specific to the component classifier.
  - The library is added to the OSATE environment using the provided extension points.
Power Consumption AADL Modeling
Core Power Analysis Properties

- Property set CAT_Properties

- Core power consumption properties
  - Power and energy property types and units
    - Apply to all component categories (platform and software).
    - Modeled as ranges to carry uncertainties.

- Time property types and units from existing “AADL_Project” property set.

- Use appropriate existing property definition (e.g.: Compute_Execution_Time for threads and Transmission_Time for buses).

- Added time per byte property type and units
Power Consumption AADL Modeling
Core Power Analysis Properties

--- Power Properties
Power_Consumption_Range: CAT_Properties::Power_Consumption_Range_Type applies to (processor, memory, bus, device, process, thread, subprogram
Power_Consumption_Range_Type: type range of CAT_Properties::Power_Consumption_Type;
Power_Consumption_Type: type aadrreal units CAT_Properties::Power_Units;
Power_Units: type units (pW, nW => pW * 10^6, uW => nW * 10^6, mW => uW * 10^6, W => mW * 10^6, KW => W * 10^3);

--- Time Properties
Time_Per.Byte.Range: CAT_Properties::Time.Per.Byte.Range_Type applies to (memory, data, device);
Time_Per.Byte.Range_Type: type range of CAT_Properties::Time.Per.Byte_Type;
Time_Per.Byte_Type: type aadrreal units CAT_Properties::Time.Per.Byte_Units;

--- Energy Properties
Energy_Consumption.Range: CAT_Properties::Energy.Consumption.Range_Type applies to (processor, memory, bus, device, process, thread, subprogram
Energy_Consumption.Range_Type: type range of CAT_Properties::Energy_Consumption_Type;
Energy_Consumption_Type: type aadrreal units CAT_Properties::Energy_Units;
Energy_Units: type units (pJ, nJ => pJ * 10^9, uJ => nJ * 10^9, mJ => uJ * 10^9, J => mJ * 10^9, KJ => J * 10^3);
Energy_Consumption.Per.Byte_Type: type aadrreal units CAT_Properties::Energy_Per.Byte_Units;
Power Consumption AADL Modeling
Core Power Analysis Properties

- Core Hardware Properties
  - Frequency (more handy than SEI cycle time).
  - Hardware technology (e.g.: ASIC 130 nm, ASIC 90 nm)
  - Physical property types (length, etc…).

```
-- Frequency Properties
Frequency: inherit CAT_Properties::Frequency_Type applies to (processor, bus, system, memory, device);
Frequency_Type: type aadreal units CAT_Properties::Frequency_Units;
Frequency_Units: type units (GHz, MHz => GHz * 1000, KHz => MHz * 1000, Hz => KHz * 1000);

-- Technology Properties
Hardware_Technologies: type enumeration (FPGA_VP30_FF896_6C, FPGA_VP30_FF896_7C, ASIC_130NM, ASIC_90NM, ASIC_65NM);

-- Hardware Platform Properties
Hardware_Technology: inherit CAT_Properties::Hardware_Technologies applies to (system, processor, device, memory, bus);

-- Length Properties
Length_Type: type aadreal units CAT_Properties::Length_Units;
Length_Units: type units (pm, nm => pm * 1000, um => nm * 1000, mm => um * 1000, m => mm * 1000, km => m * 1000);
```

Dominique BLOUIN, Skander TURKI, Eric SENN, Saâdia DHOUIB

Lab-STICC

Brest, 11/06/2009
Two phases to analyze the power consumption:

- Phase 1: power estimation: The power consumption is computed for every software component. Apply the FLPA methodology.

- Phase 2: Consumption analysis: The power consumption of SW components are combined to get the power consumption for the HW component
  - At deployment, SW components model are bounded to HW components in the “AADL target platform” model
  - AADL instance model ➔ use the binding properties of software components onto hardware components:
    - Threads onto processors
    - Processes, threads and data onto memories
    - Connections onto bus

- Energy analysis using timing information performed afterwards
Component AADL model libraries consist in property sets and classifiers.

Dedicated Property Sets for components
- Used to store the power estimation model parameters in the AADL model (cache miss rate, memory modes, etc.).

