Simulink®-based Programming Environment for Heterogeneous MPSoC

Katalin Popovici
katalin.popovici@mathworks.com

Software Engineer, The MathWorks

DATE 2009, Nice, 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 embedded applications
    - TI OMAP, ST Nomadik, NXP 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

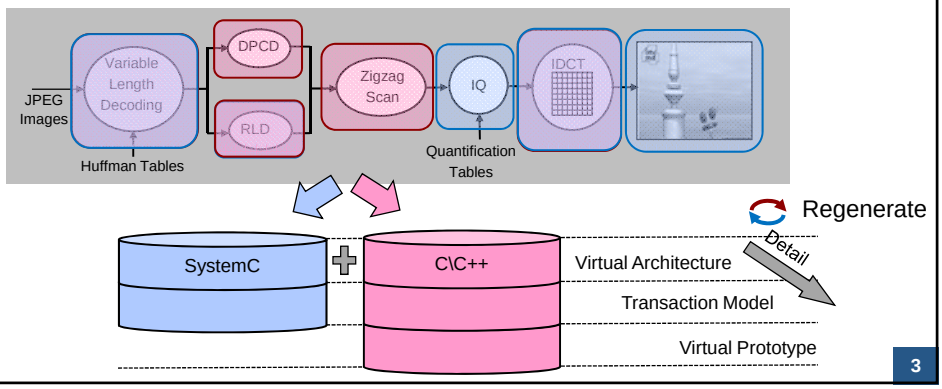
- **Challenge:**
  - Efficient and fast programming environment for heterogeneous MPSoC

2

The MathWorks MATLAB&SIMULINK®

Proposal

- Simulink based SW development and validation environment
  - Combined architecture and application model in Simulink
    - Architecture markers to generate executable SystemC models
  - Performance analysis with different levels of details
  - Computation and communication mapping exploration



3

The MathWorks MATLAB&SIMULINK®

Outline

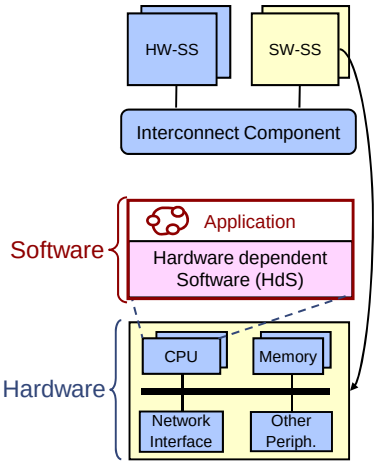
- **Introduction**
- **Software Design and Validation**
  - **System Architecture**
  - **Virtual Architecture**
  - **Transaction Accurate Architecture**
- **Conclusions**

4

The MathWorks MATLAB&SIMULINK

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 & synchronization

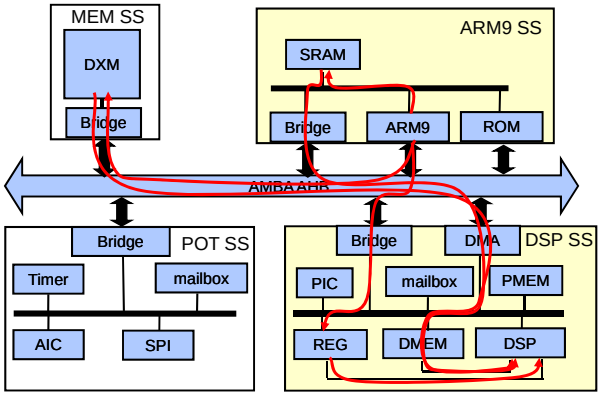


5

The MathWorks MATLAB&SIMULINK

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

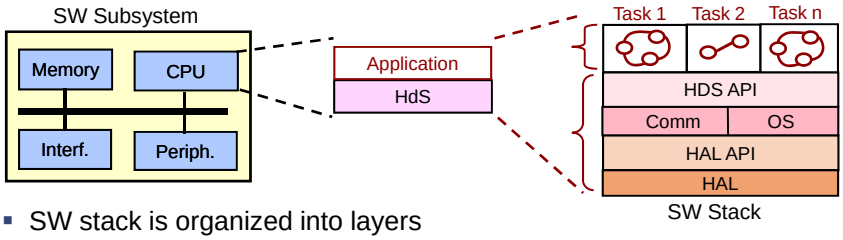


- Local & global memories accessible by both processing units
  - Different communication schemes between CPUs
- Require multiple software stacks (ARM + DSP)

