Acknowledgement

- TIMA (Techniques of Informatics and Microelectronics for Computers Architecture) Labs, INPG, CNRS
- Dr. Ahmed Jerraya, Head of Design Group, CEA-LETI, France
- Prof. Frederic Petrot, TIMA Labs, France
- Prof. Frederic Rousseau, TIMA Labs, France
Summary

- **MPSoC (Multi-Processor System On Chip)** integrates different components (hardware and software) on a single chip

**Context:**
- Heterogeneous MPSoC are required by current multimedia applications
  - E.g. TI OMAP, ST Nomadik, Philips Nexperia, Atmel Diopsis
  - DSP + µC + Sophisticated Communication Infrastructure
- Multiple Software (SW) Stacks

**Problem:**
- Classic programming environments do not fit:
  - High level programming environments are not efficient to handle specific architecture capabilities (e.g. C/C++, Simulink)
  - HW (Virtual) Prototypes are too detailed and time consuming for SW debug

**Challenge:**
- Efficient and Fast Programming Environment for Heterogeneous MPSoC

**Proposal:**
- SW development and validation environment using Simulink & SystemC
- Communication mapping exploration

Outline

- **Introduction**

- **Software Design and Validation**
  - System Architecture
  - Virtual Architecture
  - Transaction Accurate Architecture

- **Conclusions**
**MPSoC Architecture**

- **Heterogeneous MPSoC:**
  - SW subsystems for flexibility
  - HW subsystems for performance
  - Complex communication network
    - Bus based architectures
    - Network on Chip (NoC) architectures

- **SW Subsystem:**
  - Specific CPU Subsystem:
    - CPUs: GPP, DSP, ASIP
    - I/O + memory architecture + other peripherals
  - Layered SW Architecture:
    - Application code (tasks)
    - Hardware dependent Software (HdS)
      - Specific to Architecture/Application to achieve efficiency

- **Programming Heterogeneous MPSoC:**
  - Generate SW efficiently by using HW resources for Communication & Synchro.

---

**Example of Heterogeneous MPSoC: Reduced Atmel Diopsis RDT**

- ARM9 SS
- DSP SS
- MEM SS
- POT SS (Periph. On Tile)
  - I/O peripherals
  - System Peripherals
- Interconnect: AMBA bus

- Local & global memories accessible by both processing units
  - Different communication schemes between CPUs
- Require Multiple Software Stacks (ARM + DSP)
Software Stack Organization

- **SW Stack is organized into layers:**
  - **Application code:**
    - SW code of tasks mapped on the CPU
  - **HdS (Hardware dependent Software)** made of different components:
    - OS (Operating System)
    - Comm (Communication Primitives)
    - HAL (Hardware Abstraction Layer)
    - API (Application Programming Interface)
  - Different SW components need to be validated incrementally
    - Different abstraction levels corresponding to the different SW components
    - SW development platforms (HW abstraction models) to allow specific SW components debug and communication refinement

Software Development Platform

- **User SW Code**
  - C/C++, Simulink functions, binary,…
- **Development platform to abstract architectures**
  - Runtime library, simulator (ISS, Simulink)
- **Executable model generation**
  - Compile, Link
- **Debug**
  - Iterative process
  - Different SW components need different detail levels

- **Requirements for MPSoC executable models:**
  - Speed
    - Easily experiment several mapping schemes
    - Multiple SW stacks
  - Accuracy
    - Evaluate the effect on performance by using specific HW resources
    - Debug low level SW code
Why Simulink?

- Adapted environment for Complex Algorithm modeling
- Rich library of predefined functional blocks
- Offers a set of algorithms blocks for a variety of applications
  - Signal Processing Blockset: FFT, DCT, IDCT, IFFT, …
  - Video Processing Blockset: SAD, Edge Detection, PSNR, Block matching
- User defined blocks integration (S-Functions)
- Provides simulation, profiling and code generation facilities
  - Real Time Workshop (RTW) for C code generation
  - HDL Coder for VHDL generation

➢ Open issue: Algorithm mapping and refining for MPSoC
### Why SystemC?

- **Standard System Level Design Language**
  - Unified language for HW & SW development based on C++ extension
- **Concurrency support:** hardware modules
- **Concept of time** (clocks, delays with custom `wait()` calls)
- **Communication model:** signals, protocols
- **Reactivity to events:** support of events, sensitivity list
- **Integrated Simulation Core for the Realization of Executable models**
  - Modeling and Simulation within a wide range of abstraction levels

- **Still Low Level Design Language**
  - Not easy to specify complex systems at algorithm level

### Software Abstraction Levels

#### SW Development

- **Platform**
- **Sw-SS1**
- **Sw-SS2**

#### SW Architecture

- **System Architecture**
- **Virtual Architecture**
- **Transaction Accurate Architecture**
- **Virtual Prototype**

Models to debug the SW

SW Design
Outline

- Introduction

- Software Design and Validation
  - System Architecture
  - Virtual Architecture
  - Transaction Accurate Architecture

