What Is Your Innovation Platform?

Winning With Innovation Platforms

Building Innovation Platforms for Consumer Electronic Product Development

Feb 28, 2012
IEEE Consumer Electronics Society
Santa Clara Valley Chapter

Gopi Kumar Bulusu
Sankhya Technologies Private Limited
www.sankhya.com
Agenda

- A Story of Long Term Growth
- Markets, Opportunities and The Consumer
  - Imperatives for Success
- Principles of System Level Design
  - System Level Design Works!
- Innovation Platforms for CE Device Development
  - Platform Elements
  - Design Flows
  - Example with SANKHYA Teraptor
  - Innovation Platform Maps
  - Emerge a Winner!
## Story of Long Term Growth

- Companies in the space show consistent growth

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>7.68 UK Pound</td>
<td></td>
<td>100</td>
<td>232.4</td>
<td>406.6</td>
<td></td>
</tr>
<tr>
<td>Broadcom</td>
<td>16.33 US $</td>
<td></td>
<td>1100</td>
<td>2670</td>
<td>6818</td>
<td></td>
</tr>
<tr>
<td>Texas Instruments</td>
<td>34.6 US $</td>
<td></td>
<td>9200</td>
<td>13392</td>
<td>13966</td>
<td></td>
</tr>
<tr>
<td>Samsung</td>
<td>137 US $</td>
<td></td>
<td>8940</td>
<td>17210</td>
<td>27834</td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>365.83 US $</td>
<td></td>
<td>7980</td>
<td>13930</td>
<td>65220</td>
<td>108250</td>
</tr>
</tbody>
</table>

Disclaimer: Based on Information from Internet, For Illustration Purpose Only
A CE Device - CE 2012

Embedded System;
A computer embedded into a larger system or device

Generally Consumer Electronic devices are examples of embedded systems

SoC (System on Chip)
All components of an Embedded System designed on a Single Chip
Winning with Innovation Platforms

Markets and Opportunities

Consumer Needs Drive The Market

- Consumer and Control Centric
  - Convergence
    • Multi-Function Devices
  - Divergence
    • Multiple Devices
- Information and Action Centric
  - Divergence
    • Local – NFC, Blue Tooth
    • Global
- Compute Centric
  - Convergence
    • Cloud

Convergence, Divergence and Emergence!

Consumers Learn and Want To Use Their Knowledge Even More.
Succeeding In The Market Place

- Innovate: Deliver products that are
  - intuitive to use
  - have the right mix (convergence) of functions and performance empowering the consumer
  - at the right price point, globally
  - at the right time, ahead of competition

- Segment: Meet the needs of different consumer segments
  - Deliver multiple products and product variants (divergence)
  - Meet the needs of the global consumer

- Emerge(nt): Deliver products that are in-sync with consumer needs
  - Build core competencies and reusable IP
  - Create Innovation Platforms
    - Enable market driven specification, design, integration, verification and synthesis
    - Concurrent agile development
Model Driven, System Level

- Model Driven
  - Use abstraction. (Ex: Processor Model, System Model)

- Meta-Model Driven
  - Multi-level abstraction – Model for a Model

- Transparent Models
  - Models conforming to a meta-model

- Dimensions of Design and Development
  - Component Types, Simulation, Realization, Programming Tools (SDK), Intent Driven Verification

- Degrees of Freedom
  - Ability to modify the design in a particular dimension
    - Example: Processor core for specific application

- System Level – Architectural and Hierarchical System Models

- Multi-Dimensional Design Automation – Using a single model to automate multiple design and development activities
A System Model - Example

Component Types

Memory Device Model

CPU Model

USART Model

Components

Memory

Processors

USART

Relationship

Component Types

Read/Write

Set/Get

Copyright © Sankhya Technologies Private Limited, Company Confidential
## Principles of System Level Design / Benefits

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Benefit To Designer</th>
<th>Translating To Better Design</th>
<th>Competitive Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Level Design</td>
<td>Use Architectural Modeling Languages for Faster Development</td>
<td>Globally Optimal Designs Leading To High Performance, Low Power</td>
<td>Lower Design and Manufacturing Costs</td>
</tr>
<tr>
<td>Transparent Hierarchical Models</td>
<td>Multi-Dimensional Design Automation</td>
<td>A Comprehensive Solution</td>
<td>Higher Quality Low Support Costs</td>
</tr>
<tr>
<td>Architectural Models</td>
<td>Greater Large Block Design Reuse Across Projects</td>
<td>Improved Quality</td>
<td>Reduced Time To Market Segment with Product Variants</td>
</tr>
</tbody>
</table>

*System Level Design Enables Innovation, Segmentation and Emergence*
Innovation Platforms for CE Design and Development
Platform Elements (partial list)