Classifier families for components
- Classifier and package names hierarchy (AADL extension mechanism)
  - E.g.: processor ∈ processor family ∈ builder ∈ processor model (PowerPC 405)
  - Classifiers are used to mark the components in the AADL spec so that they can be identified by power estimation tools.
- The classifier and properties are linked. Property applies to classifiers.
- AADL Modelling of Specialized Processors
  - AADL V2: Properties applicable to classifiers.
The processor model (power model) represents the way the processor's consumption varies with its activity.
The algorithm model links the algorithm with the activity it induces in the processor.
From C source: the compiler model (prediction model) represents the compiler behaviour.
Processor and associated bus declarations

```plaintext
processor ProcessorType
  extends cat::processors::ibm::ProcessorPOWERPC405_Type
  features
    Data_jtag: in out data port;
  properties
    CAT_Properties::Frequency => 300.0 MHz;
    CAT_Processor IBM Properties::Data_Memory_Config => PLB_SDRAM_CACHE;
    CAT_Processor IBM Properties::Inst_Memory_Config => PLB_SDRAM_CACHE;
end ProcessorType;

thread implementation control_thread.speed
  properties
    Source_Text => "E:/Projets/spices/osate/pet/benchmarks/mpeg.c";
    CAT_Thread_Properties::Data_Cache_Miss_Rate => 25.0;
    CAT_Thread_Properties::Inst_Cache_Miss_Rate => 50.0;
end control_thread.speed;

bus implementation Bus_OnChip.Bus_PLB
  properties
    CAT_Properties::Frequency => 100.0 MHz;
end Bus_OnChip.Bus_PLB;

bus implementation Bus_OnChip.Bus_OPB
end Bus_OnChip.Bus_OPB;
```
Platform technology declaration (inherit property definition)

```
BusAccessConnection19: bus access OPB_BUS -> SATA1_CTL.Bus_OPB;
BusAccessConnection20: bus access OPB_BUS -> SATA2_CTL.Bus_OPB;
BusAccessConnection21: bus access OPB_BUS -> Video_Codec_CTL.Bus_OPB;

properties
    CAT_Properties::Hardware_Technology => FPGA_VP30_FF896_7C;
end FPGA.Xilinx_VirtexIIPro_multiproc;
```

Next steps:
- Instantiate the system
- Bind the threads
- Compute power estimation
AADL modeling of synchronization, local and remote Inter-process communications

- Provide AADL packages that describe
  - Synchronization: mutex, semaphores, signals
  - Communication: pipes, shared memory, message queues, sockets

- Each package contains a specific data type component corresponding to a communication mechanism

- Services of an IPC are modeled as server subprogram features

- Only socket services are modeled as features of the socket data component

- Definition of a POSIX properties set
Modeling Examples
IPC Sockets

Ethernet controllers energy model:
\[ E = F(F_{\text{cpu}}, F_{\text{bus}}, \text{Protocol}, \text{Data}) \]
Modeling Examples

IPC Sockets: SW Components

- **IPC AADL extension**
  - Socket data classifier
  - Features socket communication subprograms

- **Known by the power analysis tool.**

- **Send subprogram**
- **Requires access to:**
  - Socket data
  - Message data
  - Message length data

---

Dominique BLOUIN, Skander TURKI, Eric SENN, Saâdia DHOUIB

Lab-STICC

Brest, 11/06/2009
- Application model
  - Extends the library classifiers
  - Refines the features
  - Send / Receive subprograms need to be defined in application model because they access the actual message being send.
- RPC Socket thread
  - Access application socket and message data
  - Defines the calls to socket subprograms

```plaintext
thread Conversion_rpc_socket
  features
  sock_access: requires data access MJpeg_Data::socket.tcp_posix;
  addr_access: requires data access MJpeg_Data::IP_address.impl;
  block_access: requires data access MJpeg_Data::Block.RGB;
  input_RGB_block: in event data port MJpeg_Data::socket;
  output_YUV_block: out data port MJpeg_Data::Block.YUV;
  properties
    Dispatch_Protocol => Periodic;
  end Conversion_rpc_socket;

thread implementation Conversion_rpc_socket.impl
  calls
  {
    so_connect: subprogram MJpeg_Data::connect;
    so_recv: subprogram MJpeg_Data::recv;
    so_close: subprogram MJpeg_Data::close;
  };
  connections
  connection1: data access sock_access -> so_connect.sock;
  connection2: data access addr_access -> so_connect.serv_addr;
  connection3: data access sock_access -> so_recv.sock;
  connection4: data access block_access -> so_recv.msg;
end Conversion_rpc_socket.impl;
```
RPC thread inside a process.
- Set the source code for image conversion.

```plaintext
process MJpegCompressor_socket
features
  input_from_ImageAcquisition: in event data port MJpeg_Data::socket.tcp_posix;
  output_to_Flash: out data port;
end MJpegCompressor_socket;

process implementation MJpegCompressor_socket.impl
subcomponents
  RGB2YUV: thread Conversion_rpc_socket.impl { 
    Source_Text => "E:/Projets/spices/catDataModel/XUPV2P/ccode/rgb2yuv.c";
  }
  DCT: thread DCT;
  Quantif: thread Quantification;
  socket: data MJpeg_Data::socket.tcp_posix;
  IP_address: data MJpeg_Data::IP_address.impl;
  blockRGB: data MJpeg_Data::Block.RGB;
  blockYUV: data MJpeg_Data::Block.YUV;
connections
  block_in_conversion: event data port input_from_ImageAcquisition -> RGB2YUV.input_RGB_block;
  block_in_DCT: data port RGB2YUV.output_YUV_block -> DCT.input_from_Conversion;
```