6

The MathWorks MATLAB&SIMULINK®

Software Stack Organization

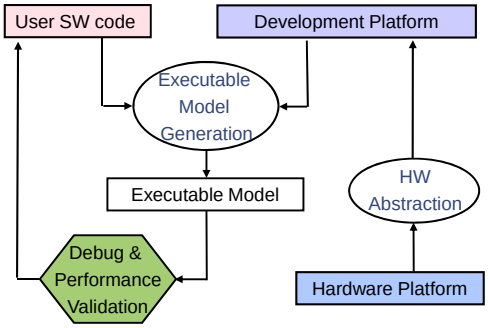


- SW stack is organized into layers
  - Application code
    - SW code of tasks mapped on the CPU
  - HdS (Hardware dependent Software) made of different components
    - OS (Operating System)
    - Comm (Communication Primitives)
    - HAL (Hardware Abstraction Layer)
    - API (Application Programming 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

7

The MathWorks MATLAB&SIMULINK®

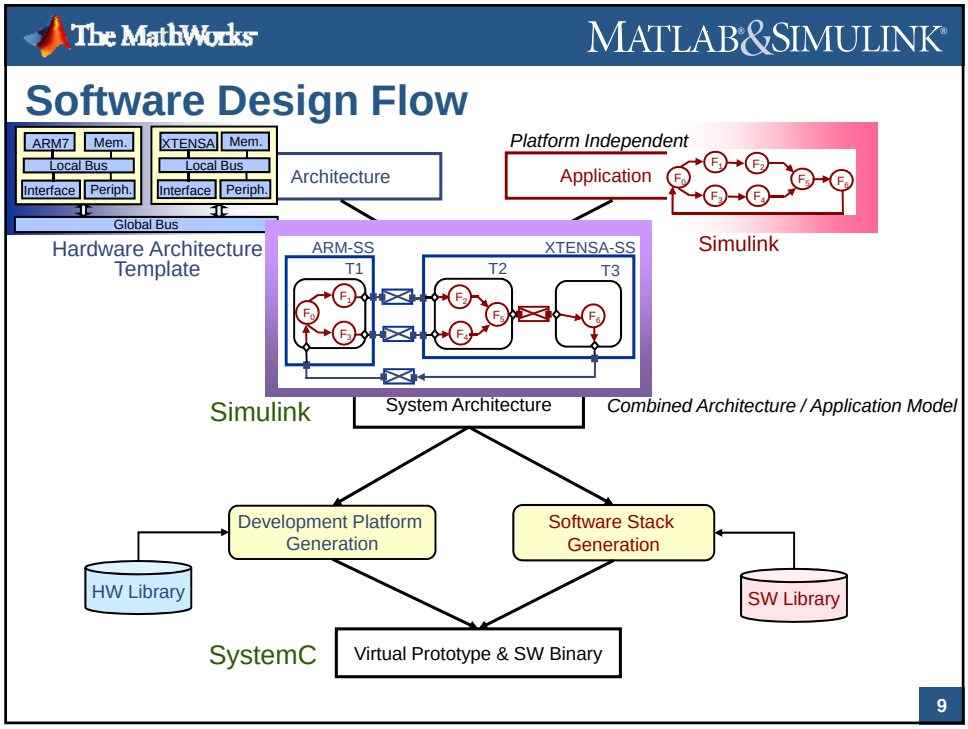
Software Development Platform

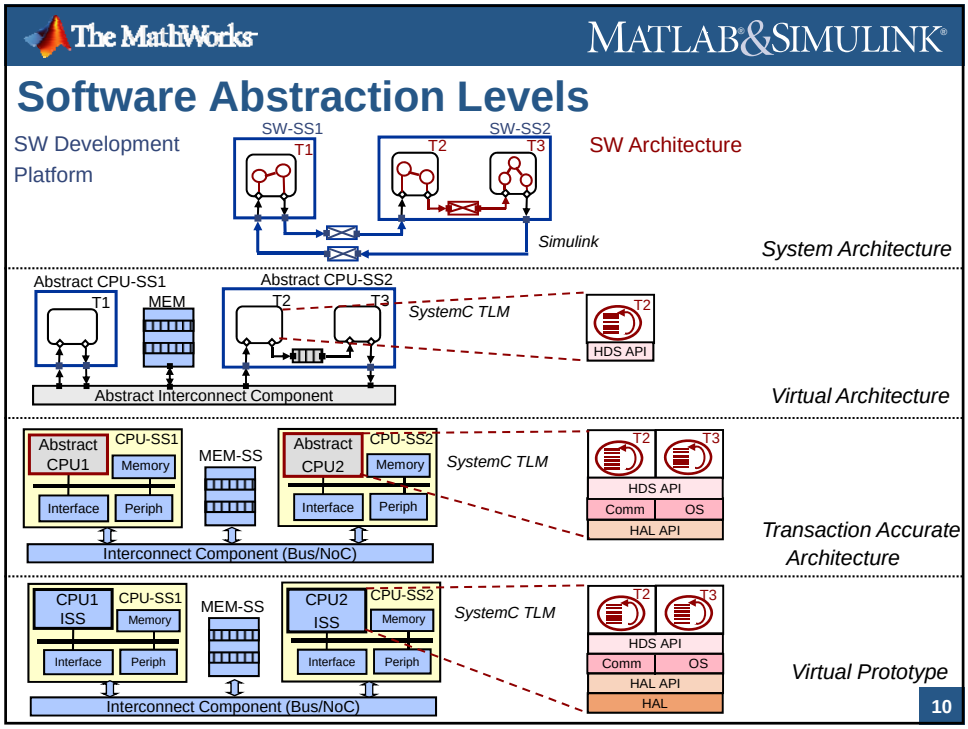


- User SW Code
  - C/C++, Simulink functions, binary,…
- Development platform to abstract architectures
  - Runtime library, simulator (ISS, Simulink)
- Executable model generation
  - Compile, Link
- Debug
  - Iterative process
  - Different SW components need different detail levels

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

8

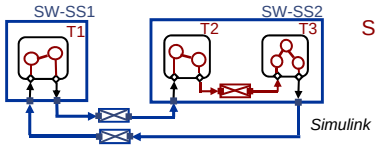

The MathWorks
MATLAB&SIMULINK
Software Design Flow
ARM7
Mem.
Local Bus
Interface
Periph.
XTENSA
Mem.
Local Bus
Interface
Periph.
Global Bus
Hardware Architecture Template
Architecture
Platform Independent
Application
Simulink
ARM-SS
XTENSA-SS
T1
T2
T3
Simulink
System Architecture
Combined Architecture / Application Model
