



## *Lab 4*

# ***Platform Architect (I)***

v2 TA: 謝明倫

v1 TA: 陳宥融

# Objectives – the following 2 w:

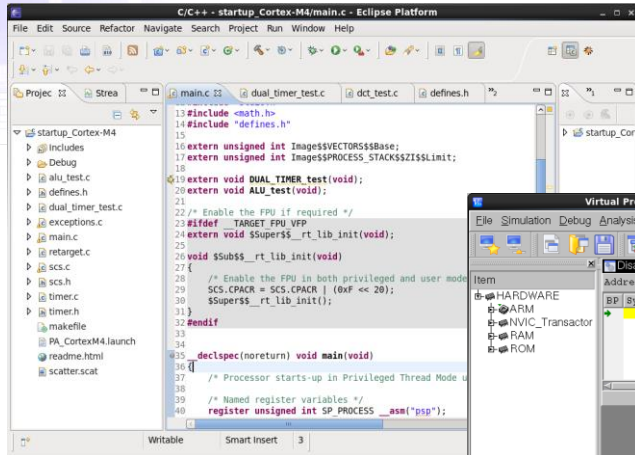
- Doing hardware-software co-simulation under ESL environment
- Be familiar with Platform Architect
  - SystemC transaction-level system build up
  - Packing HDL module (with AHB wrapper)
  - SystemC & RTL HDL co-simulation

# Objectives – today:

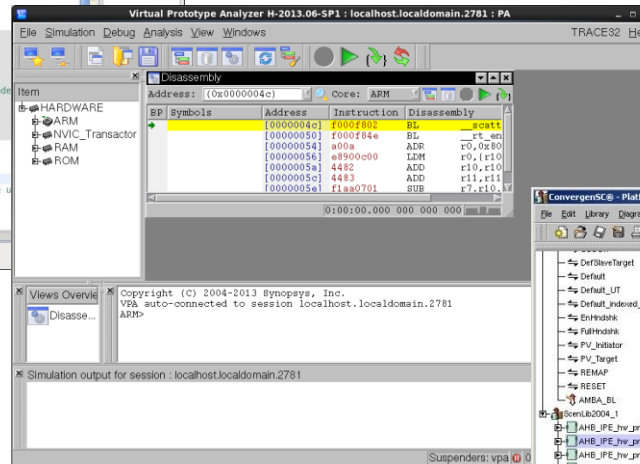
- SystemC transaction-level system build up
  - What is Platform Architect ?
  - Build up our system
  - Transaction-level system model software simulation / debug

# Platform Architect consists of

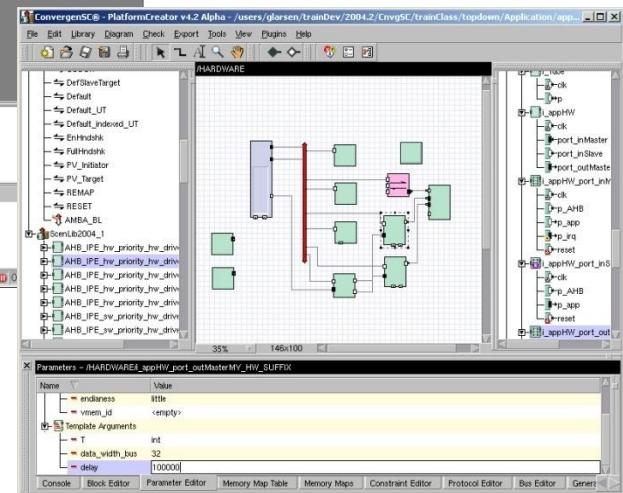
DS-5  
(SystemC IDE)



Virtual Prototype Analyzer  
(Analysis API)



System Designer  
(Platform Creator)



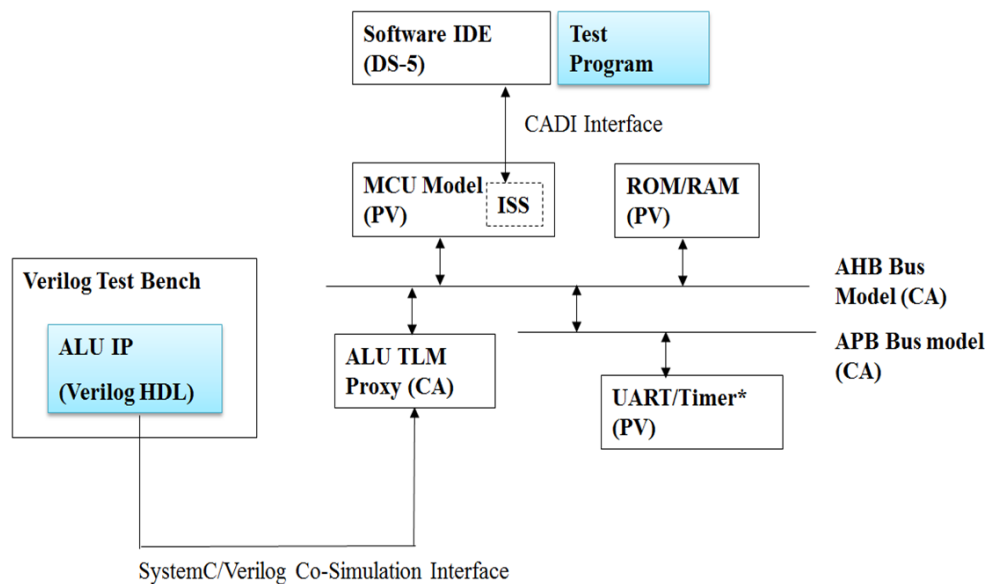
Simulation  
and  
Debugging

Analysis

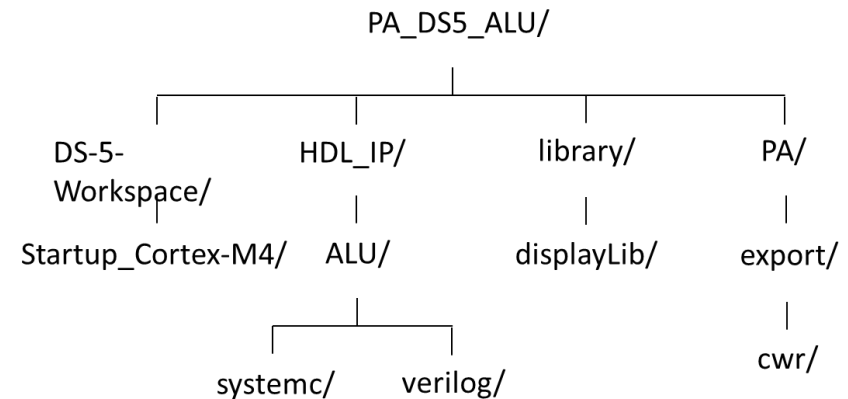
Platform  
Assembly and  
Configuration

# Top view

## ■ Target architecture



## ■ Files





# Using Platform Architect

- `source pa_setup.csh`

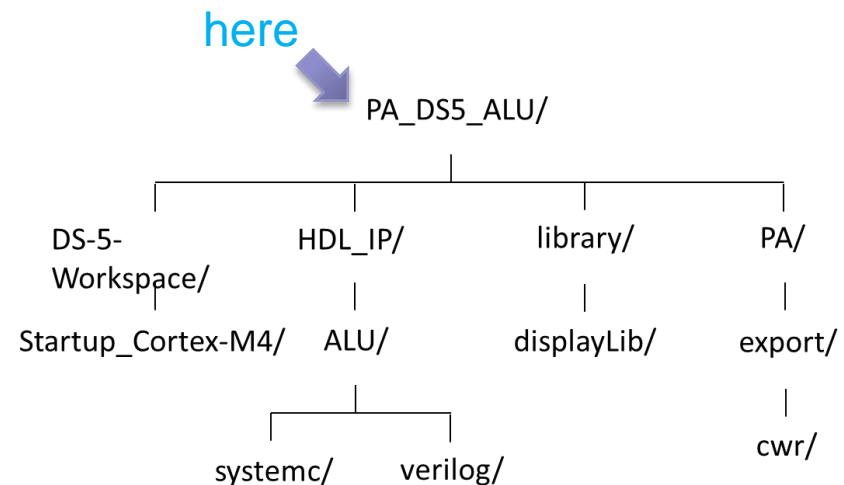
- `source run.csh` ← *To turn on tools for the project*

- ☐ To start Platform Creator:

- `% pct &`

- ☐ To start VPA

- `% vpa &`







# PA – main environment: Platform Creator

IP Templates

Model  
Libraries

*Library  
Drawer*

Block Diagram  
Editor

*System  
Diagram*

Details Area

*Parameters,  
Block editing,  
Memory Map,  
Command Line Interface*

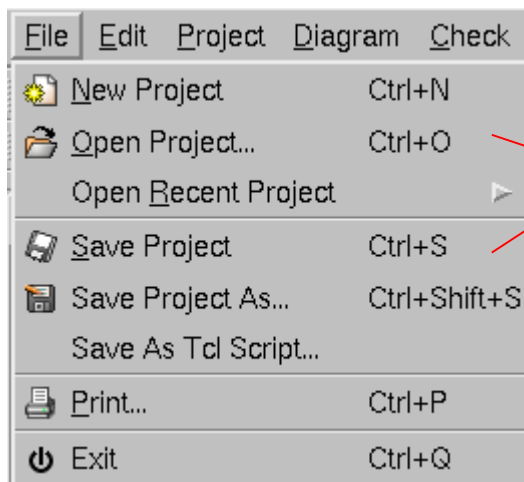
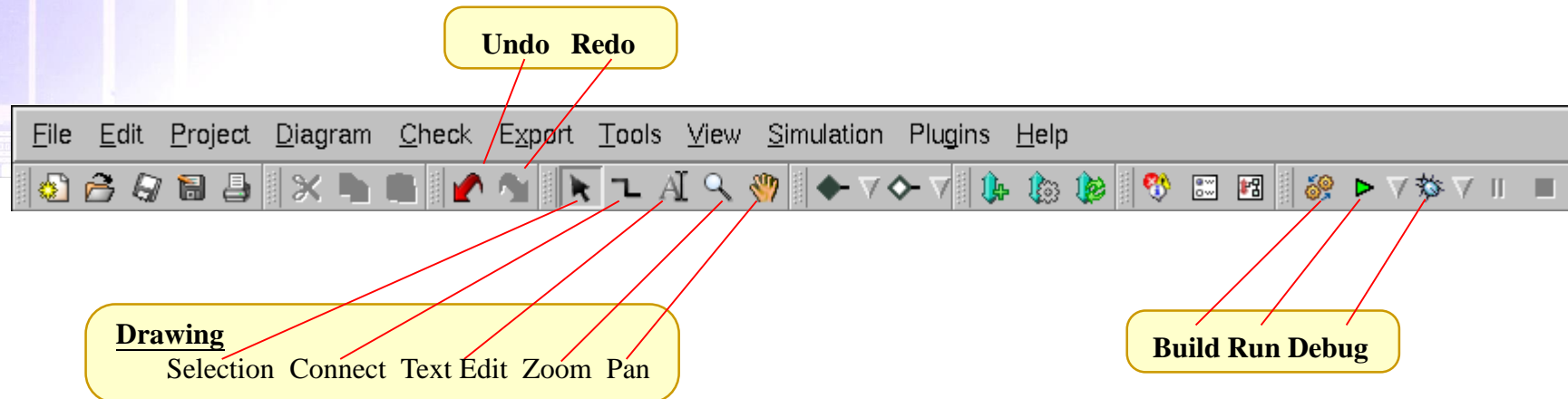
*System  
Hierarchy  
Tree*