Dominique BLOUIN, Skander TURKI, Eric SENN, Saâdia DHOUIB
Lab-STICC
Brest, 11/06/2009
Hardware components:
- MAC controller extends the library internal Ethernet controller device

```device implementation OnChip_OPB_IO_Controller.Ethernet_Controller
   extends cat::peripherals::Peripheral_Ethernet_Internal_Type.Basic_Impl
end OnChip_OPB_IO_Controller.Ethernet_Controller;
```

Hardware components:
- OffChip IO controller extends the library external Ethernet controller device

```device implementation OffChip_IO_Controller_Ethernet_Impl
   extends cat::peripherals::Peripheral_Ethernet_External_Type.Basic_Impl
end OffChip_IO_Controller_Ethernet_Impl;
```
- Power consumption model: \[ P = f(F_{\text{cpu}}, \text{Protocol}, S_{\text{data}}) \]

- Socket data power estimation algo
  - \( S_{\text{data}} \):
    - From the Send subprogram feature, get the send subprogram.
    - From the send subprogram msg data access feature, get the message data.
    - From the message data, get the size property (CAT\_Properties::Data\_Size).
  - \( \text{Protocol} \):
    - Infer from the socket data POSIX\_Socket\_Type property.
  - \( F_{\text{cpu}} \):
    - Find the platform MAC controller device through which the data is send
      - Search all the instances of MAC controller classifiers in the system.
      - Retain the controller that has the socket connected to it.
      - The controller connected sockets are found by filtering all connected components through features instances + buses.
      - From the MAC controller device, find the connected processor through OPB bus.
    - The links between the controller and its processor and socket are set during model transformation. In the power-oriented model, the device has a socket collection and processor properties.
Interconnect Explorer Tool back end has been integrated into CAT

Power estimation model parameters are:

- Data being send (due to cross talk on bus lines)
  - A data profile can be used instead the data is not known.
  - RANDOM (noise): 50% commutation rate assumed for each bit.
  - DATA (e.g. image): High weight bits have low commutation rates, lowest weight bit has 50% commutation
  - ADDRESS: Low weight bit has 100% commutation rate.

- Bus physical length
- Number of bits (bus width)
- Technology (ASIC)
- Number of metal layers (ASIC)
- Bufferization.
Define two bus property sets + classifiers:

- Generic properties
- ASIC specific properties

```property set CAT_Bus_PROPERTIES is

-- Physical Bus Length
Bus_Length: CAT_Properties::Length_Type applies to (bus);

-- Bus Width (i.e. the number of bits)
Bus_Width: Size applies to (bus);

-- Bus Data Source
Bus_Data_Source: aadlstring applies to (bus);

-- Bus Data Profile
Bus_Data_Profile: CAT_Bus_Properties::Bus_Data_Profiles applies to (bus);
Bus_Data_Profiles: type enumeration (RANDOM, GENERIC_DATA, ADDRESS);
end CAT_Bus_PROPERTIES;
```

```property set CAT_Bus_ASCII_Properties is

Number_Metal_Layers: aadlinteger applies to (bus);

-- According to the desired frequency on the bus, it is sometimes necessary to insert
-- some buffers along the bus in order to accelerate the data propagation. This parameter
-- according to the bus frequency required has to be set. If your bus frequency is too high then
Bufferization: aadlbooleann applies to (bus);
end CAT_Bus_ASCII_Properties;
```
Conclusions and Future Work

- AADL is suitable for power estimation modeling
- Extensive use of
  - Component classifiers
  - Property sets
  - Extension mechanism
  - Access features.

Other AADL Power Estimation Models
- Flash Memories
- FPGAs (Hardware Threads)
- Power estimation due to thread scheduling (cheddar)
- System Composition
CAT Demo

- **CAT** : Consumption Analysis Toolbox
  - System level power analysis

  - Extensible. Possibility to add new power estimation models for new components.
    - Actually: Uses Eclipse extension mechanism
    - Future: CAT SDK

- Based on an Ecore power-oriented meta model

- Integrated with AADL (and OSATE) via model transformation with ATL (low coupling)

- Works on AADL instance models (depends on binding properties)

- Windows and Linux compatible.

Dominique BLOUIN, Skander TURKI, Eric SENN, Saâdia DHOUIB  
Lab-STICC  
Brest, 11/06/2009
Use Case
Client-Server Distributed Application

Dominique BLOUIN, Skander TURKI, Eric SENN, Saâdia DHOUIB
Lab-STICC
Brest, 11/06/2009