Development Platform Generation
Software Stack Generation
HW Library
SW Library
SystemC
Virtual Prototype & SW Binary
9


The MathWorks
MATLAB&SIMULINK
Software Abstraction Levels
SW Development Platform
SW-SS1
T1
SW-SS2
T2
T3
SW Architecture
Simulink
System Architecture
Abstract CPU-SS1
T1
MEM
Abstract CPU-SS2
T2
T3
SystemC TLM
T2
HDS API
Abstract Interconnect Component
Virtual Architecture
Abstract CPU1
CPU-SS1
Memory
Interface
Periph
MEM-SS
Abstract CPU2
CPU-SS2
Memory
Interface
Periph
SystemC TLM
T2
T3
HDS API
Comm
OS
HAL API
Interconnect Component (Bus/NoC)
Transaction Accurate Architecture
CPU1 ISS
CPU-SS1
Memory
Interface
Periph
MEM-SS
CPU2 ISS
CPU-SS2
Memory
Interface
Periph
SystemC TLM
T2
T3
HDS API
Comm
OS
HAL API
HAL
Interconnect Component (Bus/NoC)
Virtual Prototype
10

The MathWorks MATLAB&SIMULINK

Software Abstraction Levels

SW Development Platform



SW Architecture

Simulink

System Architecture

Models to debug the SW

SW Design

Virtual Architecture

Transaction Accurate Architecture

Virtual Prototype

11

The MathWorks MATLAB&SIMULINK

Outline

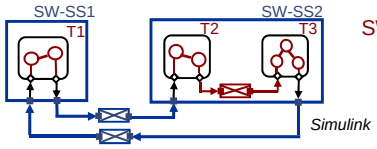
- **Introduction**
- **Software Design and Validation**
  - **System Architecture**
  - **Virtual Architecture**
  - **Transaction Accurate Architecture**
- **Conclusions**

