Software Design and Integration for Embedded Multimedia Applications by Successive Refinement

Katalin Popovici
katalin.popovici@mathworks.com

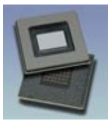
The MathWorks, France

Acknowledgement

- TIMA (***T***echniques of ***I***nformatics and ***M***icroelectronics for Computers ***A***rchitecture) 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** (***M***ulti-***P***rocessor ***S***ystem ***O***n ***C***hip) 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

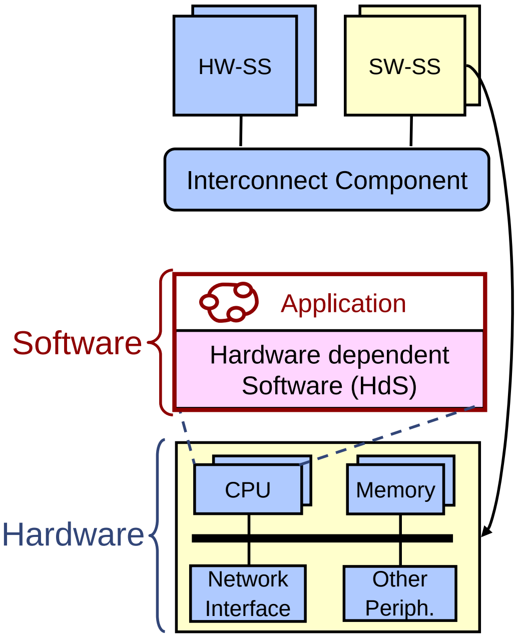
- **Challenge:**
  - Efficient and Fast Programming Environment for Heterogeneous MPSoC
- **Proposal:**
  - SW development and validation environment using Simulink & SystemC
  - Communication mapping exploration

Outline

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

MPSoC Architecture

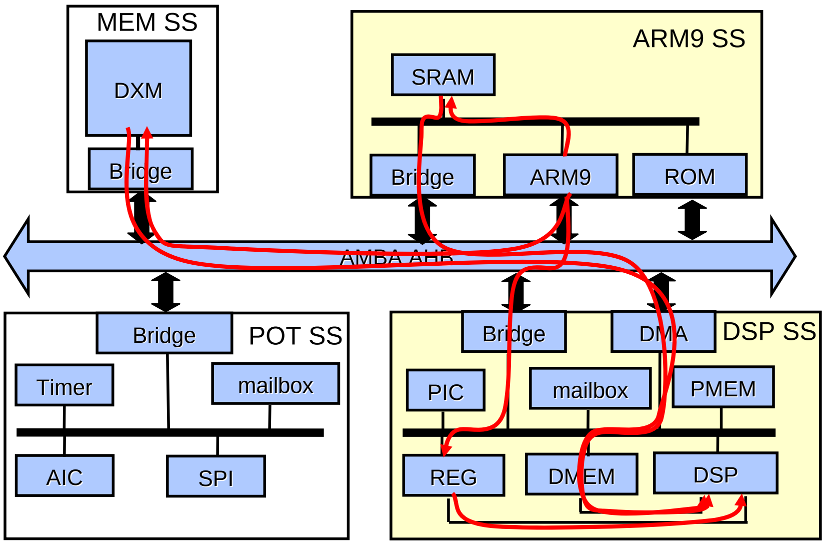
- Heterogeneous MPSoC:
  - SW subsystems for flexibility
  - HW subsystems for performance
  - Complex communication network
    - Bus based architectures
    - Network on Chip (NoC) architectures
- SW Subsystem:
  - Specific CPU Subsystem:
    - CPUs: GPP, DSP, ASIP
    - I/O + memory architecture + 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.

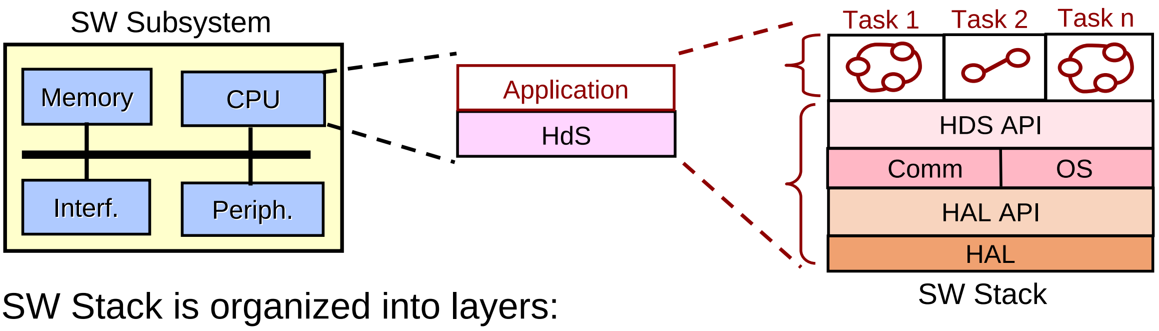
Example of Heterogeneous MPSoC: Reduced Atmel Diopsis RDT

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



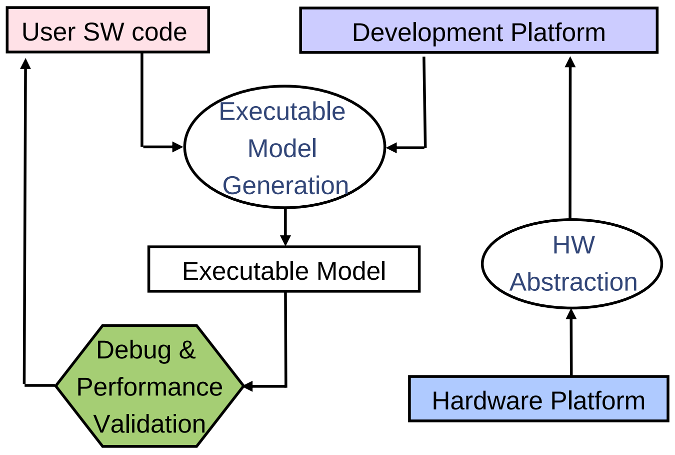
- Local & global memories accessible by both processing units
  - Different communication schemes between CPUs
- Require Multiple Software Stacks (ARM + DSP)