Design  
Browser

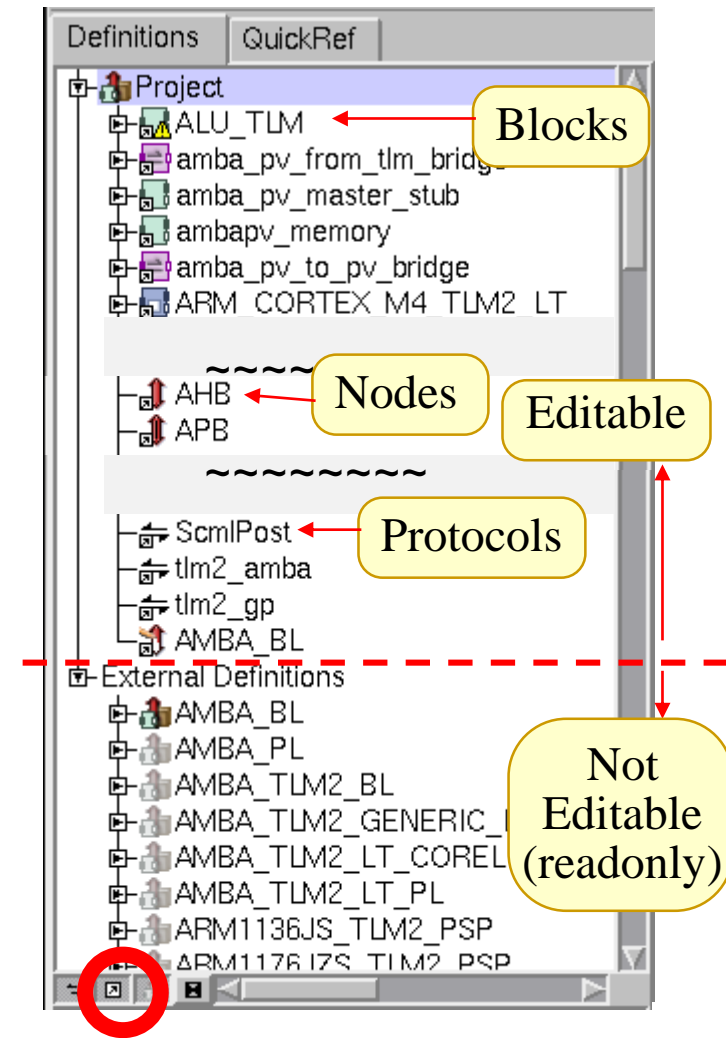
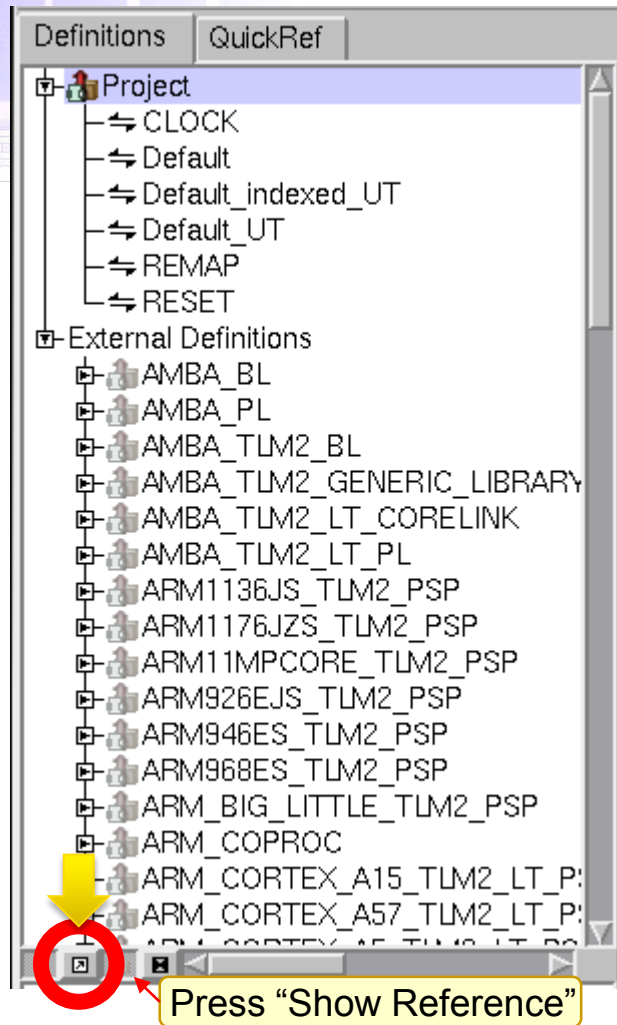
IP Instances



# Menus and the toolbar



# A closer look at model libraries



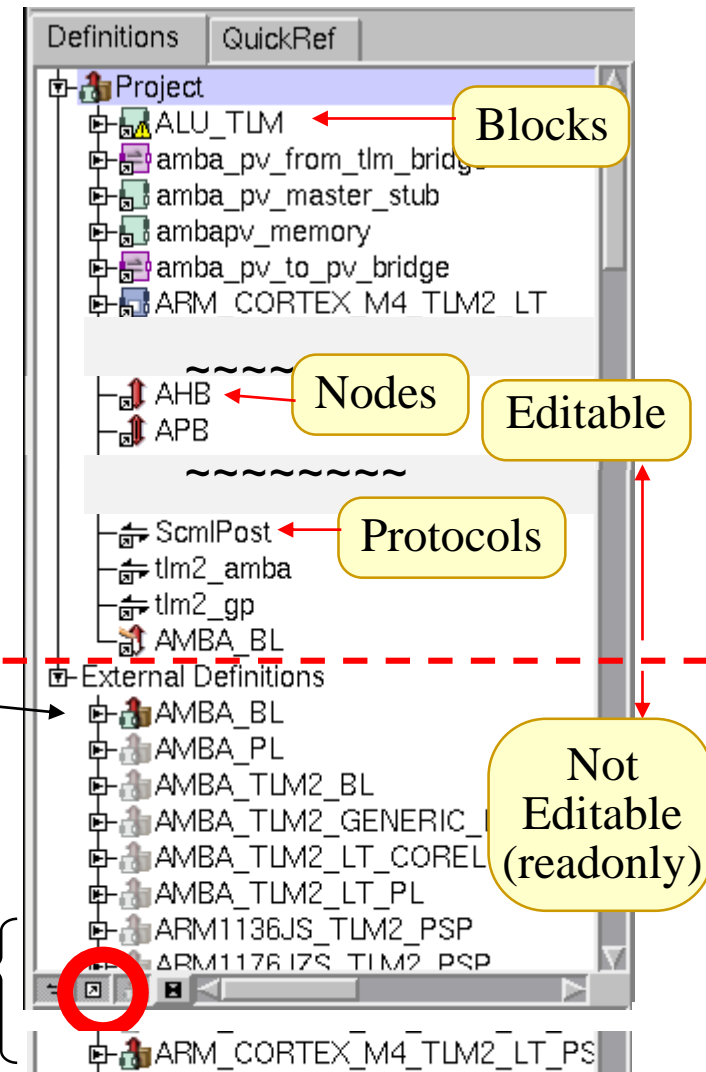
# A closer look at model libraries

- System Library

- Your system workspace, the system under construction
- Displays the system blocks, nodes, and protocols of your design

- Currently Open IP Libraries

- AMBA Bus Library
  - Nodes
  - Bridges
  - Protocols
- Processor Library



# Typical usage - drag and drop

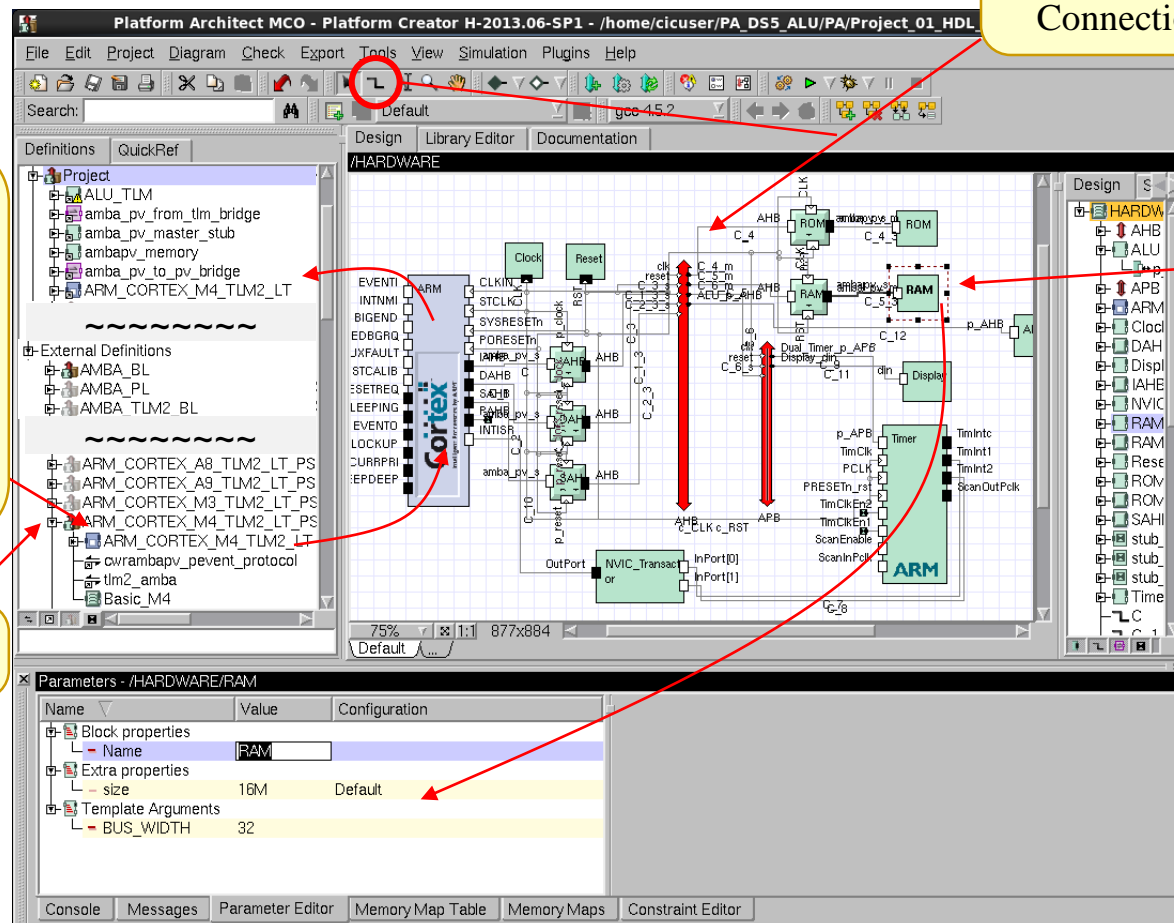
2. **Drag and drop** blocks and bus nodes from the IP libraries into the Design Editor window. The block appears in your Workspace