12

The MathWorks MATLAB&SIMULINK®

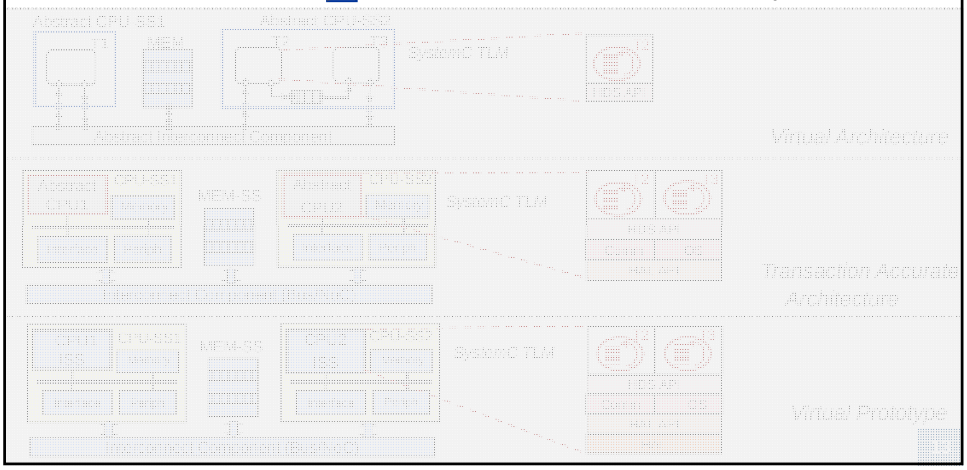
System Architecture Model

SW Development Platform



SW Architecture

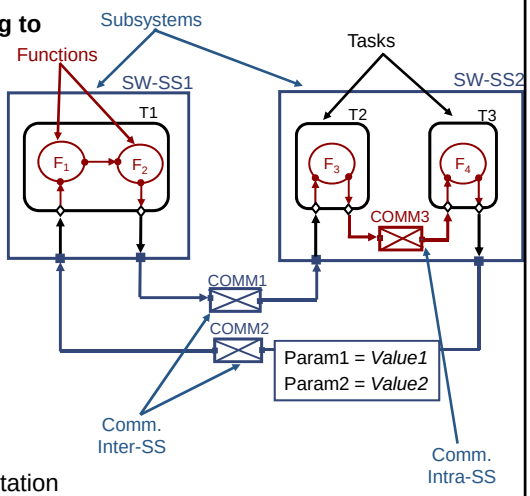
System Architecture



The MathWorks MATLAB&SIMULINK®

System Architecture Design

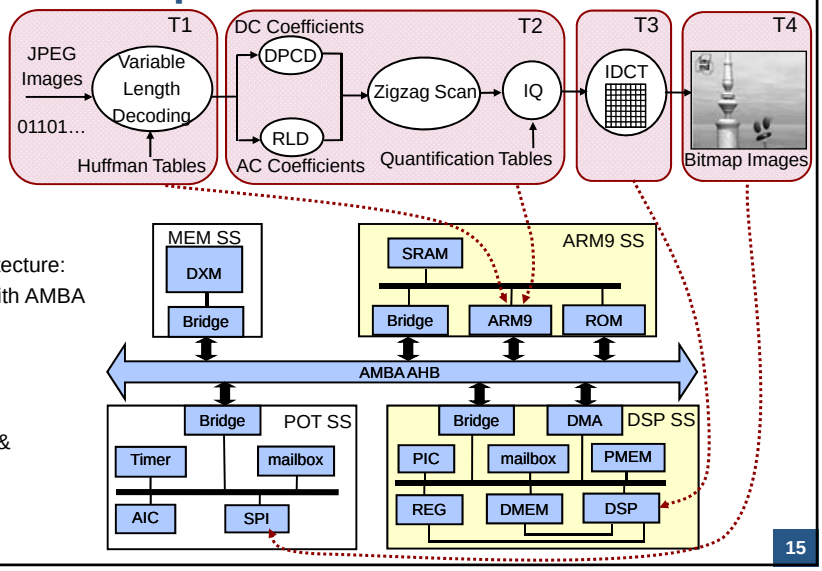
- **Captures application and mapping to architecture**
- **Task**
  - Set of functions:
    - Simulink blocks, S-functions
- **Subsystem**
  - Set of tasks
- **Abstract communication types**
  - Communication Intra-SS
  - Communication Inter-SS
- **Communication protocol**
  - Generic Simulink I/Os
  - Explicit annotation for implementation
- **Algorithm validation through simulation**