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

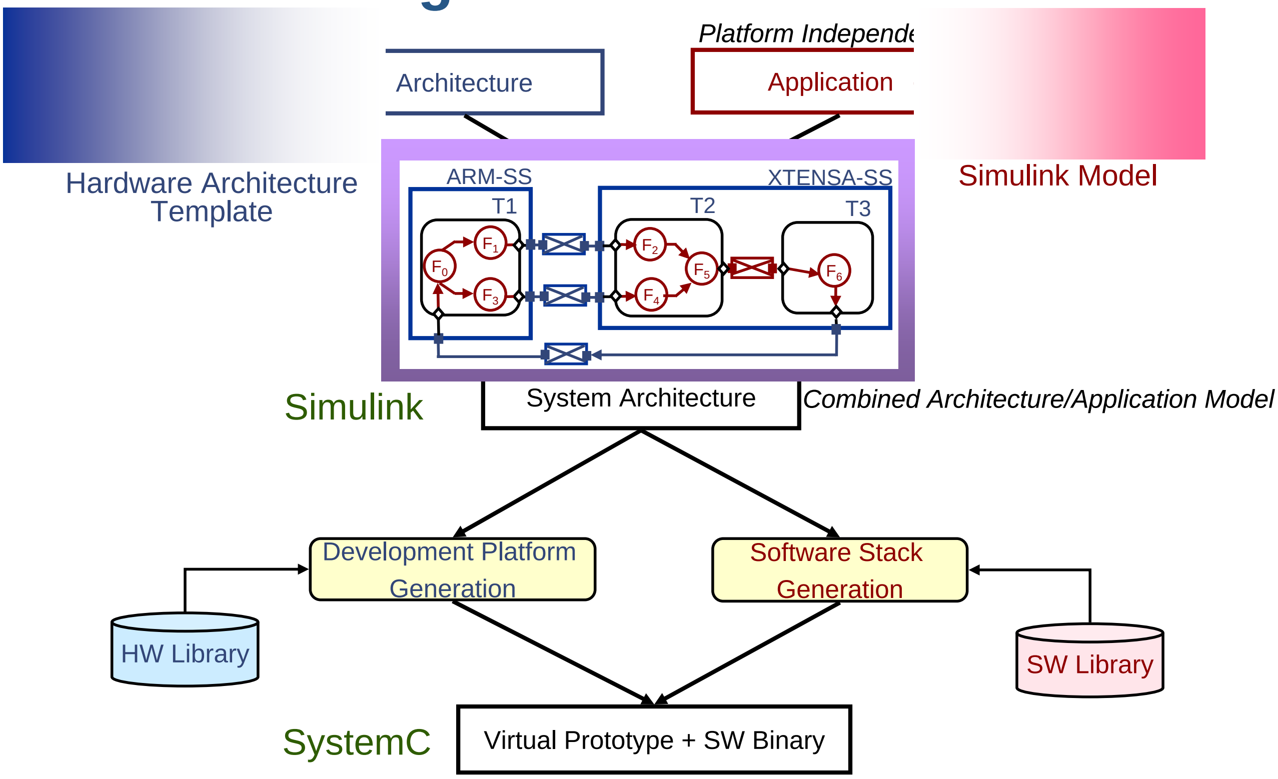
Software Development Platform



- 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

Software Design Flow



Platform Independe

Architecture

Application

Hardware Architecture Template

Simulink Model

Simulink

System Architecture

Combined Architecture/Application Model

Development Platform Generation

Software Stack Generation

HW Library

SW Library

SystemC

Virtual Prototype + SW Binary

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

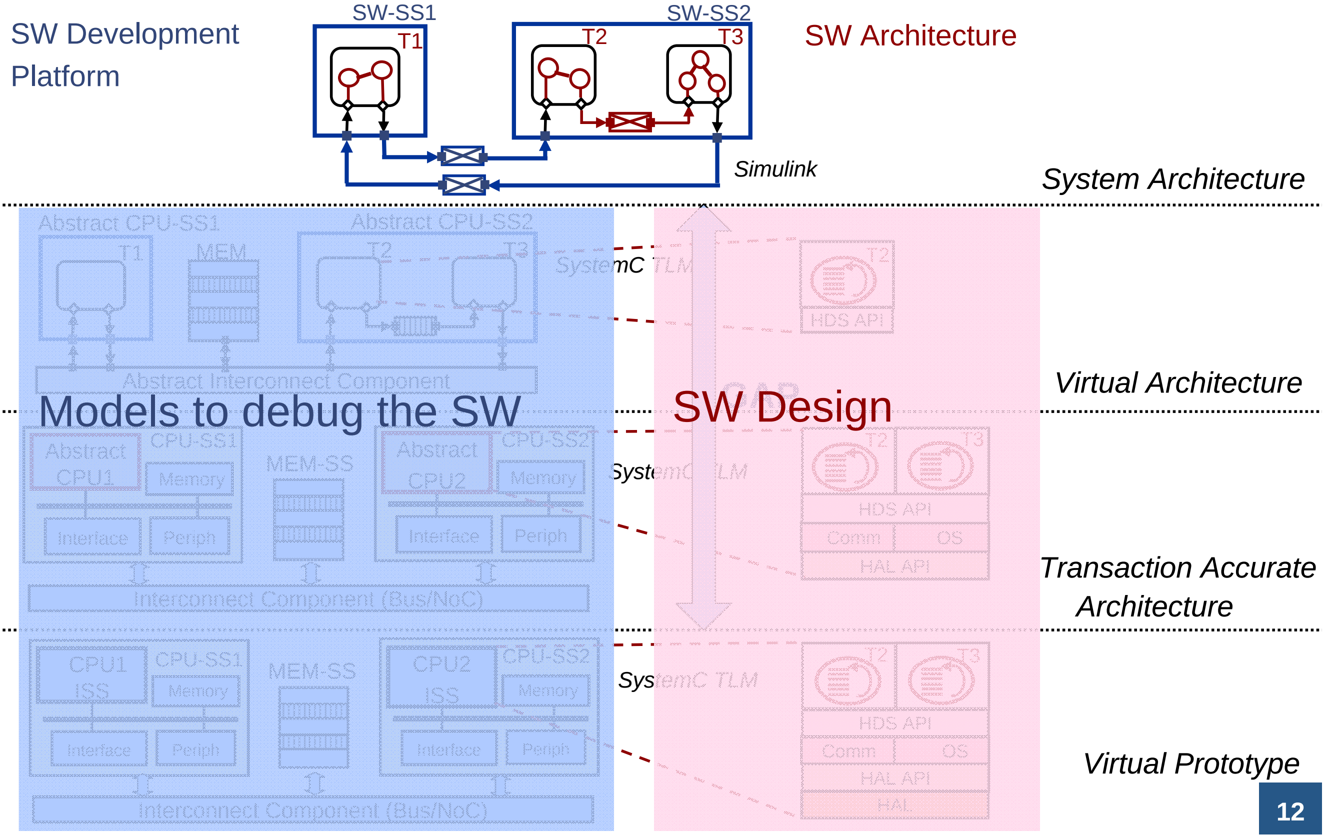
➢ Open issue: Algorithm mapping and refining for MPSoC