1. **Open IP libraries**

3. **Connect** elements with Connection Tool

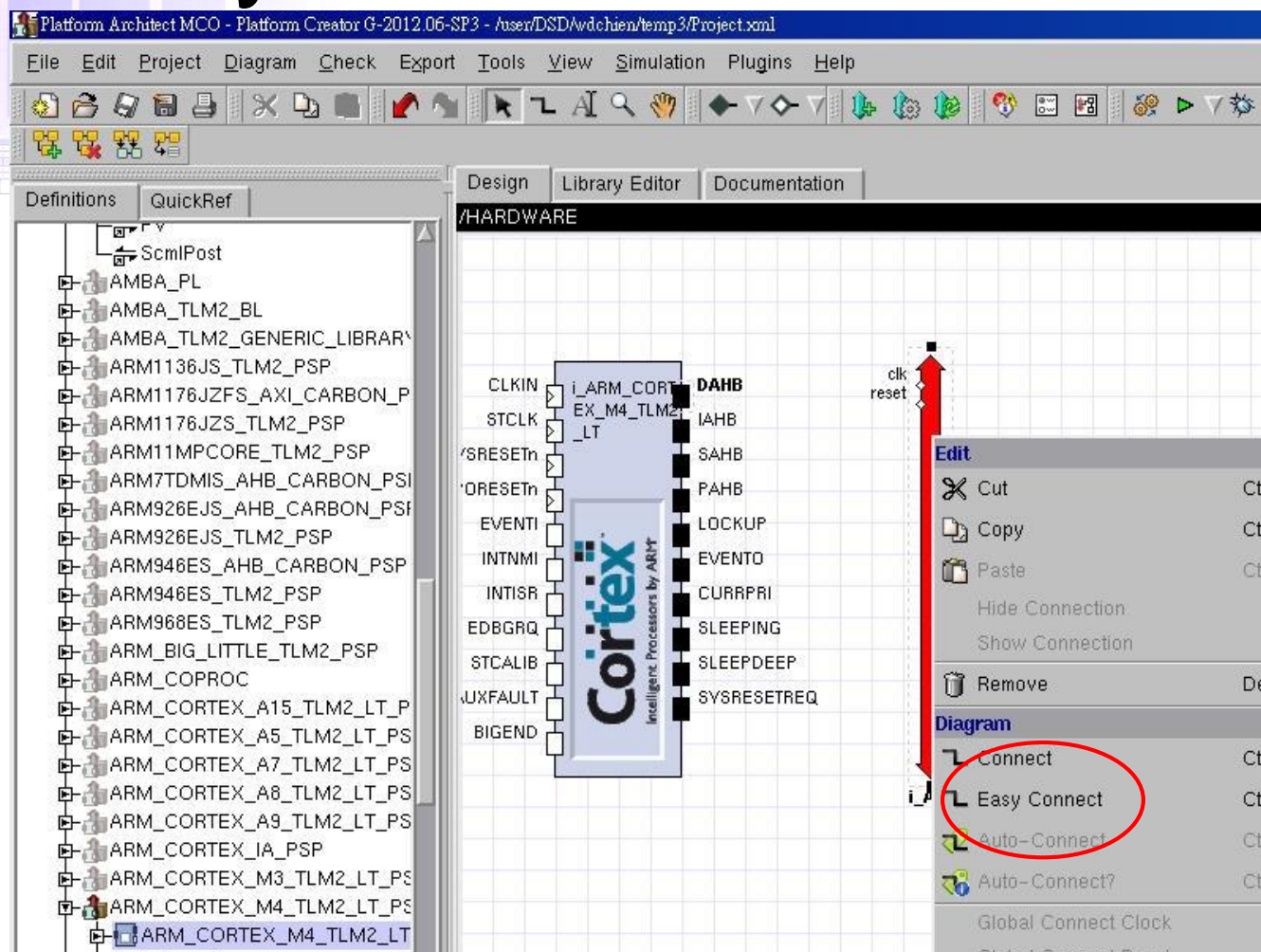
4. **Select** blocks and nodes to set memory map and other parameters