The MathWorks MATLAB&SIMULINK

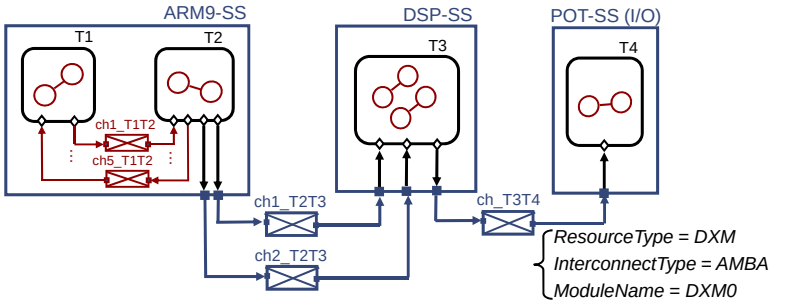
Application Example: M-JPEG Decoder Mapped on Diopsis RDT

- Application specification (~ 1700 LOC)
- Target architecture: Diopis RDT with AMBA
- Partitioning & mapping



The MathWorks MATLAB&SIMULINK

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 markers annotating the model:

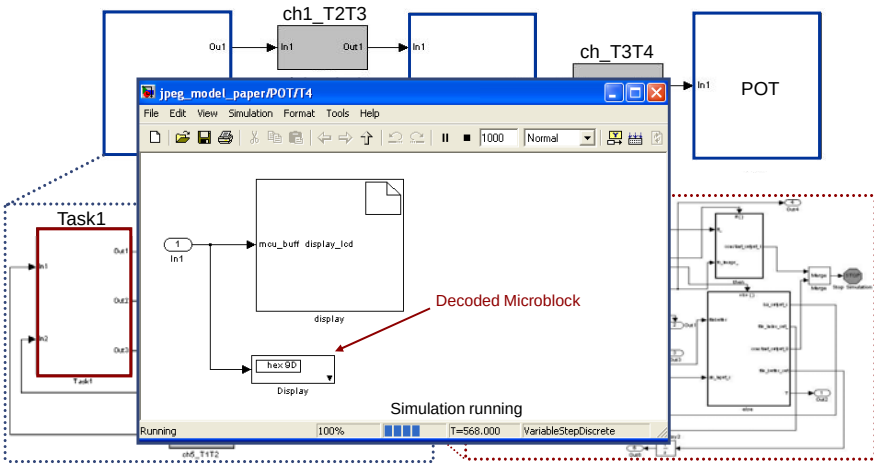
- ResourceType
  - ARM9, DSP, POT, Task
  - Communication: swfifo, dmem, sram, reg, dxm
- NetworkType: AMBA_AHB, NoC
- AccessType: DMA, direct
- MemName

➢ Validation of application functionality

The MathWorks MATLAB&SIMULINK®

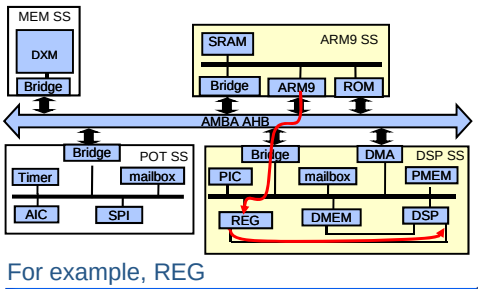
MJPEG System Architecture in Simulink

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



The MathWorks MATLAB&SIMULINK®

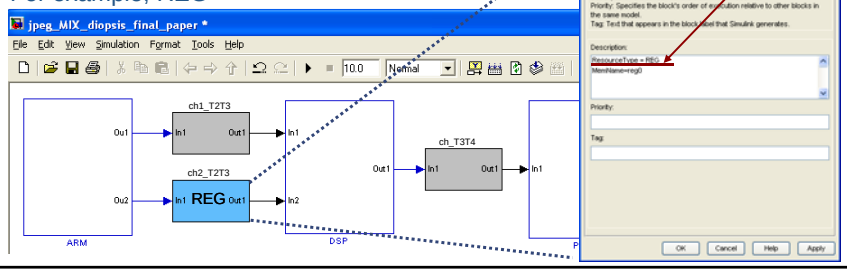
Capture of Low Level Architecture Features in Simulink for MJPEG