- Conclusions
System Architecture Design

- Captures application and mapping to architecture

- Task
  - Set of functions: Simulink blocks, S-functions

- Subsystem
  - Set of tasks

- Abstract communication types
  - Communication Intra-SS
  - Communication Inter-SS

- Communication protocol
  - Generic Simulink I/Os
  - Explicit annotation for implementation

- Algorithm validation through simulation

Application Example: M-JPEG Decoder Mapped on Diopis RDT

- Application Specification (~ 1700 LOC)

- Target Architecture: Diopis RDT with AMBA

- Partitioning & Mapping
System Architecture Level Software Development Platform for Diopsis RDT

Communication units: Simulink Signals
- 5 Intra SS communication units
  - depends on application
- 3 Inter SS communication units
  - depends on application
- Generic channels to be mapped on resources
- Execution model in Simulink

Architecture parameters annotating the model:
- **ResourceType**
  - ARM9, DSP, POT, Task
  - Communication: swfifo, dmem, sram, reg, dxm
- **NetworkType**: AMBA_AHB, NoC
- **AccessType**: DMA, direct
- **MemName**
  - Validation of application functionality

MJPEG System Architecture in Simulink

- 7 S-Functions
- Algorithm validation, 10 frames QVGA YUV 444
- Simulation time: 15s on PC 1.73GHz, 1GBytes RAM

Decoded Microblock

Simulation running
Capture of low level architecture features in Simulink for MJPEG

- 8 communication units:
  - 3 Inter-SS + 5 Intra-SS
- Communication schemes:
  - Changing annotation of Simulink model
- Nb AMBA cycles REG to transfer "N" words:
  - ARM wr: \(N/4+8+(N-1)\)
  - DSP rd: \(N/4\)

ResourceType = REG

E.g. REG

Capture of low level architecture features in Simulink for MJPEG

- Nb AMBA cycles DXM to transfer "N" words:
  - ARM wr: \(2^N\)
  - DSP rd: \(14+(N-1)\)

ResourceType = DXM
AccessType = DMA

E.g. DXM
Capture of low level architecture features in Simulink for MJPEG

- Nb AMBA cycles SRAM to transfer “N” words:
  - ARM wr: N/2
  - DSP rd: 5+(N-1)

E.g. SRAM

- Easy to experiment different communication schemes

Virtual Architecture Model

- Abstract CPU-SS1
- Abstract CPU-SS2
- SystemC TLM
- HDS API

Virtual Architecture

- Transaction Accurate
- Architecture
- Virtual Prototype
Virtual Architecture Design

- **Hardware Architecture**
  - SystemC TLM, message accurate model
  - Tasks encapsulated in SC_THREADS
  - Inter-SS communication units partially mapped on the resources
  - Abstract interconnect component
  - Intra-SS communication units become software communication channels

- **Simulation**
  - Task scheduled by the SystemC scheduler
  - Task code validation and partitioning

```cpp
class fifo_ch : public sc_prim_channel {
  word *buffer;
  public:
  word *read (int size) {
    for (i=0; i<size; i++)
      *(ret+i)=*(buffer+i);
    return ret;
  }
  void recv (fifo_ch* ch, void* dst, int size) {
    dst = ch->read (size);
  }
};

SC_MODULE (CPU_SS2) {
  Task_T2  * T2;
  // tasks
  in_port *in;
  // ports
  out_port *out;
  fifo_ch*ch1;  // channels
  ...}

SC_MODULE (Task_T2){
  SC_CTOR (Task_T2){
    SC_THREAD(task_T2, clk);
  }
  void task_T2( )  {
    while(1)
      recv(CH1, B, 10);    // Comm. API
      F1(B,C);                  // Computation
      F2(C,D);
      send(CH2, D, 10);  // Comm. API
  }...
```

**Application Example:**
M-JPEG Decoder mapped on Diopsis RDT

- **Task code of T2**
  ```cpp
  int B[10], C[20], D[10];
  void task_T2( ) {
    while(1) {
      recv(CH1, B, 10);    // Comm. API
      F1(B,C);                  // Computation
      F2(C,D);
      send(CH2, D, 10);  // Comm. API
    }...
  ```

- **Abstract SS**
  - Inter-SS communication partially mapped on explicit resources (dxm, sram, reg, dmem)
  - Intra-SS communication becomes swfifo channels
  - Abstract AMBA bus: virtual arbiter + address decoder

- **Software Architecture**
  - Task C code based on HdS APIs
**Application Example:**
**M-JPEG Decoder mapped on Diopsis RDT**

![Diagram of task code on ARM9](image1)

- **Tasks code on ARM9**
- **Tasks C code based on HdS API**
- **Tasks scheduled by SystemC scheduler**

**Software**

**Abstract ARM9-SS**

![Diagram of task code on DSP](image2)