5. **Export** design for simulation build



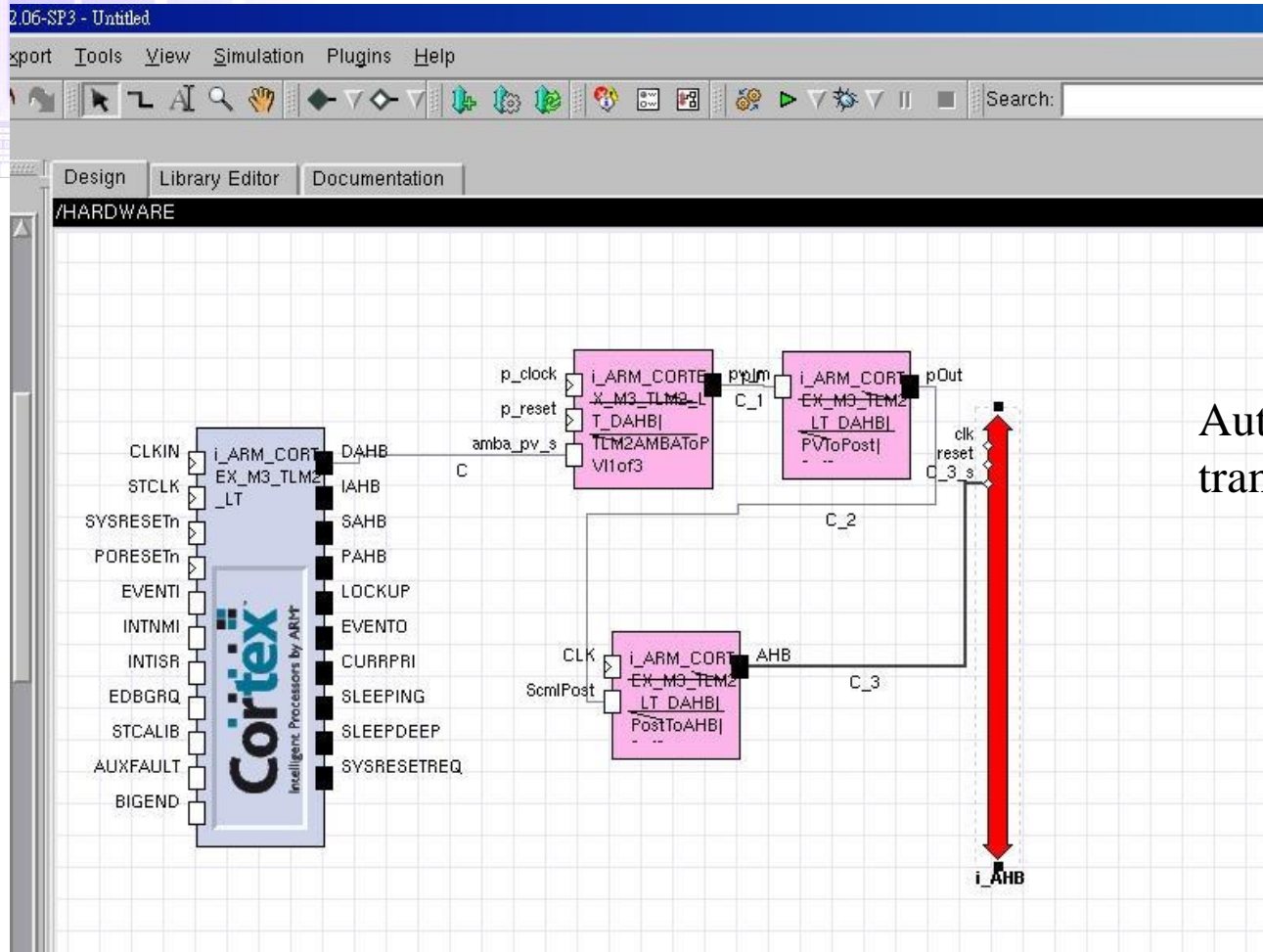


# Easy connect





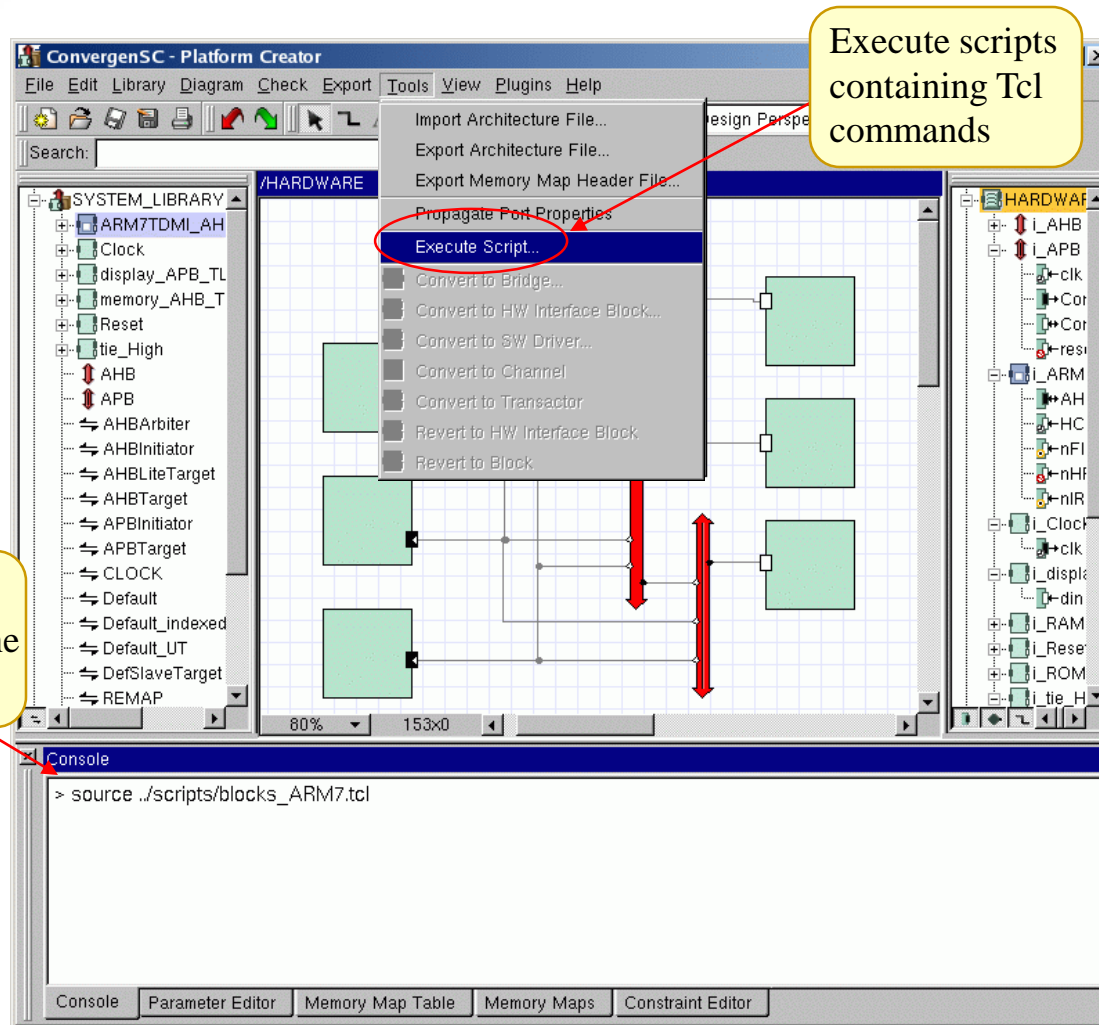
# Easy connect



Auto-links the corresponding transactor(s) for you



A second typical usage is that of scripting. All GUI commands have matching Tcl commands





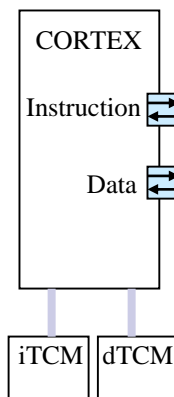
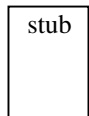
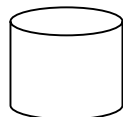
# Platform-based Design

- Architecture build-up (PC)
  - 1. Place Blocks
  - 2. Complete Connections
  - 3. Create Memory Map
  - 4. Set Parameters
- Software programs writing (DS-5)
  - 1. Make Application Software
  - 2. Build image (.axf)
- Run simulation & debug

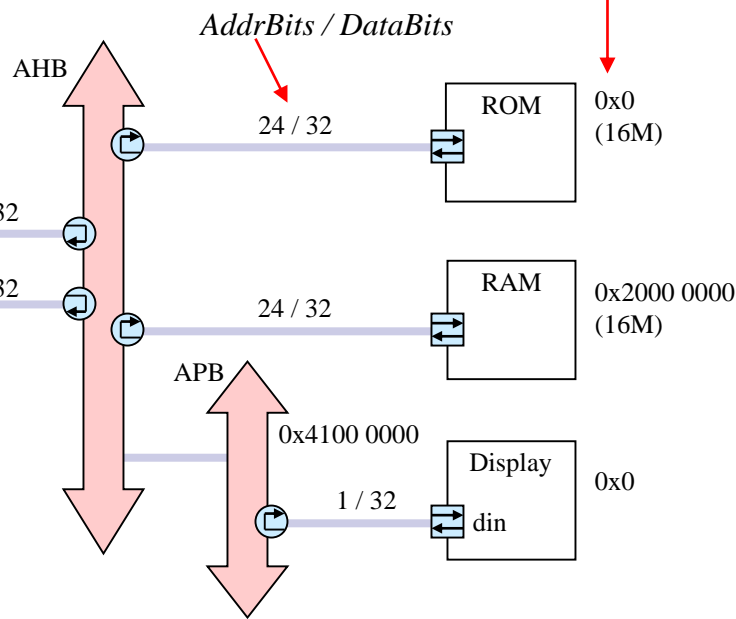
# To illustrate platform-based design, we will build a small system. This is the block diagram