- **Languages and Models**
  - Architectural System Modeling, Timing Models
  - Processor Architecture (ISA) Modeling, Timing Models
  - Hardware Behavioral Modeling
  - Behavior Modeling (Software, Hardware Emulation)
  - HMI / User Interface Modeling
  - IDE

- **Dimensions of Modeling, Design and Automation (Tools)**
  - Simulation: Component, System, Processor; Functional, Timing, Performance
  - Verification Synthesis: Processor, System
  - Debugger
  - SDK Synthesis / Meta-SDK: Assembler, Linker, Code Generator
  - HLS / VHLS: Architectural to Synthesizable RTL
  - Communication and Bus Synthesis
  - Partitioning and Scheduling
  - C/C++ Compiler Synthesis / Meta-Compilers
  - Driver (HAL) and PAL Synthesis

- **Degrees of Design Freedom (Transparent Models)**
  - System Model, Processor Models, Components, System Software, Application Software, HMI Models

- **Library of Components**
  - Processor Models, Memory and Cache Models, Serial and Parallel Devices
Platform => Agile Concurrent Development

Architectural Models

System Architect
Software Engineer
Hardware Engineer
Verification Engineer
Concurrent Development Activities

- Solution Specialist: Market Research and Product Specification
- System Architect: Product Prototyping, UI and UE and Analysis
- Processor and Tools Architect: Processor Modeling and Software Development Tools Like Compilers, Assemblers, Linkers, Debuggers, Simulators
- Peripheral Designer: Peripheral Modeling USB, Ethernet, Bluetooth, CAN, RFID, GPS, GSM, Sensors – Behavior, Hardware
- System Software Architect: OS and Driver Development / Porting; Linux, Android Etc.
- Media Architect: DSP, Audio/Video Codecs
- Hardware/Prototype Designer: FPGAs/ASICs
- Application Architect: UML, Developing UI, MVC, Web
- Circuits and Circuit Board Design: PCB Design, DFM
- Physical Design: “Crash Testing” ...
Idea

System Level Design Flows

STB, DVD Player, Camera, MP3

System Models

Processor, Perihperal Models

Technology

Product Abstraction

Embedded System Level Design

Innovation Platforms for CE Design

Winning with Innovation Platforms

Design Abstraction

Market

Solution
Winning with Innovation Platforms

Application Driven Embedded System Level Design
For Consumer Electronic Product Development
Specify => Model => Integrate
Virtualize => Verify => Synthesize

Faster to market translates to higher price points and greater market share.

Off-The-Shelf IP
Hardware and Software Components

Reusable HW/SW Developed In Parallel
Using Virtual Prototypes
Reduced TIME TO MARKET

Virtual Prototypes
With Teraptor Player

Model Driven Tools For Managing Design Tradeoffs

Idea

Market Driven Application Specification

Automatic component reuse spanning application, system, peripherals and processors

System Models (SSDL)
Teraptor Designer

Processor Models (SMDL)
Teraptor MSE, SDK, Debugger
Model Export, Synthesis

Transparent Modeling Languages for Processor and System Design

Off-The-Shelf IP
Hardware and Software Components

Very High Level Synthesis
Custom Synthesis Tools

Transparent Modeling Languages for Processor and System Design

3-9 Months!

Teraptor™
Eclipse Based IDE
MSE and Verifier
SMET for LLVM
Very High Level Synthesis Debugger
SDK
Player
Channels
C Compiler SA*
C Compiler HPC*

Off-The-Shelf IP
Hardware and Software Components

Virtual Prototypes With Teraptor Player

Model Driven Tools For Managing Design Tradeoffs

3-9 Months!
## Innovation Platform Map

### 1. Languages

<table>
<thead>
<tr>
<th>SNO</th>
<th>Layer</th>
<th>Model-Type</th>
<th>Modeling Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System</td>
<td>Architectural ER Model</td>
<td>SSDL (or SysML)</td>
</tr>
<tr>
<td>2</td>
<td>Component</td>
<td>Processor ISA</td>
<td>SMDL</td>
</tr>
<tr>
<td>3</td>
<td>Component</td>
<td>Hardware Emulation</td>
<td>C++</td>
</tr>
</tbody>
</table>

### 2. Dimensions of Automation and Tools

... 

### 3. Degrees of Design Freedom (Transparent Hierarchical Models)

... 

### 4. Library of Components

...
Emerge a Winner!

Innovation Platforms for CE Design
Help Your Organization To Win In The Market Place By Practicing System Level Design and Thereby:
+ Supporting Concurrent Agile Development
+ Enabling Globally Optimal Designs
+ Allowing Greater Design Reuse Across Product Lines
+ Ensuring Multi-Dimensional Design Automation

Processors

Peripherals

Software

Product Line 1

Product Line 2

Product Line 3
Questions?