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

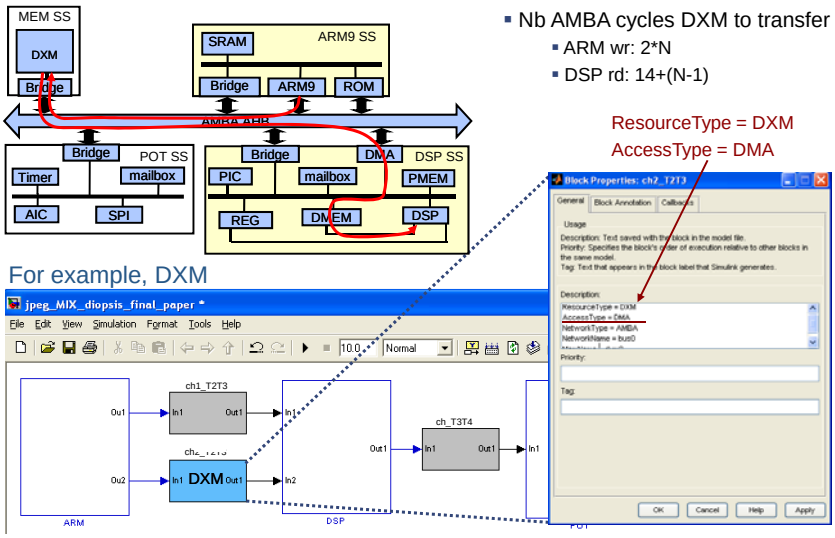
For example, REG

ResourceType = REG



The MathWorks MATLAB&SIMULINK®

Capture of Low Level Architecture Features in Simulink for MJPEG

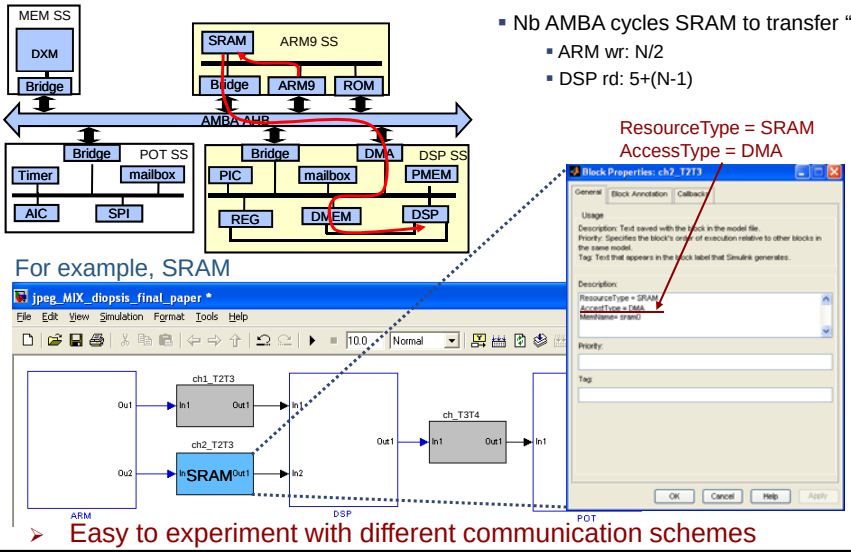


- Nb AMBA cycles DXM to transfer “N” words
  - ARM wr: 2*N
  - DSP rd: 14+(N-1)

19

The MathWorks MATLAB&SIMULINK®

Capture of Low Level Architecture Features in Simulink for MJPEG

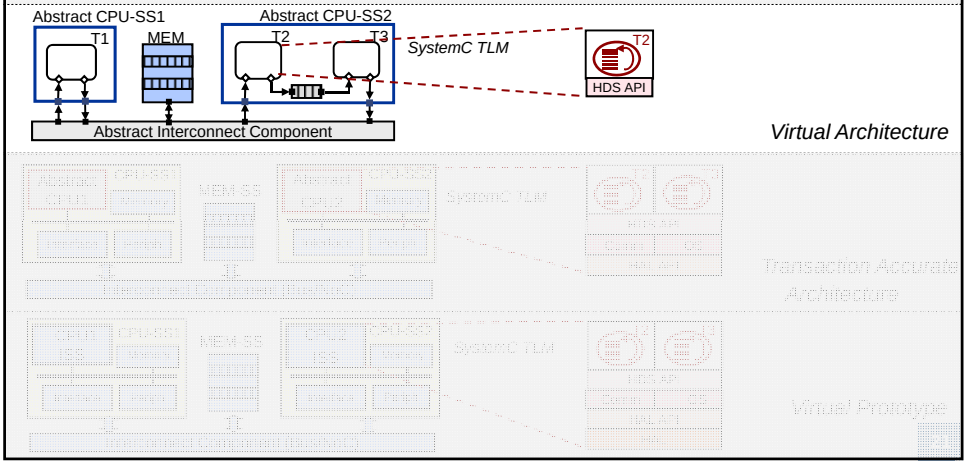


- Nb AMBA cycles SRAM to transfer “N” words
  - ARM wr: N/2
  - DSP rd: 5+(N-1)