*A short loop that writes to the Display block*

mySoftware.c



Block diagram of platform

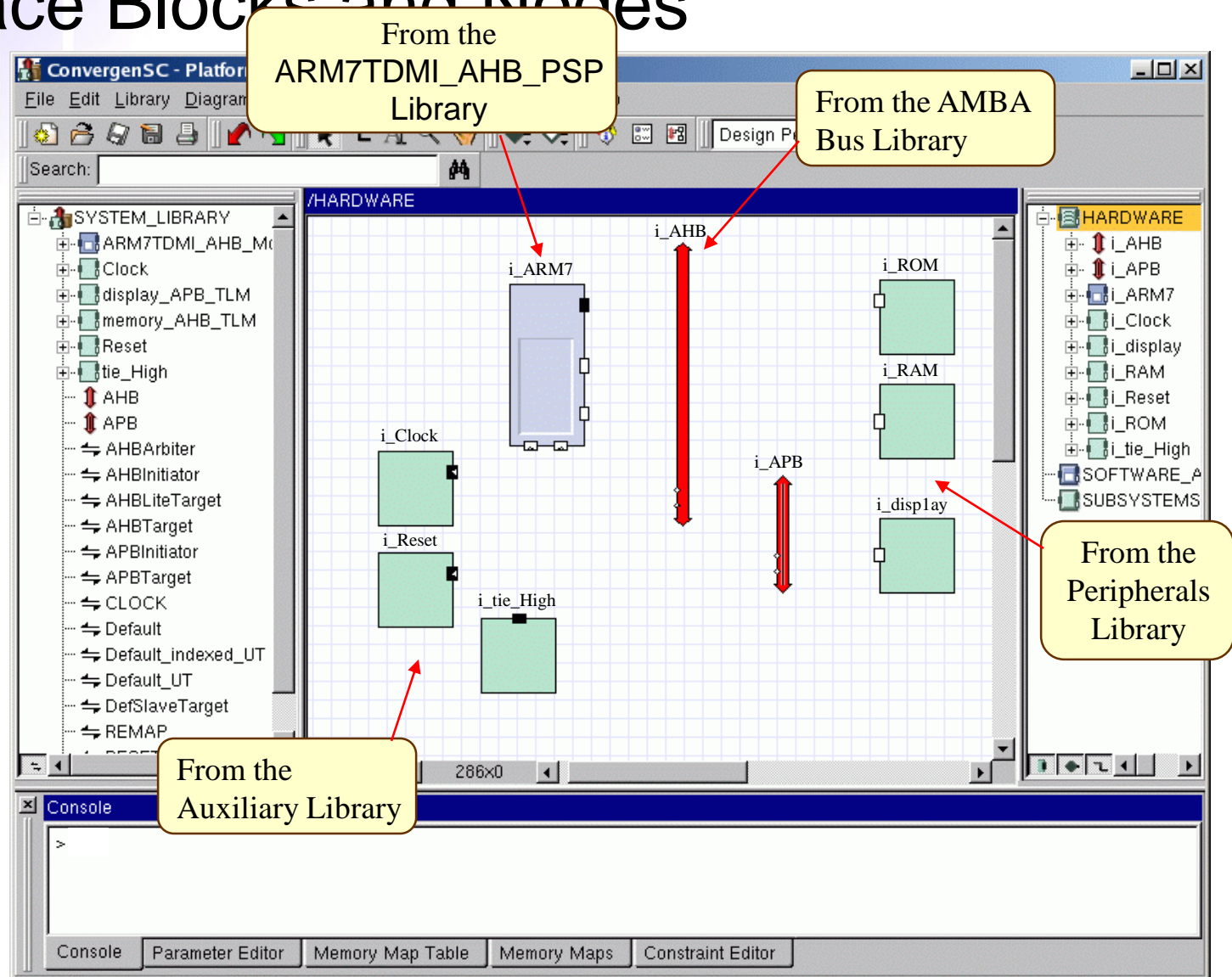


*Memory location (size)*



# ARCHITECTURE BUILD-UP (PC)

# 1. Place Blocks and Nodes

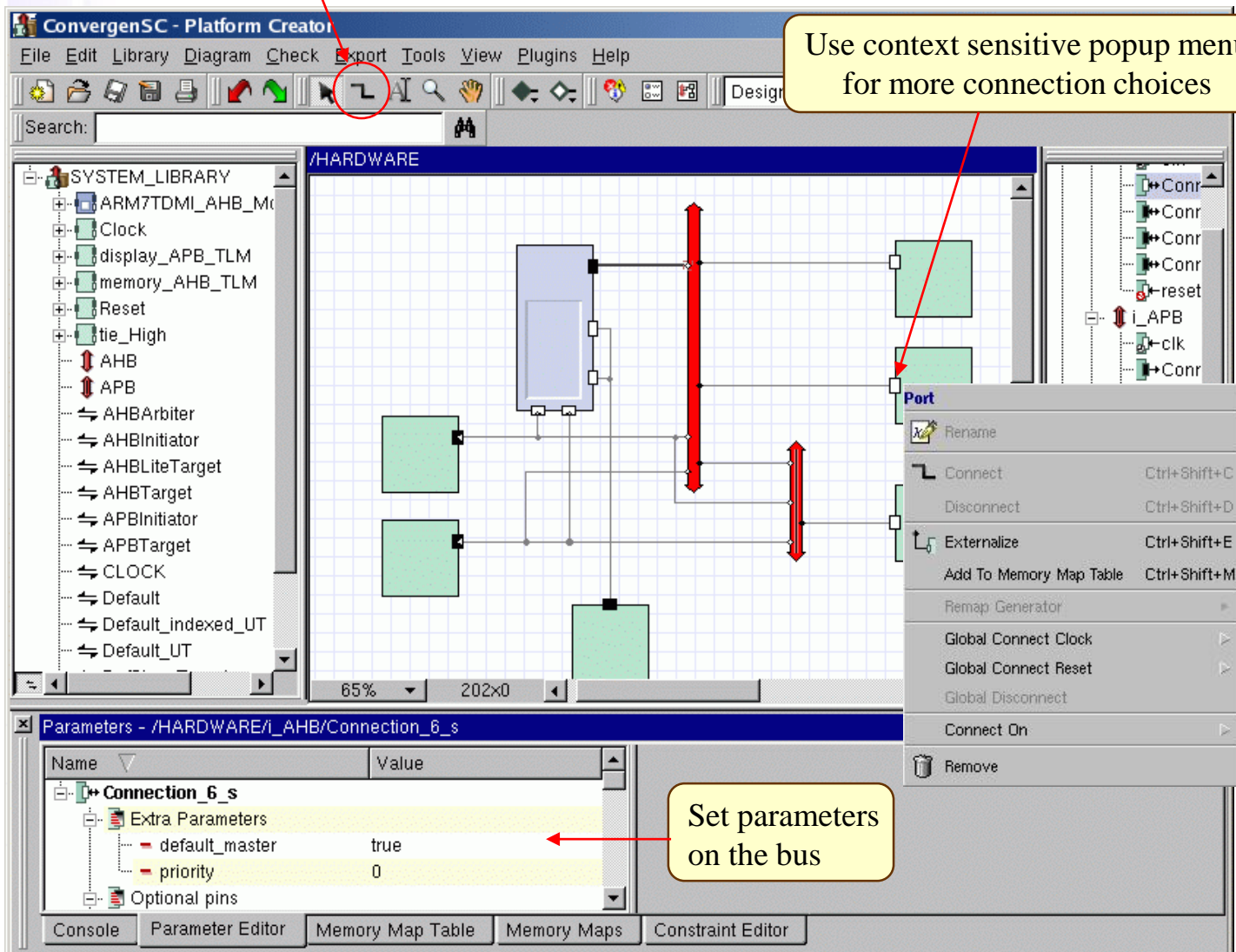




## 2. Complete Connections

Connection tool

Use context sensitive popup menu for more connection choices



The screenshot shows the ConvergenSC - Platform Creator interface. The main workspace displays a hardware diagram with various components connected by buses. A red circle highlights the connection tool in the toolbar. A red arrow points to a context menu for a port, showing options like Connect, Disconnect, Externalize, etc. Another red arrow points to the 'default\_master' parameter in the 'Parameters' panel.

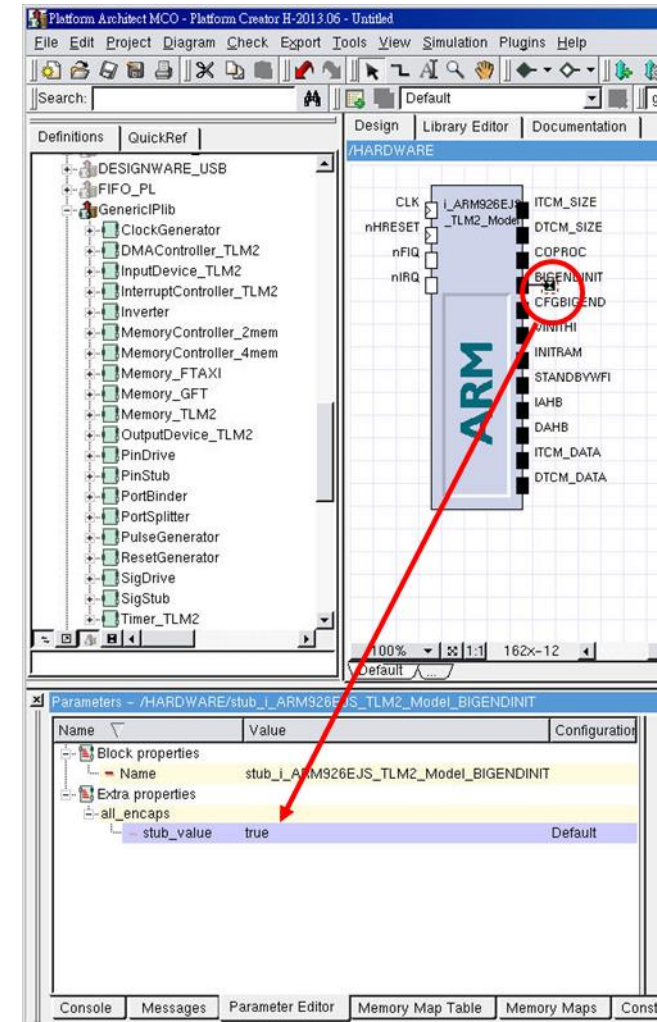
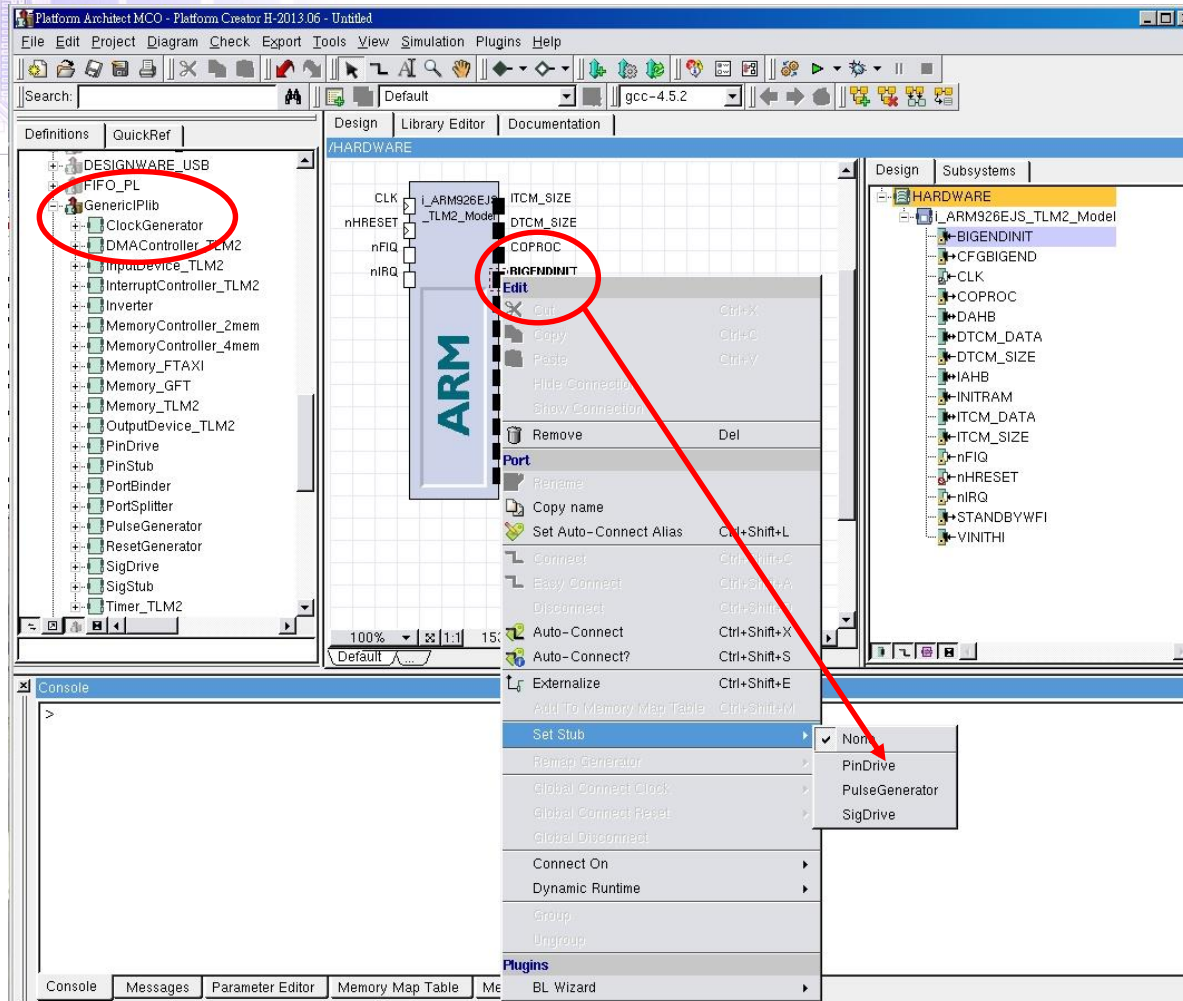
**Parameters - /HARDWARE/\_AHB/Connection\_6\_s**

Name	Value
Connection_6_s	
Extra Parameters	
- default_master	true
- priority	0
Optional pins	