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

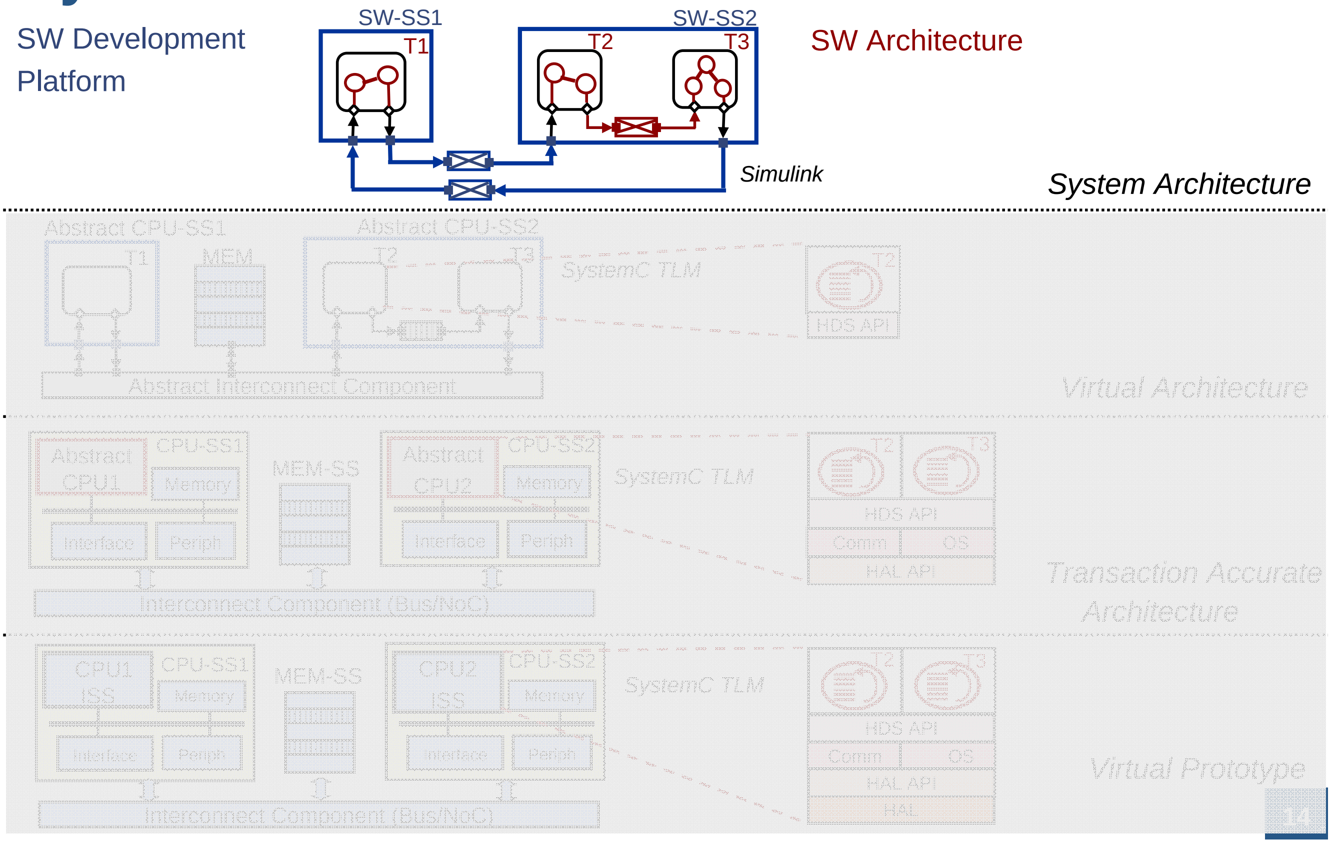


Outline

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

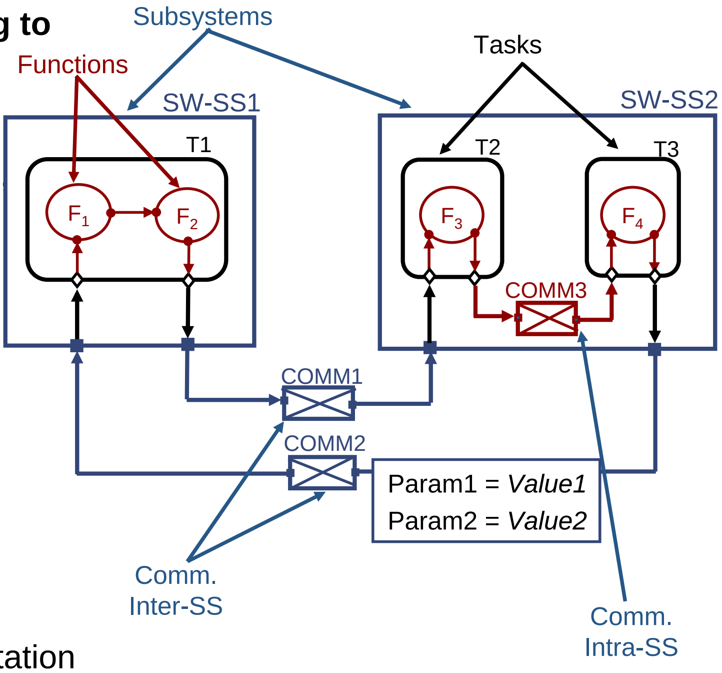
System Architecture Model

SW Development Platform



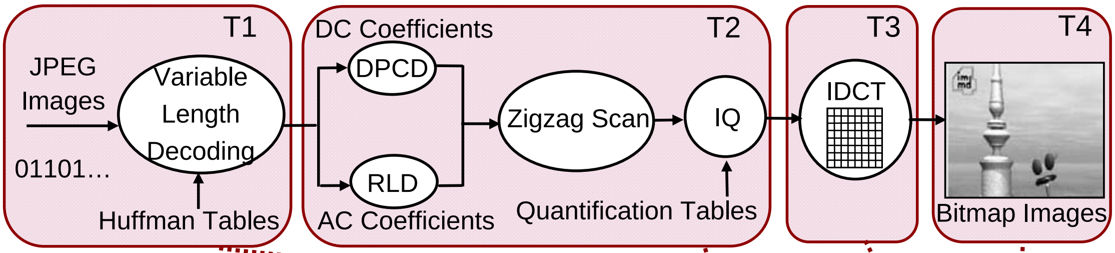
System Architecture Design

- **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**

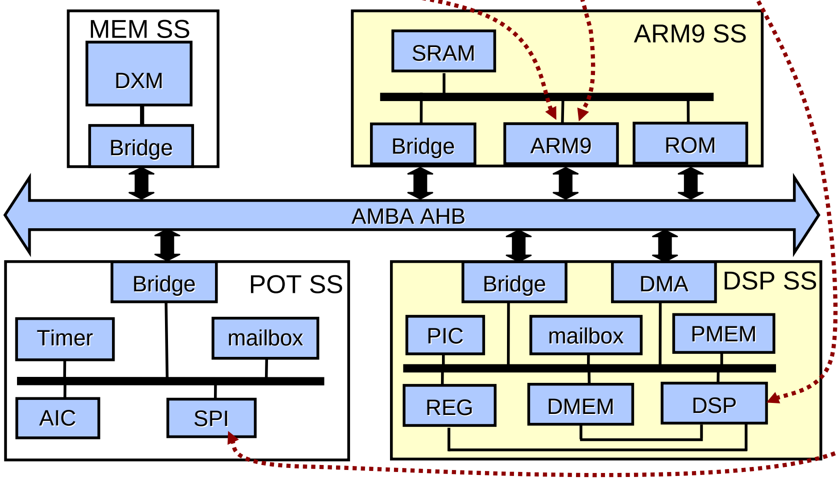


Application Example: M-JPEG Decoder Mapped on Diopsis RDT

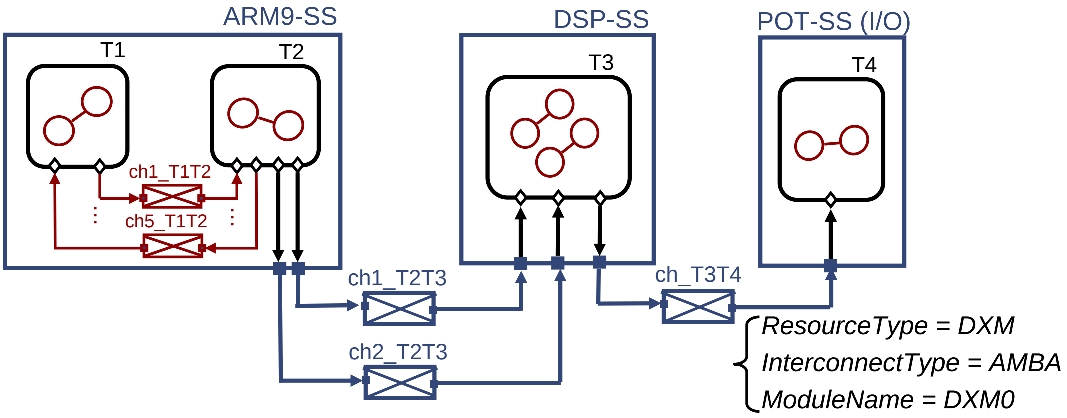
- Application Specification (~ 1700 LOC)



- Target Architecture: Diopis RDT with AMBA
- Partitioning & Mapping



System Architecture Level Software Development Platform for Diopsis RDT



ResourceType = DXM
InterconnectType = AMBA
ModuleName = DXM0

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

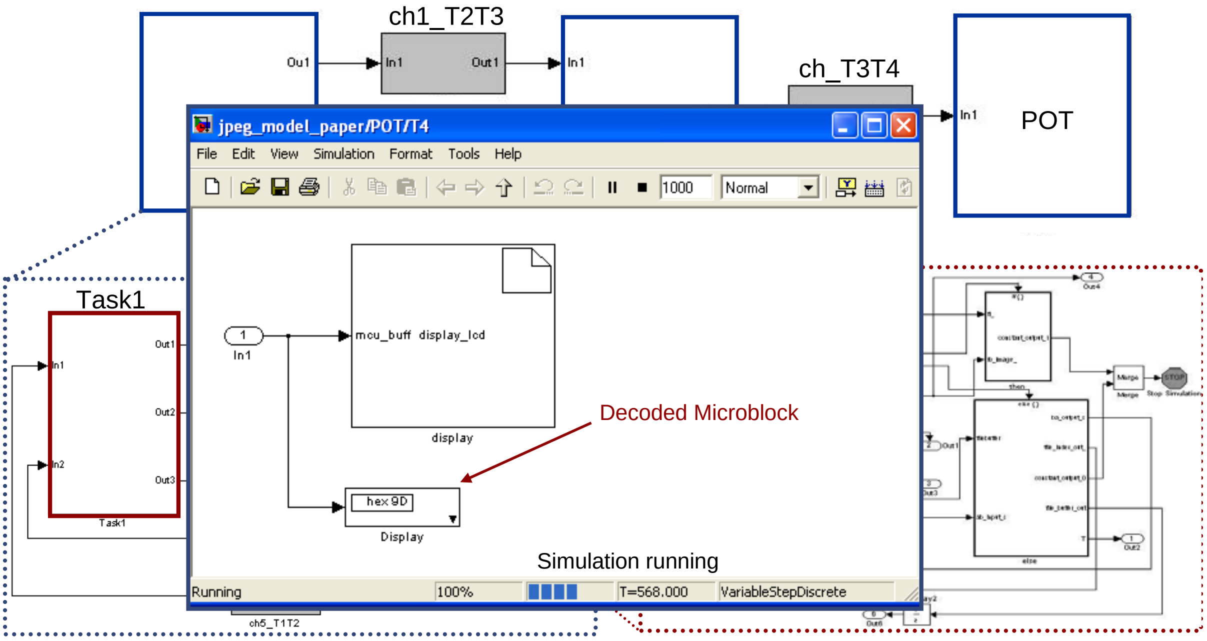
Architecture parameters annotating the model:

- 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

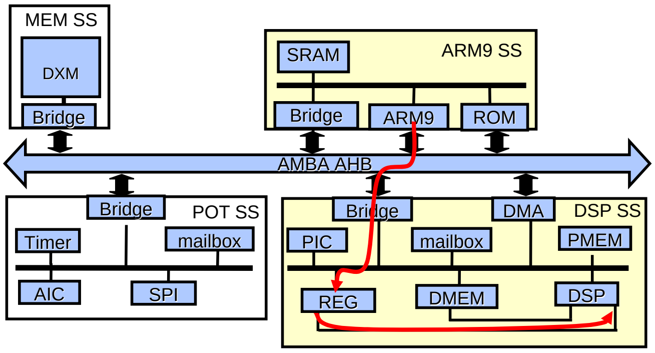
- 7 S-Functions
- Algorithm validation,10 frames QVGA YUV 444
- Simulation time: 15s on PC 1.73GHz, 1GBytes RAM



Decoded Microblock

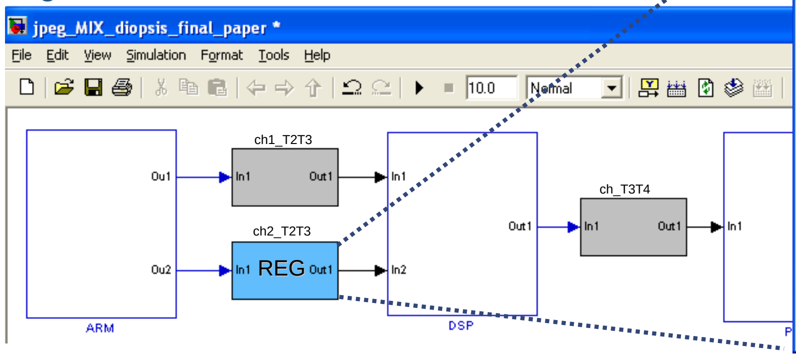
Simulation running

Capture of low level architecture features in Simulink for MJPEG

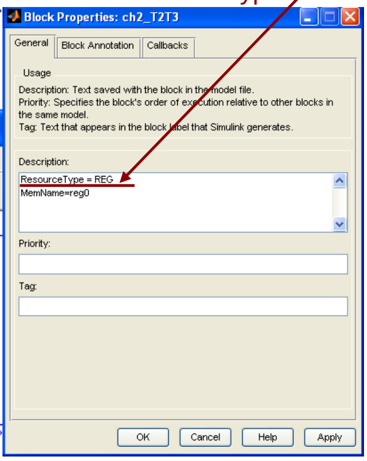


- 8 communication units:
  - 3 Inter-SS + 5 Intra-SS
- Communication schemes:
  - Changing annotation of Simulink model
- Nb AMBA cycles REG to transfer “N” words:
  - ARM wr: N/4+8+(N-1)
  - DSP rd: N/4

E.g. REG



ResourceType = REG


Block Properties: ch2_T2T3

Usage

Description: Text saved with the block in the model file.
Priority: Specifies the block's order of execution relative to other blocks in the same model.
Tag: Text that appears in the block label that Simulink generates.