➢ Easy to experiment with different communication schemes

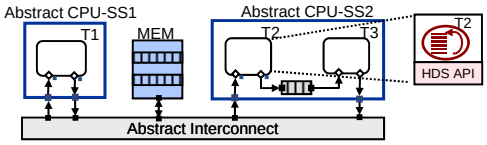
20

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)	Communication channel

SC_MODULE (CPU_SS2) {
Task_T2 * T2; // *tasks*
in_port *in; // *ports*
out_port *out;
fifo_ch*ch1; // *channels*
	void recv *(fifo_ch* ch, void* dst, int size)* {
	dst = ch->read (*size*);
	}

Task_T2 (SC_THREAD)	*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;}

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

Task code of T2

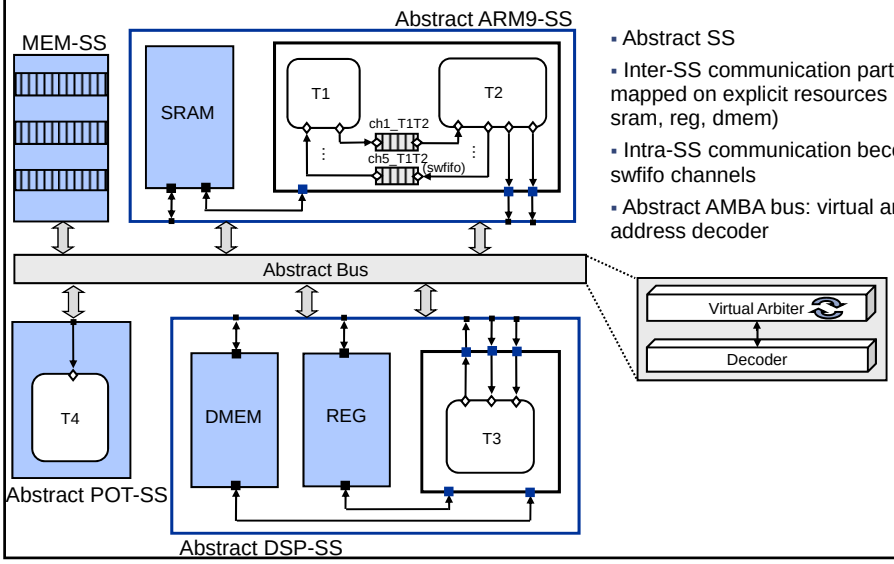
C code of Task_T2

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
}…

22

The MathWorks MATLAB&SIMULINK

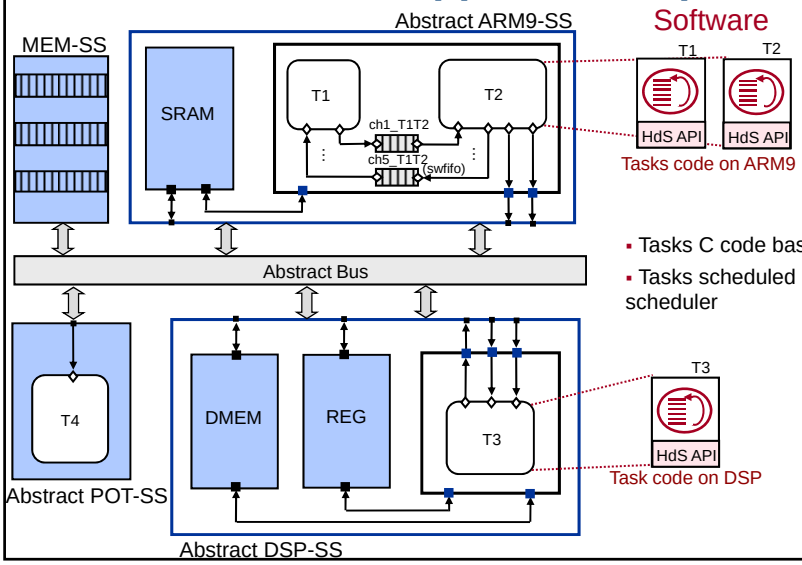
Application Example:
M-JPEG Decoder mapped on Diopsis RDT



- 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

The MathWorks MATLAB&SIMULINK

Application Example:
M-JPEG Decoder mapped on Diopsis RDT



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

The MathWorks MATLAB&SIMULINK®

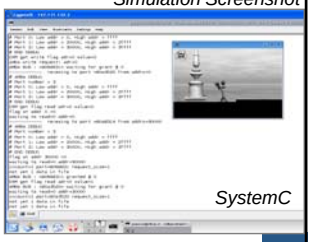
Results for the M-JPEG Decoder mapped on Diopsis RDT at the Virtual Architecture Level