# 2. Complete Connections

## Pin Stub



# 3. Create the Memory Map

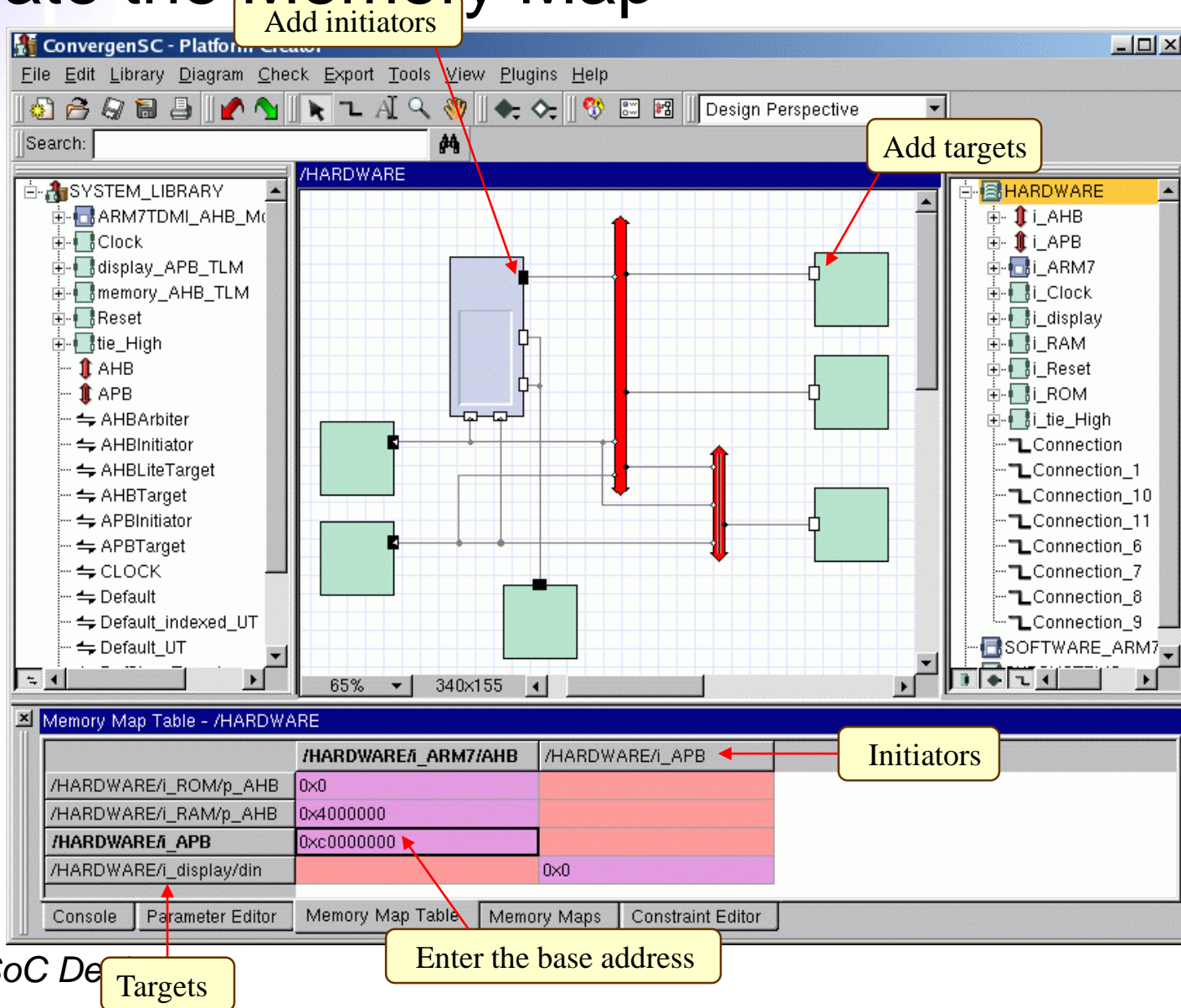
**Add initiators**

**Add targets**

**Initiators**

**Enter the base address**

**Targets**



The screenshot shows the ConvergenSC Platform Creator interface. The main workspace displays a hardware diagram with a central vertical red bar representing a bus, connected to various components. The left sidebar shows the 'SYSTEM\_LIBRARY' with components like ARM7TDMI\_AHB\_Mc, Clock, display\_APB\_TLM, memory\_AHB\_TLM, Reset, tie\_High, AHB, APB, AHBArbiter, AHBInitiator, AHB LiteTarget, AHBTarget, APBInitiator, APBTarget, CLOCK, Default, Default\_indexed\_UT, and Default\_UT. The right sidebar shows the 'HARDWARE' components, including i\_AHB, i\_APB, i\_ARM7, i\_Clock, i\_display, i\_RAM, i\_Reset, i\_ROM, i\_tie\_High, Connection, Connection\_1, Connection\_10, Connection\_11, Connection\_6, Connection\_7, Connection\_8, Connection\_9, and SOFTWARE\_ARM7. The bottom panel shows the 'Memory Map Table - /HARDWARE' with the following data:

	/HARDWARE/i_ARM7/AHB	/HARDWARE/i_APB
/HARDWARE/i_ROM/p_AHB	0x0	
/HARDWARE/i_RAM/p_AHB	0x4000000	
/HARDWARE/i_APB	0xc0000000	
/HARDWARE/i_display/din		0x0



## 4. Set Parameters

The screenshot shows the ConvergenSC - Platform Creator interface. The top menu bar includes File, Edit, Library, Diagram, Check, Export, Tools, View, Plugins, and Help. The left pane shows the SYSTEM\_LIBRARY with various components like ARM7TDMI\_AHB, Clock, display\_APB\_TLM, memory\_AHB\_TL, Reset, tie\_High, AHB, APB, AHBArbiter, AHBInitiator, AHBILiteTarget, AHBTarget, and APBInitiator. The center pane shows a hardware diagram with a central block and several peripheral blocks connected by lines. The right pane shows the instance tree with components like i\_AHB, i\_APB, i\_ARM7, i\_Clock, i\_display, i\_RAM, i\_Reset, i\_ROM, i\_tie\_High, Connection, and Connection\_. A yellow callout box with a red arrow points to the i\_ARM7 instance in the instance tree, containing the text "Select an instance in the diagram or in the instance tree". Below the diagram, a "Parameters - /HARDWARE/i\_ARM7" window is open, showing a table of parameters. A yellow callout box with a red arrow points to the "debugger" parameter value "rvd", containing the text "Click and enter the value".

ConvergenSC - Platform Creator

File Edit Library Diagram Check Export Tools View Plugins Help

Search: [ ]

SYSTEM\_LIBRARY

- ARM7TDMI\_AHB
- Clock
- display\_APB\_TLM
- memory\_AHB\_TL
- Reset
- tie\_High
- AHB
- APB
- AHBArbiter
- AHBInitiator
- AHBILiteTarget
- AHBTarget
- APBInitiator

/HARDWARE

50% 194x530

Parameters - /HARDWARE/i\_ARM7

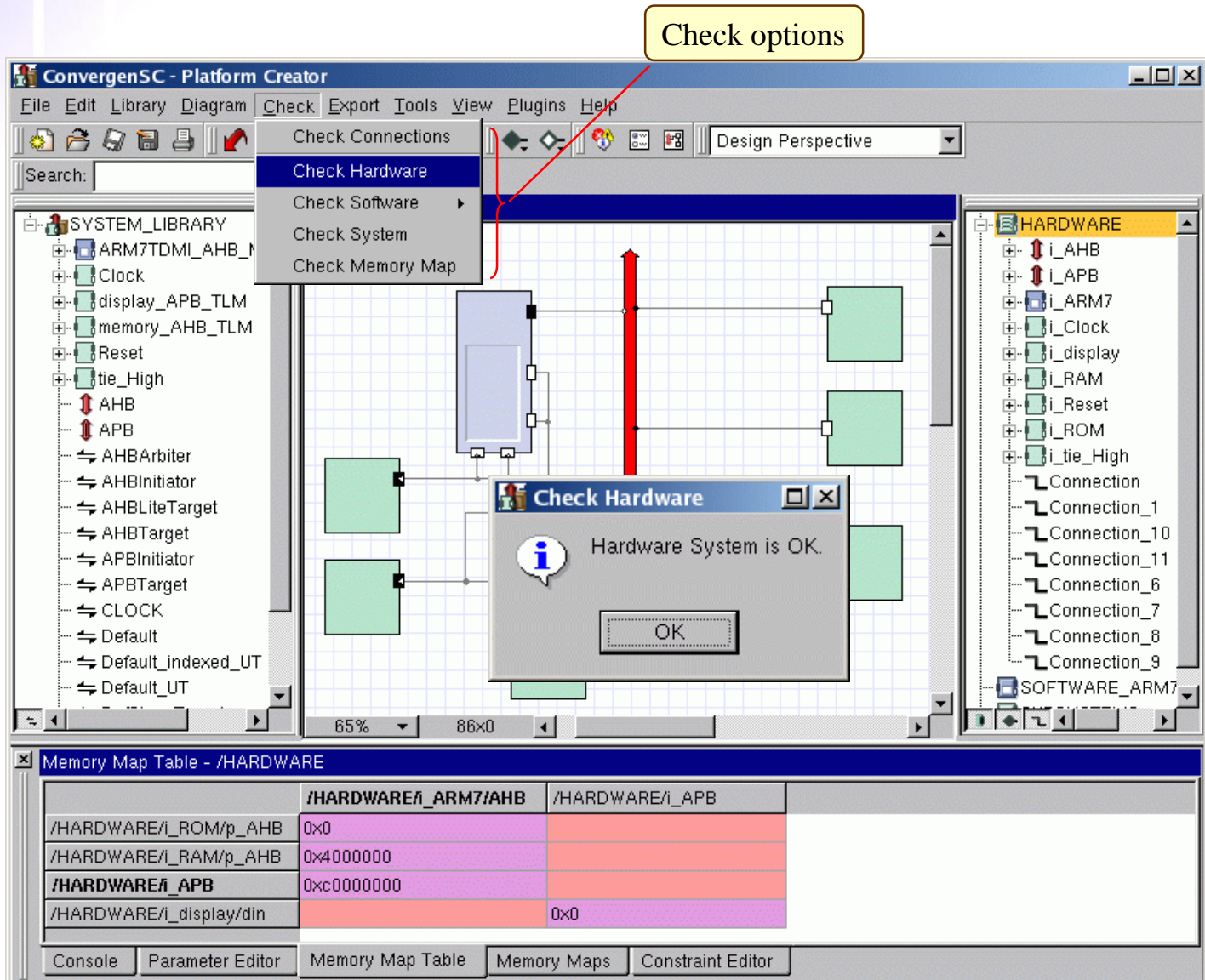
Name	Value
Block properties	
Name	i_ARM7
Constructor Arguments	
debugger	rvd
debuggerArguments	<empty>
debuggerEndian	little
debuggerScript	<empty>

Console Parameter Editor Memory Map Table Memory Maps Constraint Editor



# Check the System

Check options



The screenshot shows the 'ConvergenSC - Platform Creator' interface. The 'Check' menu is open, highlighting 'Check Hardware'. A red arrow points from the 'Check options' label to this menu item. Below the menu, a 'Check Hardware' dialog box displays the message 'Hardware System is OK.' with an 'OK' button. The background shows a system diagram with various components like i\_ARM7, i\_RAM, i\_ROM, and i\_tie\_High. The bottom panel shows the 'Memory Map Table' for the hardware.

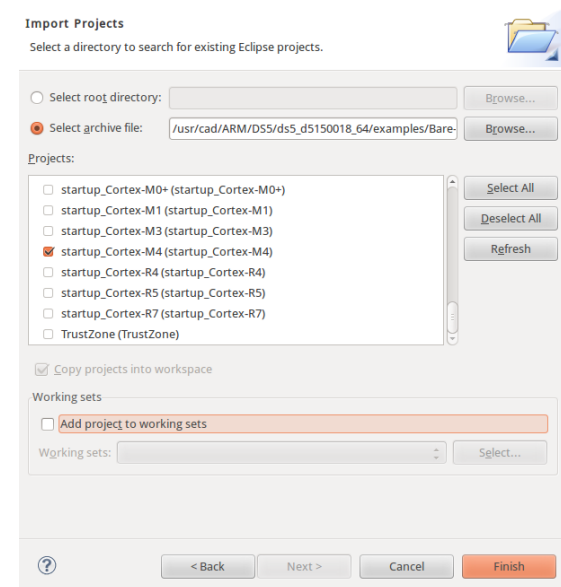
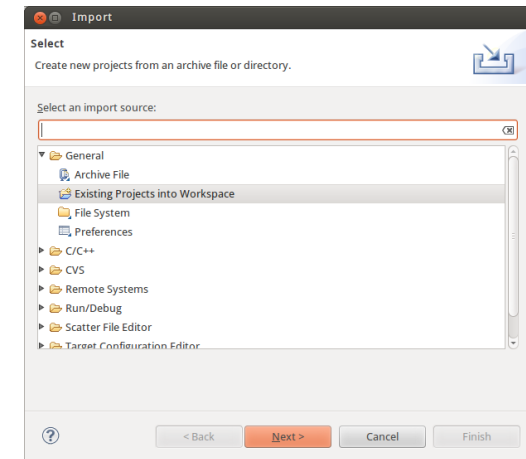
	/HARDWARE/i_ARM7/AHB	/HARDWARE/i_APB
/HARDWARE/i_ROM/p_AHB	0x0	
/HARDWARE/i_RAM/p_AHB	0x4000000	
<b>/HARDWARE/i_APB</b>	<b>0xc0000000</b>	
/HARDWARE/i_display/din		0x0