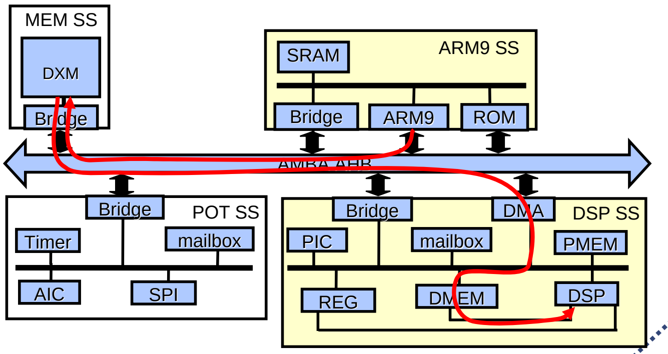
Description:

ResourceType = REG
MemName=reg0

Priority:

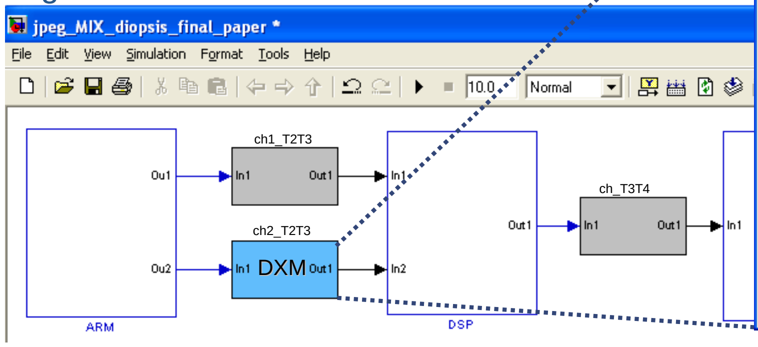
Tag:

Capture of low level architecture features in Simulink for MJPEG

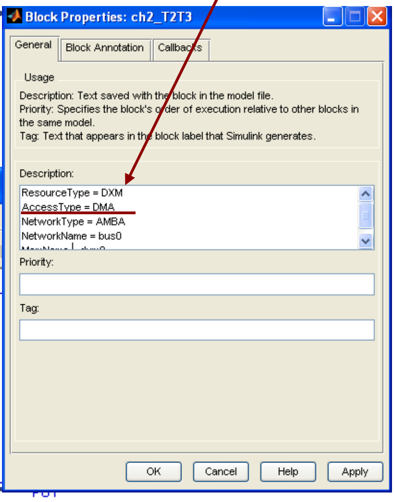


- Nb AMBA cycles DXM to transfer “N” words:
  - ARM wr: 2*N
  - DSP rd: 14+(N-1)

E.g. DXM



ResourceType = DXM
AccessType = DMA


Block Properties: ch2_T2T3

Usage

Description: Text saved with the block in the model file.
Priority: Specifies the block's order of execution relative to other blocks in the same model.
Tag: Text that appears in the block label that Simulink generates.

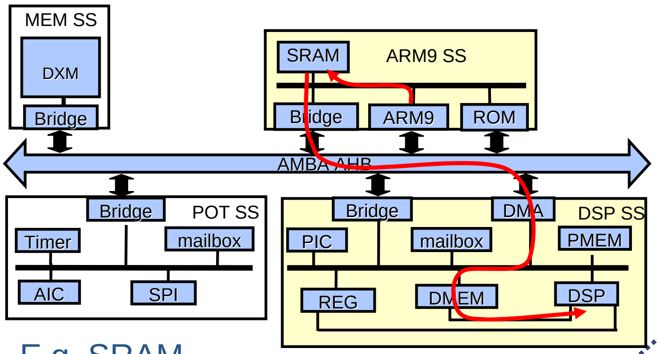
Description:

ResourceType = DXM
AccessType = DMA
NetworkType = AMBA
NetworkName = bus0

Priority:

Tag:

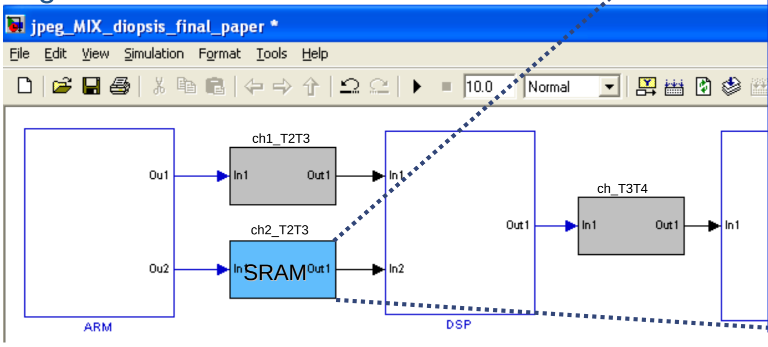
Capture of low level architecture features in Simulink for MJPEG

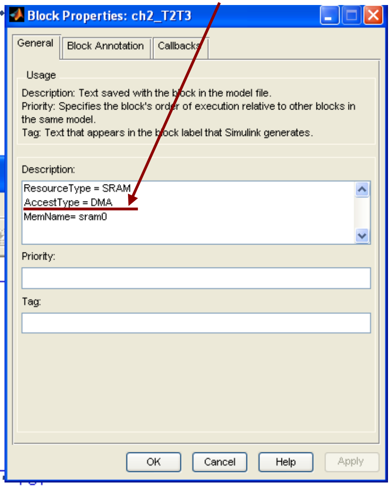


- Nb AMBA cycles SRAM to transfer “N” words:
  - ARM wr: N/2
  - DSP rd: 5+(N-1)

ResourceType = SRAM
AccessType = DMA

E.g. SRAM





Block Properties: ch2_T2T3

General | Block Annotation | Callbacks

Usage

Description: Text saved with the block in the model file.
Priority: Specifies the block's order of execution relative to other blocks in the same model.
Tag: Text that appears in the block label that Simulink generates.

Description:

ResourceType = SRAM
AccestType = DMA
MemName= sram0

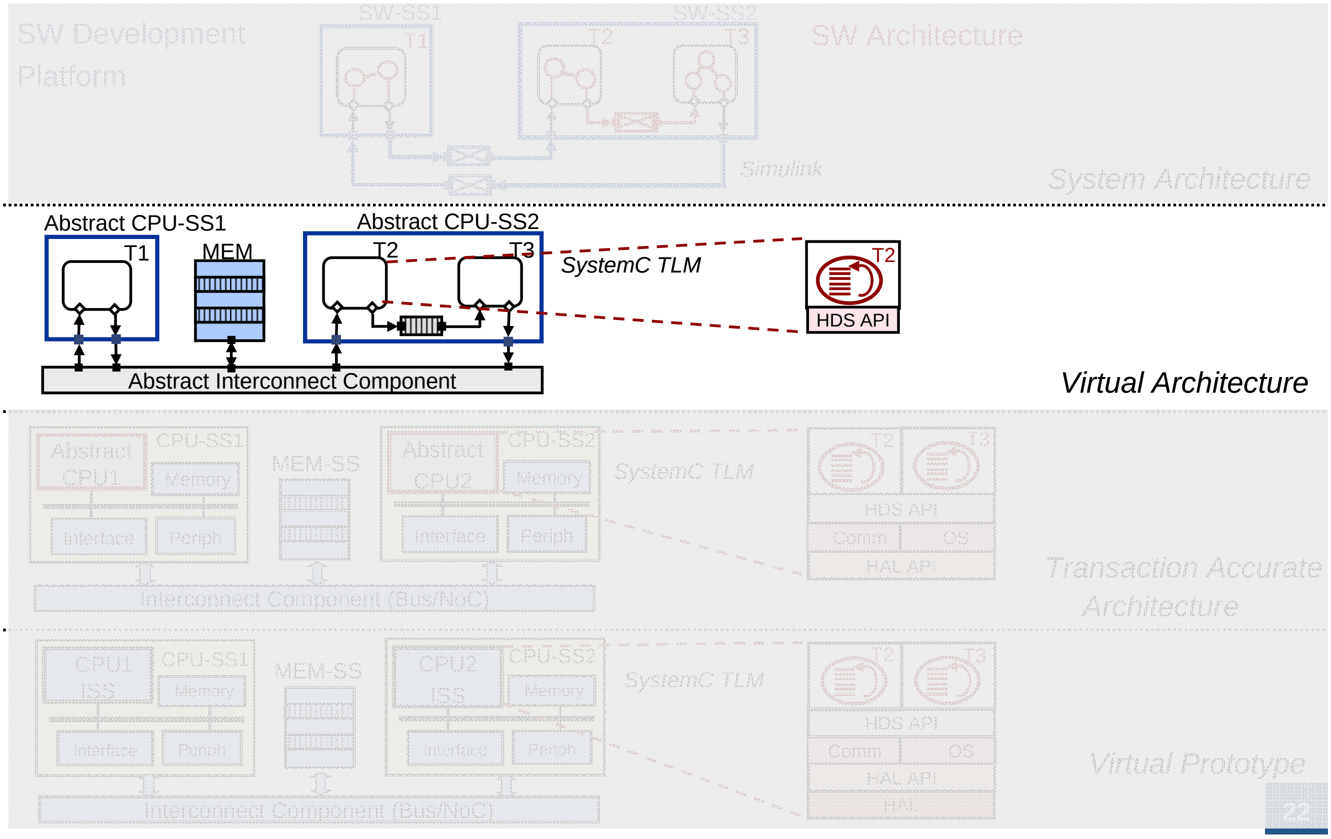
Priority:

Tag:

OK | Cancel | Help | Apply

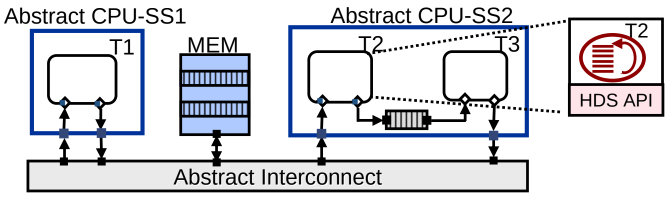
➢ Easy to experiment different communication schemes

Virtual Architecture Model



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

- **Software Architecture**
  - Task C code based on HdS APIs

Hardware platform

CPU-SS2 (SC_MODULE)

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

Task_T2 (SC_THREAD)

```
SC_MODULE (Task_T2){…
   SC_CTOR (Task_T2){
      SC_THREAD(task_T2, clk);
```

Communication channel

```
void recv (fifo_ch*  ch, void* dst, int size) {
        dst = ch->read (size);
}
```

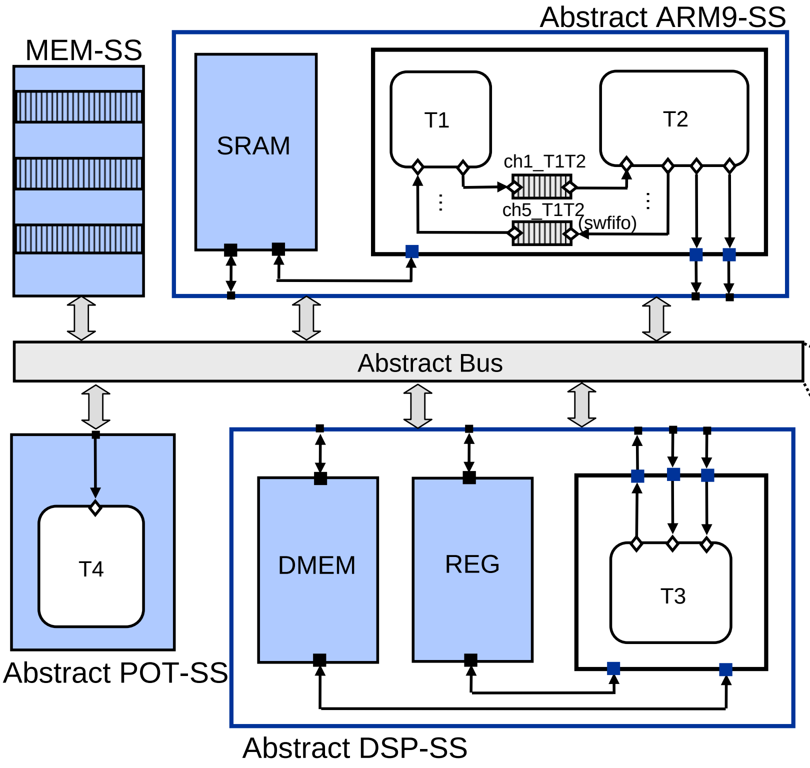
```
class fifo_ch : public sc_prim_channel {
word *buffer;
public:
        word * read (int size) {
              for (i=0; i<size; i++)
                      *(ret+i)=*(buffer+i);
               return ret;}
```

Task code of T2

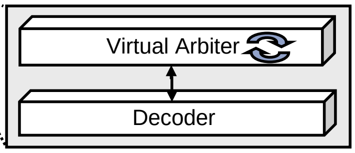
C code of Task_T2

```
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
}…
```

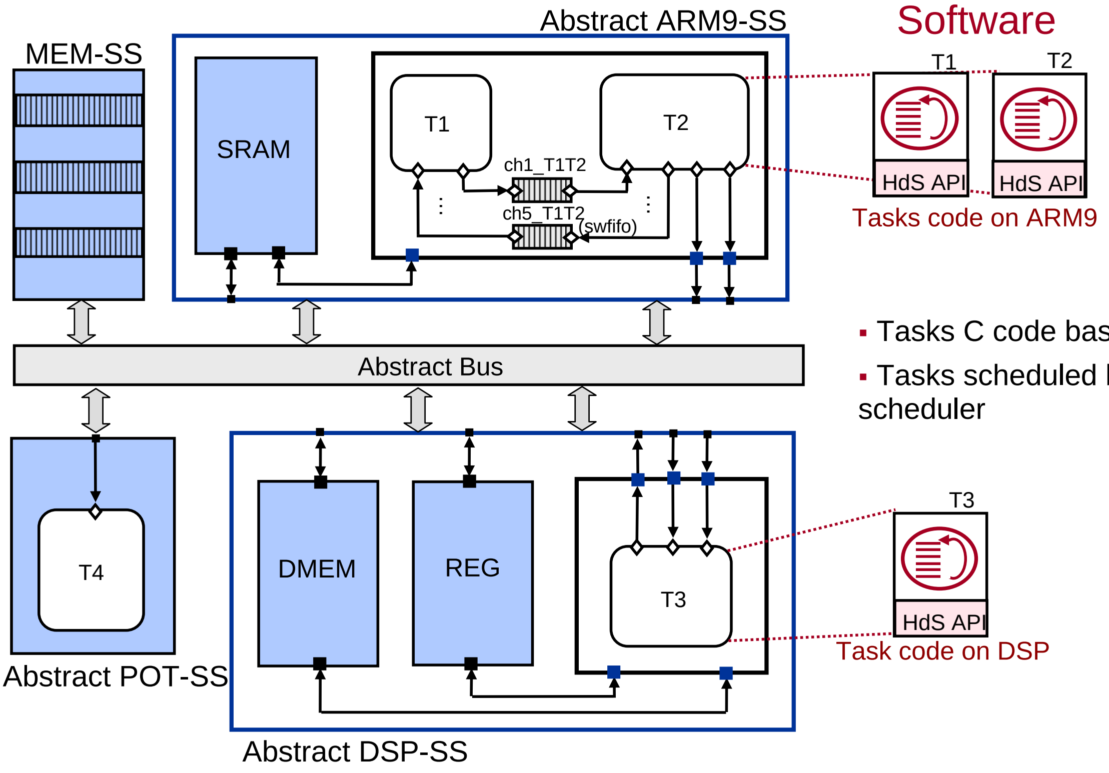
Application Example: M-JPEG Decoder mapped on Diopsis RDT