**Abstract DSP-SS**

**Abstract POT-SS**

**Abstract Bus**

**MEM-SS**

**SRAM**

**DMEM**

**REG**

**T1**

**T2**

**T3**

**T4**

**T1**

**T2**

**T3**

**T4**

**Results for the M-JPEG Decoder mapped on Diopsis RDT at the Virtual Architecture Level**

- 3 inter-SS communication mapping schemes: total messages through AMBA, execution time

<table>
<thead>
<tr>
<th>Comm. Unit</th>
<th>ch1_T2T3 (256 bytes)</th>
<th>ch2_T2T3 (4 bytes)</th>
<th>ch_T3T4 (64 bytes)</th>
<th>ch1_T1T2-ch5_T1T2</th>
<th>Total messages AMBA</th>
<th>Execution Time [ns] (1 clock cycle 20ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MJPEG</td>
<td>DXM</td>
<td>DXM</td>
<td>DXM</td>
<td>SWFIFO</td>
<td>216000</td>
<td>4464060</td>
</tr>
<tr>
<td></td>
<td>DXM</td>
<td>REG</td>
<td>DMEM</td>
<td>SWFIFO</td>
<td>144000</td>
<td>3720060</td>
</tr>
<tr>
<td></td>
<td>SRAM</td>
<td>SRAM</td>
<td>DMEM</td>
<td>SWFIFO</td>
<td>108000</td>
<td>2232020</td>
</tr>
</tbody>
</table>

- Simulation time 14s (DXM+REG+DMEM), 10 frames QVGA YUV 444 format

- **Validation of task code and partitioning**
### Transaction Accurate Architecture Model

- **Hardware Architecture**
  - SystemC TLM model
  - Detailed CPU-SS local architecture
  - Abstract CPU cores
  - Explicit communication protocol
  - Explicit interconnect component (bus, NoC)

- **Software Architecture**
  - Task code + OS + Communication
  - Based on HAL APIs

### Transaction Accurate Architecture Design

- **Hardware Architecture**
  - SystemC TLM model
  - Detailed CPU-SS local architecture
  - Abstract CPU cores
  - Explicit communication protocol
  - Explicit interconnect component (bus, NoC)

- **Software Architecture**
  - Task code + OS + Communication
  - Based on HAL APIs

### Code Example

```c
void __schedule(void) {
    int old_tid = cur_tid;
    cur_tid = new_tid;
    __cxt_switch(old_tid.cxt, cur_tid.cxt);
    ...
}
```

```c
main Communication SW
extern void task_T2();
void__start (void) {
    switch (ch.protocol)
    {
    case FIFO:
        if (ch.state==EMPTY) _schedule();
    ...}

void recv(ch, dst, size) {
    ...}
```

```c
C code of Task_T2
void task_T2() {
    while (1) {
        recv(CH1, B, 10);
        ... send(CH2, C, 20);
    }
```
Application Example:
M-JPEG Decoder mapped on Diopsis RDT

- Local SS architectures detailed
- Abstract CPU execution models
- AMBA bus protocol fully modeled
- Inter-SS communication fully mapped on explicit resources
- Intra-SS communication managed by OS

- Simulation time 5m10s (DXM+REG+SRAM), 10 frames QVGA
  - Validation of OS and Comm. integration

Results for the M-JPEG Decoder at the Transaction Accurate Architecture Level

- Different Communication Mapping Schemes
  - SRAM
  - DXM
  - REG

<table>
<thead>
<tr>
<th>Communication Scheme</th>
<th>Transactions to memories [Bytes]</th>
<th>Total AMBA cycles</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5256k</td>
<td>0</td>
<td>8856k</td>
<td>100%</td>
</tr>
<tr>
<td>4608k</td>
<td>72k</td>
<td>7884k</td>
<td>89%</td>
</tr>
<tr>
<td>0</td>
<td>576k</td>
<td>3960k</td>
<td>45%</td>
</tr>
</tbody>
</table>

- Improvement up to 55% in comm. performance by using HW resources
Outline

- Introduction
- Software Design and Validation
  - System Architecture
  - Virtual Architecture
  - Transaction Accurate Architecture
- Conclusions

Conclusion

- Definition of the different abstraction levels and the HW & SW models
  - System Architecture (SA) in Simulink
  - Virtual Architecture (VA) in SystemC
  - Transaction Accurate Architecture (TA) in SystemC

- Structuring the SW stack into layers allows:
  - Flexibility in terms of SW components reuse (OS, Communication)
  - Portability to other platforms (HAL)
  - Incremental generation and validation of the different SW components by using SW development platforms (HW abstraction models)

- HW abstraction models:
  - VA & TA SystemC platforms are automatically generated from Simulink
  - Allow early performance estimation
  - Easily experiment several communication mapping schemes
  - Allow the efficient use of architecture resources

- Programming Environment applied to:
  - Complex heterogeneous MPSoC: RDT with AMBA, R2DT with NoC,1AX (1 ARM, 1 XTENSA, AMBA)
  - Multimedia applications: H.264 Encoder, M-JPEG Decoder, MP3 Decoder, Vocoder
Thank you!

References:


- W. WOLF “High-Performance Embedded Computing”, Morgan Kaufmann, 2006