# SOFTWARE PROGRAMS WRITING (DS-5)

# Import your code from DS-5

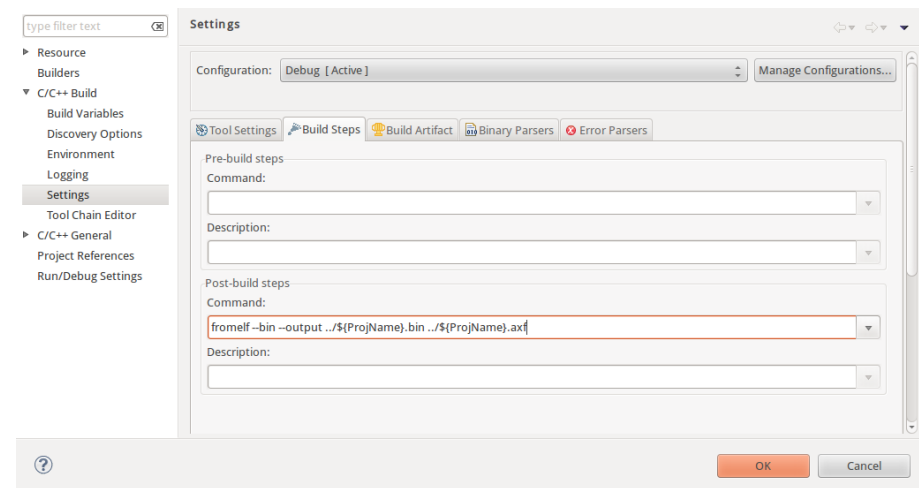
- File -> Import
  - Existing Project into Workspace
- Check “Select archive file” and Browse to find your file
- Or you can import the sample code
  - /usr/cad/ARM/DS5/cur/examples/Bare-metal\_examples.zip
  - Press “Deselect All” button
  - Select startup\_Cortex-M4 (startup\_Cortex-M4) project





# Setting and generating application images

- Set linker to generate binary
  - Project -> Properties -> C/C++ Build -> Settings
  - In Tab “Build Steps” -> Post-build steps -> Command  
fromelf --bin --output ../\${ProjName}.bin ../\${ProjName}.axf
- Generate execution file and binary
  - Project -> Build project
  - Files will be generated  
Startup\_Cortex-M4.axf  
Startup\_Cortex-M4.bin
- Use this as template to modify main() function and build your own application image.



# The application software is a simple routine for our testing purposes



## mySoftware.c

*Unused Interrupts*

**User Function**

**Main**

```
#include "defines.h"
extern void InterruptEnable(void);

extern __irq void p_nIRQ_ISR (void) { ; }
extern __irq void p_nFIQ_ISR (void) { ; }

void myFunction(void) {
    int i;
    for (i=0; i < 20; i++){
        myDevice = i;
    }
    return;
}

int main(void) {
    InterruptEnable();
    myFunction();
    return 0;
}
```

*Include header file  
Prototype for InterruptEnable()*

*Stub off the ISR's*

*Write some integers out to  
the peripheral*

## defines.h

```
#define myDevice (*(volatile unsigned int *)0xc0000000)
```



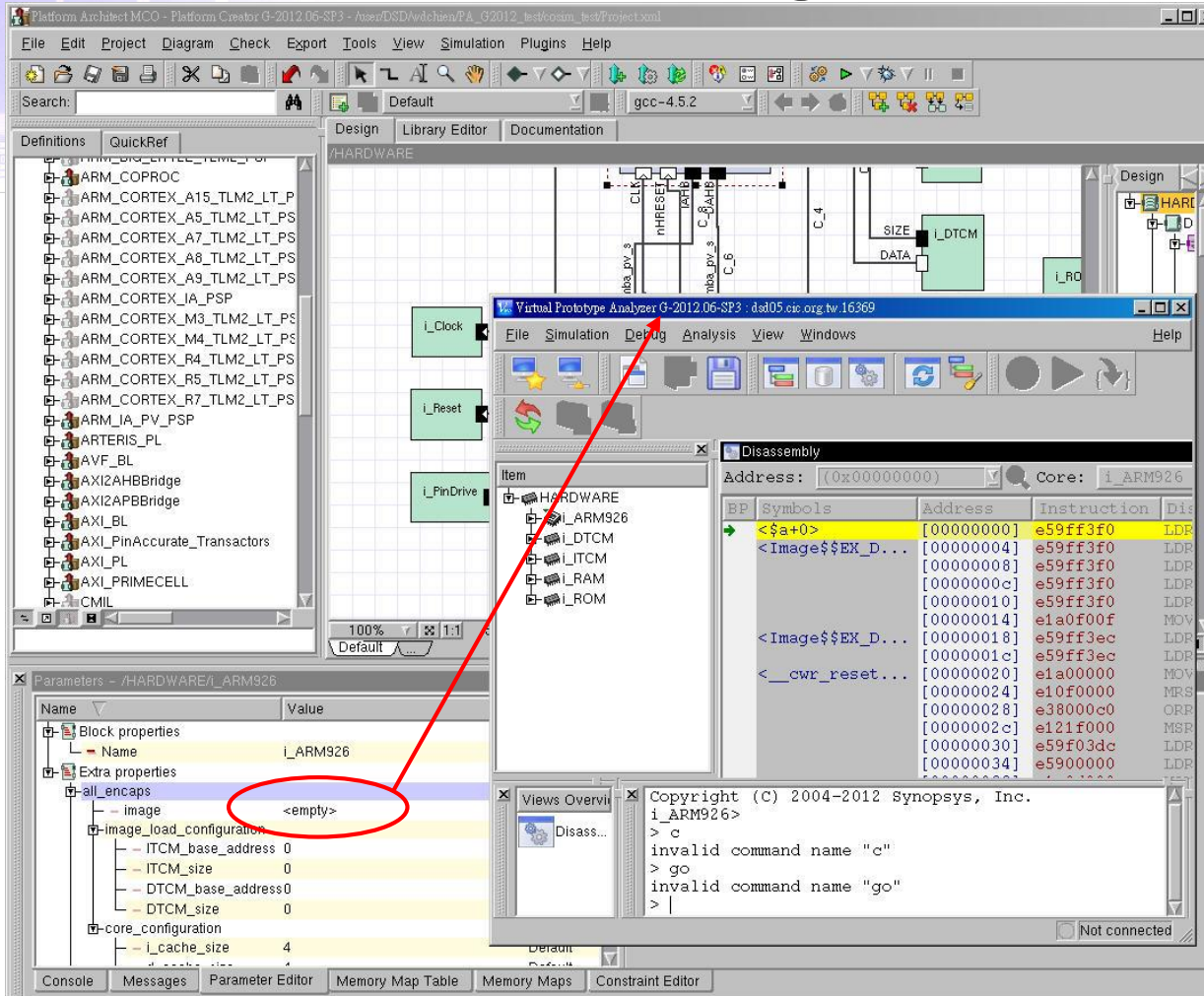
# Addressing

```
1 #define ALU_BASE      0x40001000|
2 #define DISPLAY_BASE  0x41000000
3 #define DUAL_TIMER_BASE 0x41001000
4
5 #define OPCODE         (*((volatile unsigned int *) (ALU_BASE)))
6 #define OPERAND1       (*((volatile unsigned int *) (ALU_BASE+0x04)))
7 #define OPERAND2       (*((volatile unsigned int *) (ALU_BASE+0x08)))
8 #define RESULT         (*((volatile unsigned int *) (ALU_BASE+0x0C)))
9
10 #define DISPLAY        (*((volatile unsigned int *) (DISPLAY_BASE)))
11
12 #define TIMER1LOAD     (*((volatile unsigned int *) (DUAL_TIMER_BASE)))
13 #define TIMER1VALUE    (*((volatile unsigned int *) (DUAL_TIMER_BASE+0x04)))
14 #define TIMER1CONTROL  (*((volatile unsigned int *) (DUAL_TIMER_BASE+0x08)))
15 #define TIMER1INTCLR   (*((volatile unsigned int *) (DUAL_TIMER_BASE+0x0C)))
16 #define TIMER1RIS      (*((volatile unsigned int *) (DUAL_TIMER_BASE+0x10)))
17 #define TIMER1MIS      (*((volatile unsigned int *) (DUAL_TIMER_BASE+0x14)))
18 #define TIMER1BGLOAD   (*((volatile unsigned int *) (DUAL_TIMER_BASE+0x18)))
```



# RUN SIMULATION & DEBUG

# Software debug



The screenshot displays the Platform Architect MCO - Platform Creator G-2012.06-SP3 interface. The main window shows a hardware design with components like i\_Clock, i\_Reset, and i\_PinDrive. A red arrow points from the 'Parameters' window to the 'Virtual Prototype Analyzer' (VPA) window.