- 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



Application Example: M-JPEG Decoder mapped on Diopsis RDT



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

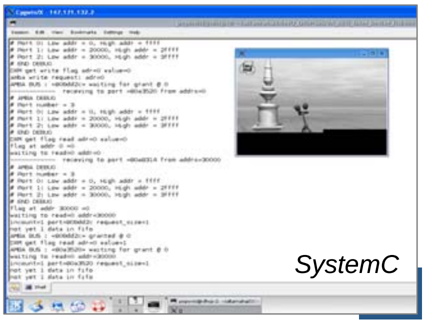
Results for the M-JPEG Decoder mapped on Diopsis RDT at the Virtual Architecture Level

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

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

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

Simulation Screenshot

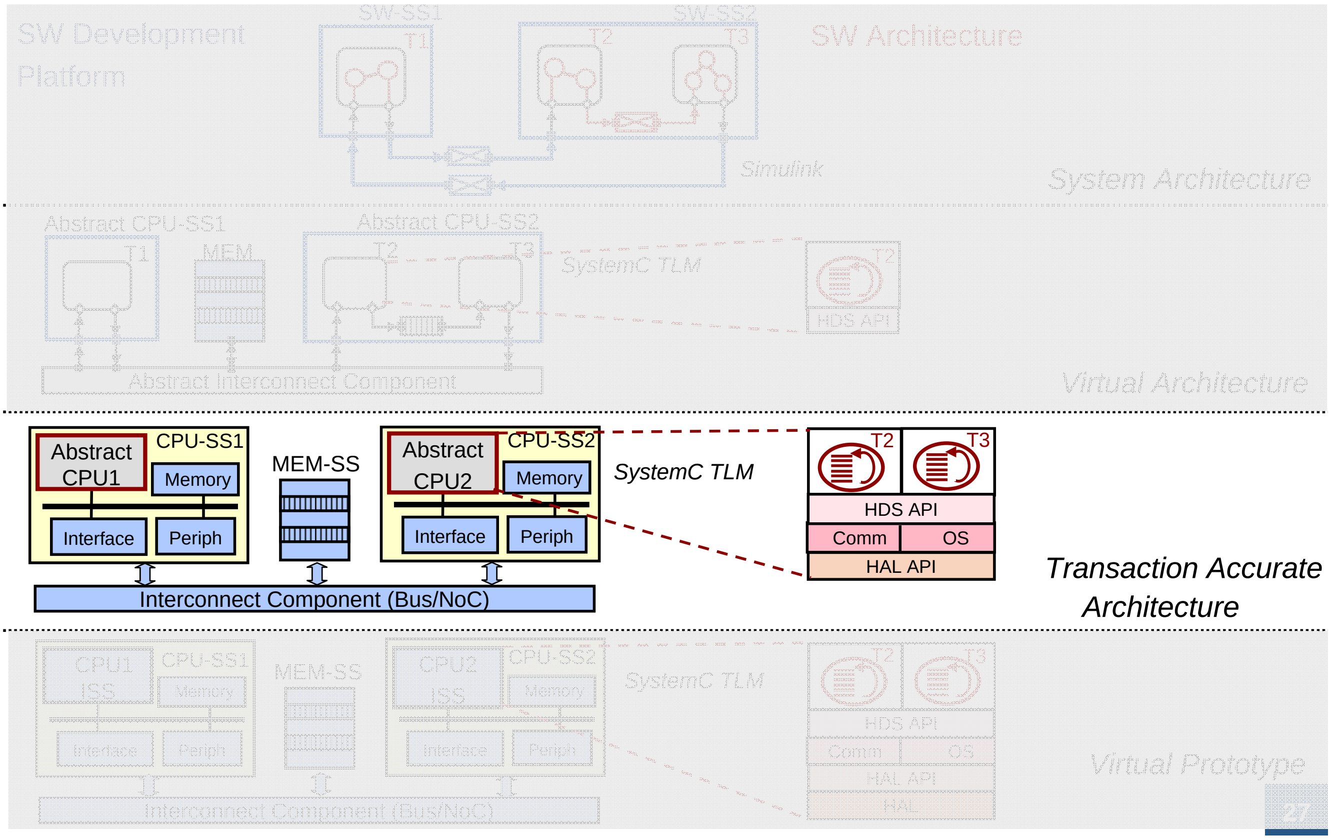


```
# Port 0: Low addr = 0, High addr = ffff
# Port 1: Low addr = 20000, High addr = 2ffff
# Port 2: Low addr = 30000, High addr = 3ffff
# END DEBUG
DXM get write flag adr=0 value=0
amba write request: adr=0
AMBA BUS : <[illegible]> waiting for grant @ 0
--------------  receiving to port <[illegible]> from addrs=0
# AMBA DEBUG
# Port number = 3
# Port 0: Low addr = 0, High addr = ffff
# Port 1: Low addr = 20000, High addr = 2ffff
# Port 2: Low addr = 30000, High addr = 3ffff
# END DEBUG
DXM get flag read adr=0 value=0
flag at addr 0 =0
waiting to read=0 addr=0
--------------  receiving to port <[illegible]> from addrs=30000
# AMBA DEBUG
# Port number = 3
# Port 0: Low addr = 0, High addr = ffff
# Port 1: Low addr = 20000, High addr = 2ffff
# Port 2: Low addr = 30000, High addr = 3ffff
# END DEBUG
flag at addr 30000 =0
waiting to read=0 addr=30000
incount=1 port=[illegible] request_size=1
not yet 1 data in fifo
AMBA BUS : <[illegible]> granted @ 0
DXM get flag read adr=0 value=1
AMBA BUS : <[illegible]> waiting for grant @ 0
waiting to read=0 addr=30000
incount=1 port=[illegible] request_size=1
not yet 1 data in fifo
not yet 1 data in fifo
```