- 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

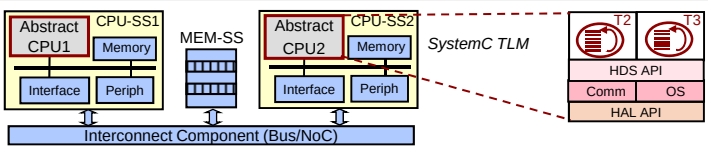


➢ Validation of task code and partitioning

25

The MathWorks MATLAB&SIMULINK®

Transaction Accurate Architecture Model

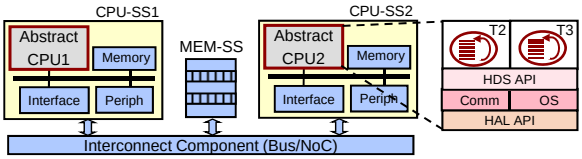


Transaction Accurate Architecture

The MathWorks MATLAB&SIMULINK

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)



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

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

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	Communication SW

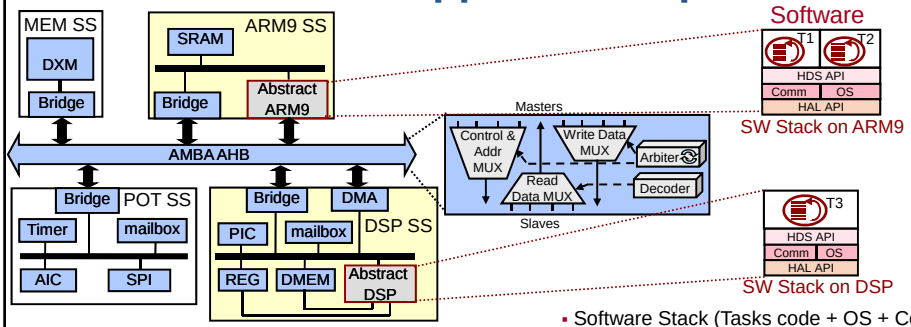
extern void task_T2 ();
void __start (*void*) {…
create_task (*task_T2*);
	void recv(*ch, dst, size*) {
	switch (ch.protocol){
	case FIFO:
	if (ch.state==EMPTY) _schedule();

C code of Task_T2	**OS**

void task_T2( ) {
while (1) {
recv (*CH1, B, 10*);
…
send (*CH2, C, 20*);
	void __schedule(*void*) {
	int old_tid = cur_tid;
	cur_tid = new_tid;
	__cxt_switch(*old_tid.cxt, cur_tid.cxt*);
	…}

The MathWorks MATLAB&SIMULINK

Application Example: M-JPEG Decoder mapped on Diopsis RDT



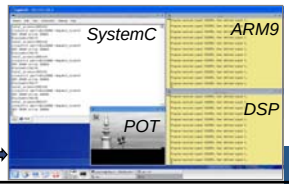
- Local SS architectures detailed
- Abstract CPU execution models
- AMBA bus protocol fully modeled
- Inter-SS communication fully mapped on explicit resources
- Intra-SS communication managed by OS

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

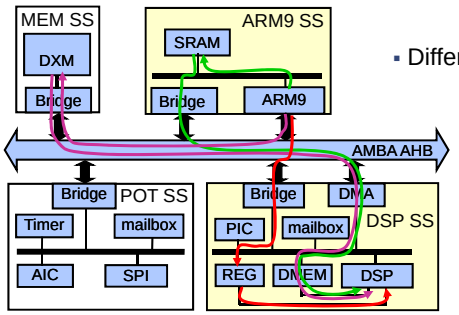
➢ Validation of OS and Comm integration

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

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] DXM	SRAM	REG	DMEM	Total AMBA cycles	___
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 communication performance by using HW resources

Outline

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

Conclusions

- 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:
  - Using markers in the Simulink model to allow automatic generation of mapping schemes
  - Allow early performance estimation at different levels of details
  - Allow the efficient use of architecture resources
- Easily experiment with several computation and communication mapping schemes

Thank you!

Katalin Popovici

{katalin.popovici@mathworks.com}

The MathWorks

Acknowledgement

- Dr. Ahmed Jerraya, CEA-LETI, France
- Prof. Frederic Petrot, TIMA Labs, France
- Prof. Frederic Rousseau, TIMA Labs, France
- Pieter Mosterman, The MathWorks, USA