➢ Innovation is the specific instrument of entrepreneurship... the act that endows resources with a new capacity to create wealth. – Peter Drucker

E-mail gopi@sankhya.com
http://www.sankhya.com/contact.html
http://www.sankhya.com/info/products/teraptor/teraptor.html
System Model – A Quick Example
model processor DLX dlx.md
model device SRAM SRAM.dll
model device USART USART.dll

component DLX cpu1
component DLX cpu2
component SRAM ram size=0x2000 mode=rw
component USART usart

map memory cpu1:address.0x0 ram:STDRAM
map memory cpu1:address.0xffff0000 usart:tx_register
• Specify several views of the processor like, Architectural view, Mnemonic View and Machine code

Instruction Set architecture
Describes the Instruction Set Architecture of the target processor in, Behavioural, Mnemonic and machine code formats.

Mnemonics

Assembler Interface
Describes the mapping between external assembler format and the internal format used in the processor model.

Binary Interface
Describes the Application Binary Interface. This includes information like stack layout and parameter passing convention.

Machine code

Relocation Interface
Describes the relocations supported by the target architecture.

Behavior Code
SMDL Example – Operands

• Immediate Operand
  // 5-bit unsigned immediate value
  Operand o_uimm5 : Integer {
    range (i = 0, 31, 1) {
      cbcode = { "$i" }; 
      asm = { "$i" }; 
      mcode = { "$i" }; 
    };
  };

• Register Operand
  // 32 general purpose registers + 3 special purpose registers
  Operand o_gprv : Register {
    range (i = 0, 34, 1) {
      cbcode = { "r$i" }; 
      asm = {"r$i"}; 
      mcode = {"$i"}; 
    };
  };}
SMDL Example – Addressing Modes

- E.g., PC Relative Offset

Operand o_i_pcrel {
    o_imm16 o;
    cbcode = { "+", "pc", "<<", o, "2" };  
    asm = { expr("<< $o 2") };  
    mcode = { o(0,16) };  
};
SMDL Example - Instruction

- Here's an Example of SMDL Representation of a simple 'add' instruction of the **DLX processor**

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>add rd rs1 rs2</td>
<td>000001</td>
<td>rs1 rs2 rd 00000 00000</td>
</tr>
</tbody>
</table>

**Instruction i_rrr_add**

```c
o_gprv rd, rs, rt;
```

```c
cbcode = { "{width=32}=" , rd , "{sign=u}+" , rs , rt };  // Common Behavior Code
```

```c
asm = { "add" , rd , rs , rt };                      // Mnemonic
```

```c
mcode = { or(0,7,0b000000000); or(7,4,0001); rd(11,5); rt(16,5);
          rs(21,5);or(26,6,0b000001) };               // Machine Code
```

```c
};
```
SMDL Example – Instruction Bundle

• Allows VLIW type instructions to be modeled.
• Example: A 128-bit VLIW

Bundle b_mii {
  i_mem i1;
  i_alu i2, i3;

  b_width = 128;
  cbcode = { i1, i2, i3 };
  asm   = { i1, i2, i3 };
  mcode = { i1(0,48); i2(48,40); i3(88,40) };
};
SMDL Example – Instruction Attributes

• Specify width as 32 bits for this instruction.
cbcode = { "width=32" , rd, "\{sign=u\}+", rs, rt };

• Cycle Attribute - specifies number of clock-cycles to be used.
{cycle=2} ++ {read-cycle=1,write-cycle=3} r1

• Effects Attribute – Specifies side-effects of a cbcode operation
E.g. Subtract with Carry (sbc) instruction for two 32-bit values

// SBC{cond}{S} Rd, Rn, <Op2>
Instruction i_rro_sbc {
  o_gprv rn, rd;
  o_operand2 op2;
  cbcode = { "=", rd, "-", "\{!effects=1,
     !effects-create-mask=NZCVttttttttttttttttttttttttttttt,
     !effects-destination=flags\}-", "-", rn, op2, "bf", "29", "1",
         "flags"
   };
  asm   = { "sbc", rd, rn, op2 };
  mcode = { or(26, 2, 0b00); or(21, 4, 0b0110);rn(16, 4); rd(12, 4); op2(0, 12) };
};

N, Z, C, V, t – Negative, Zero, Carry, Overflow, Transparent (not affected)
flags – Status register
SMDL – Key Features

− The SMDL model contains sufficient information to automate various tasks like CPU design, verification, system and tool development, code generation etc.

− Extensible Language:
  Rich Set of Language attributes enables to Model
  • Cycle-Accurate Processors
  • Complex Pipelining Architectures