SystemC

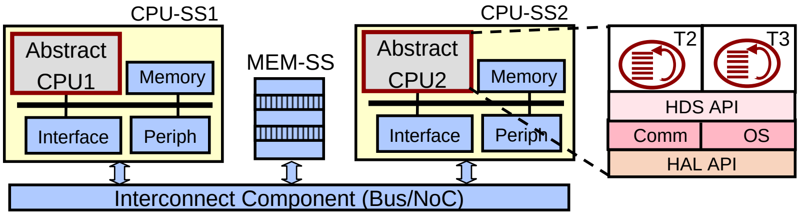
➢ Validation of task code and partitioning

Transaction Accurate Architecture Model



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

- **Simulation**
  - Task scheduled by the OS scheduler
  - Validation of OS & Comm integration

TA platform

TOP (SC_MODULE)

```
SC_MODULE (TOP){
  CPU2_SS *cpu2_ss; // Subsystems
  HW_SS *hw_ss;
  BUS *bus;              // Interconnection
```

CPU2-SS (SC_MODULE)

```
SC_MODULE (CPU_SS2) {
   Memory *mem;    // local components
   Periph *periph;
   CPU2 * cpu2; …

   Bus_port *busport; // Ports declation
```

SW Stack code on CPU–SS2

main

```
extern void task_T2 ();
void __start (void) {…
   create_task (task_T2);
```

Communication SW

```
void recv(ch, dst, size) {
   switch (ch.protocol){
   case FIFO:
      if  (ch.state==EMPTY) _schedule();
```

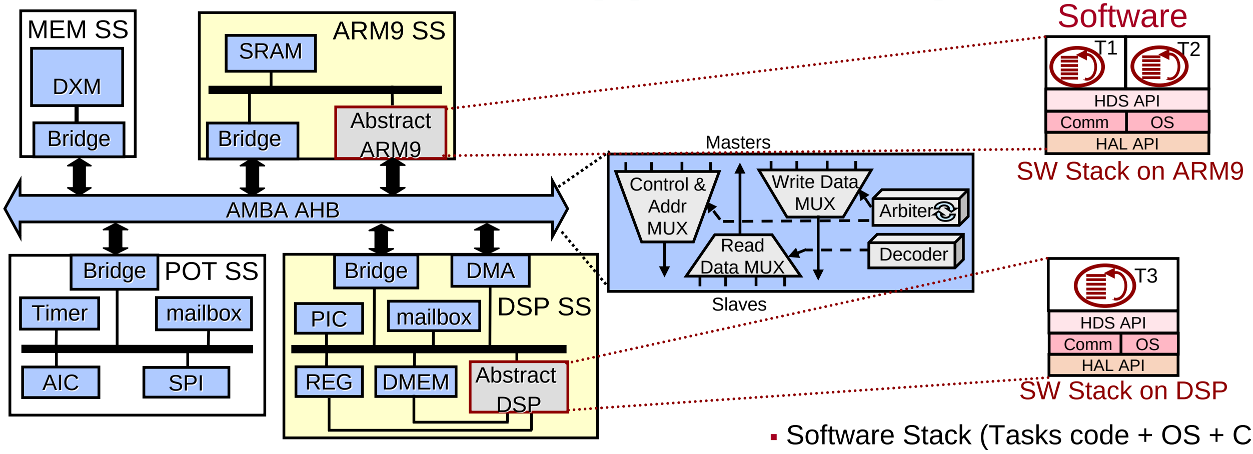
C code of Task_T2

```
void task_T2( ) {
 while (1) {
   recv (CH1, B, 10);
   …
   send (CH2, C, 20);
```

OS

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

Application Example: M-JPEG Decoder mapped on Diopsis RDT



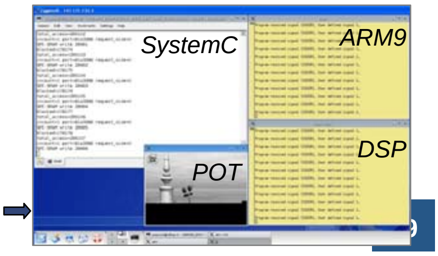
- Local SS architectures detailed
- Abstract CPU execution models
- AMBA bus protocol fully modeled
- Inter-SS communication fully mapped on explicit resources
- Intra-SS communication managed by OS

- Software Stack (Tasks code + OS + Comm) based on HAL API
- Tasks scheduled by OS scheduler

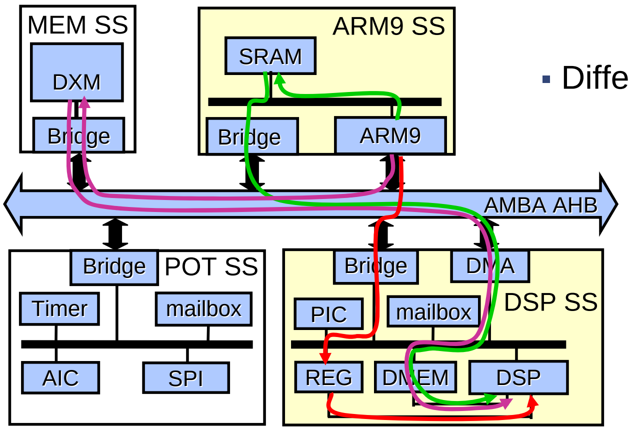
- Simulation time 5m10s (DXM+REG+SRAM), 10 frames QVGA

➢ Validation of OS and Comm. integration

Simulation Screenshot



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



- Different Communication Mapping Schemes

🟩 SRAM

🟪 DXM

🟥 REG

Communication Mapping Exploration

Communication Scheme	Transactions to memories [Bytes]				Total AMBA cycles	___
	DXM	SRAM	REG	DMEM		
DXM+DXM+DXM	5256k	0	0	0	8856k	**100%**
DXM+REG+DMEM	4608k	0	72k	576k	7884k	**89%**
SRAM+SRAM+DMEM	0	4680k	0	576k	3960k	**45%**

➢ 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:

- P.S. PAOLUCCI, A.A. JERRAYA, R. LEUPERS, L. THIELE, P. VINCINI “SHAPES : a tiled scalable software hardware architecture platform for embedded systems”, *Proceeding of CODES+ISSS 2006*, Seoul, Korea, pp. 167-172

- K. POPOVICI, X. GUERIN, F. ROUSSEAU, P.S. PAOLUCCI, A. JERRAYA “Platform based Software Design Flow for Heterogeneous MPSoC”, *ACM Transactions on Embedded Computing Systems (TECS)*, Accepted 17th January 2008

- X. GUERIN, K. POPOVICI, W. YOUSSEF, F. ROUSSEAU, A. JERRAYA “Flexible Application Software Generation for Heterogeneous Multi-Processor System-on-Chip”, *31st Annual International Computer Software and Applications Conference (COMPSAC’07)*, 23-27 July 2007, Beijing, China

- K. HUANG, S.I. HAN, K. POPOVICI, L. BRISOLARA, X. GUERIN, L. LI, X. YAN, S.I. CHAE, L. CARRO, A. JERRAYA “Simulink based MPSoC Design Flow: Case Study of Motion JPEG and H.264”*, Design Automation Conference (DAC’07)*, 4-8 June 2007, San Diego, USA

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