**Parameters - /HARDWARE/i\_ARM926**

Name	Value
Block properties	
Name	i_ARM926
Extra properties	
all_encaps	
image	
image_load_configuration	
ITCM_base_address	0
ITCM_size	0
DTCM_base_address	0
DTCM_size	0
core_configuration	
i_cache_size	4

**Virtual Prototype Analyzer G-2012.06-SP3**

File Simulation Debug Analysis View Windows Help

Disassembly

Item: i\_ARM926

Address: (0x00000000) Core: i\_ARM926

BP	Symbols	Address	Instruction	Dis
	<\$a+0>	[00000000]	e59ff3f0	LDR
	<Image\$\$EX_D...	[00000004]	e59ff3f0	LDR
		[00000008]	e59ff3f0	LDR
		[0000000c]	e59ff3f0	LDR
		[00000010]	e59ff3f0	LDR
		[00000014]	e1a0f00f	MOV
	<Image\$\$EX_D...	[00000018]	e59ff3ec	LDR
		[0000001c]	e59ff3ec	LDR
	<_cwr_reset...	[00000020]	e1a00000	MOV
		[00000024]	e10f0000	MRS
		[00000028]	e38000c0	ORR
		[0000002c]	e121f000	MSR
		[00000030]	e59f03dc	LDR
		[00000034]	e5900000	LDR

Views Overview

Copyright (C) 2004-2012 Synopsys, Inc.

i\_ARM926>

```
> c
invalid command name "c"
> go
invalid command name "go"
> |
```

Not connected

Virtual Platform Analyzer  
(VPA): PA build-in software  
debugger / analyzer



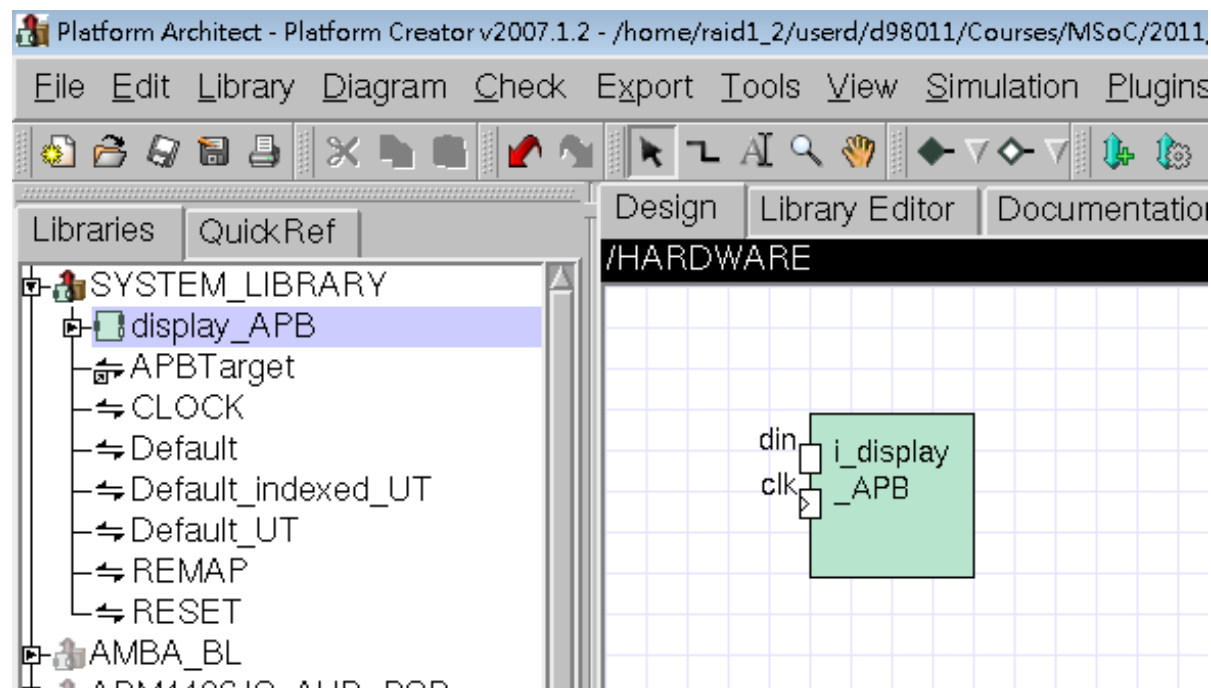
# Lab Exercise

## Platform Architect (I)



# Target

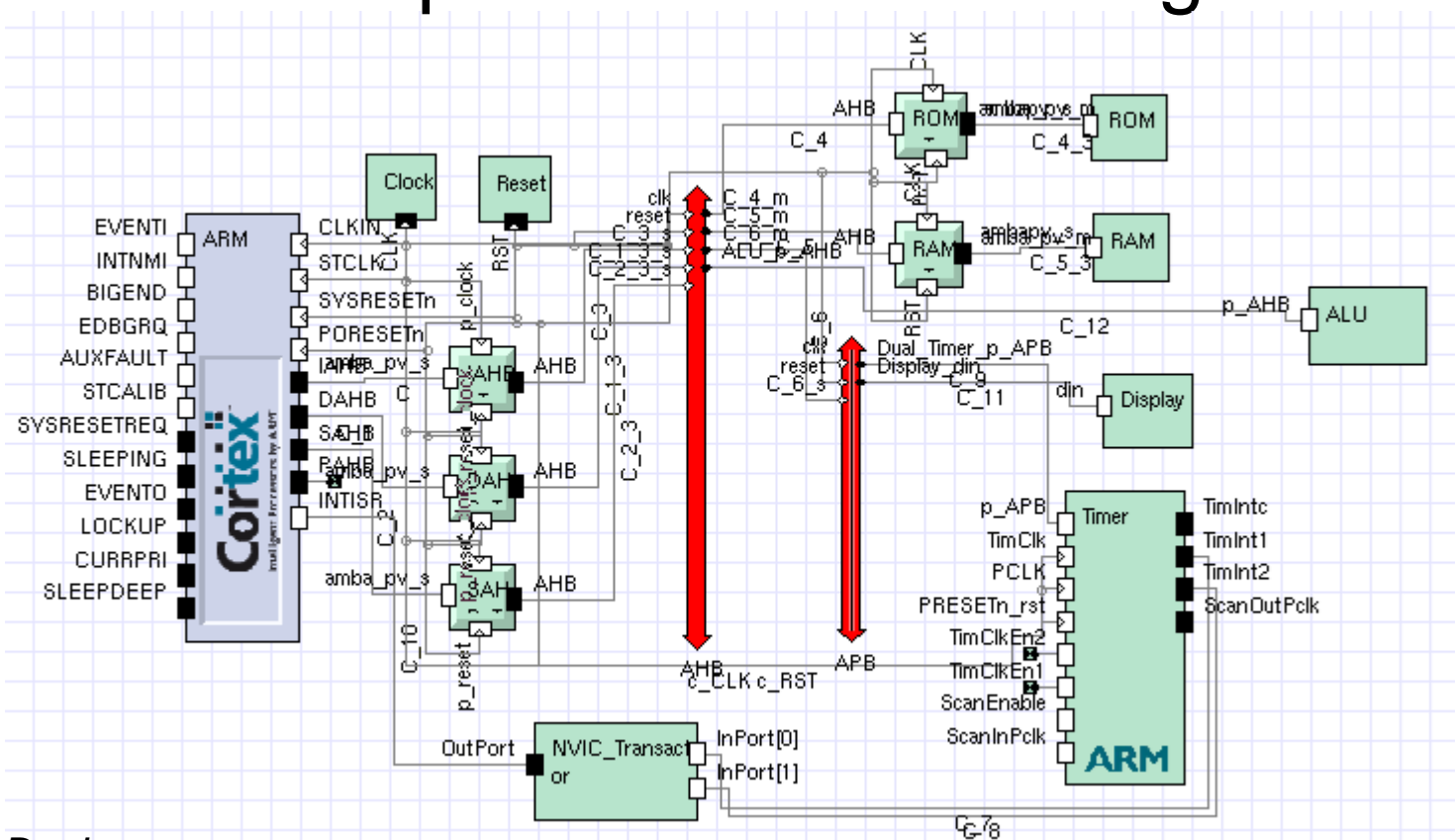
- Building a self-developed SystemC and Verilog module into a library





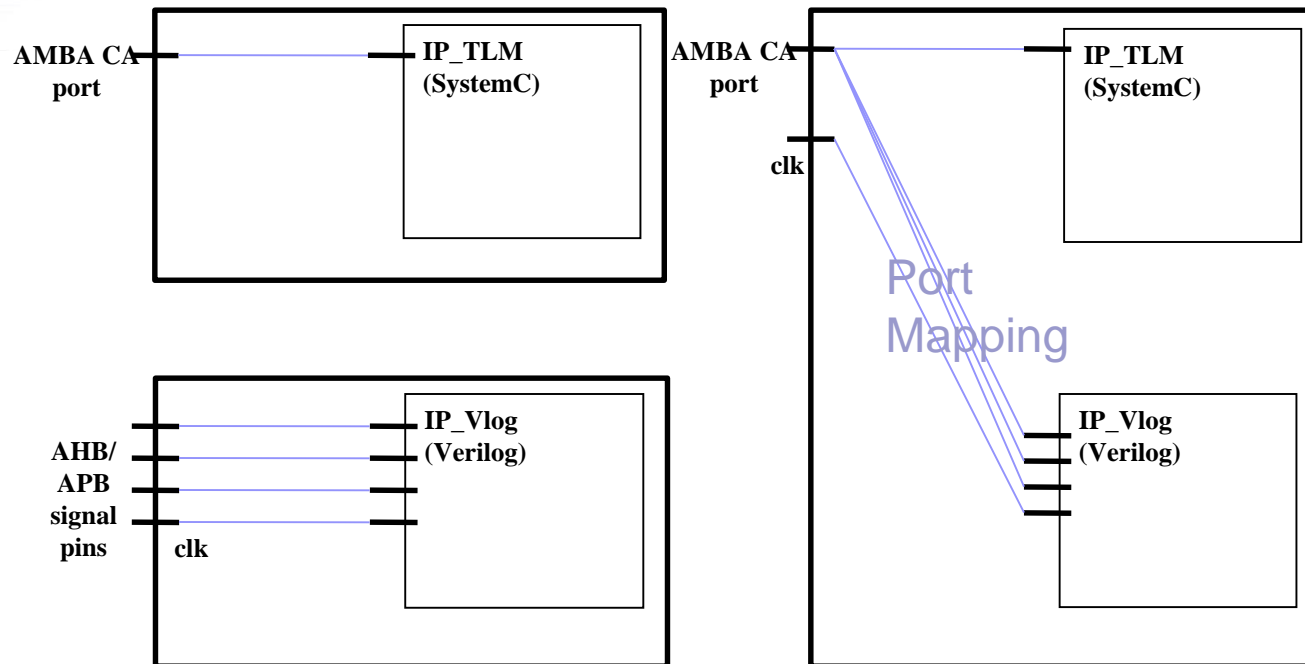
# Target

- Familiar with the platform-based design flow





# Packing Verilog module



# Commands

## ■ 解壓縮

- ☐ >tar -xvf PA\_DS5\_ALU\_v1.05.tar.gz

## ■ Source File

- ☐ >source pa\_setup.csh

- ☐ >source run.csh

Modify it :

#31 ARM -> arm

#32 ARM -> arm

run.csh #6 ARM->arm

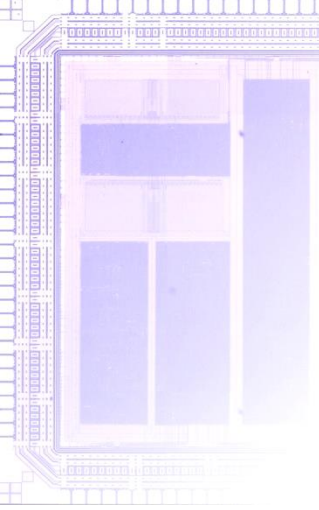
## ■ Cad lab server

- ☐ [http://cad.ee.ntu.edu.tw/ws\\_list.htm](http://cad.ee.ntu.edu.tw/ws_list.htm)

- ☐ Linux OS

Use cad42 !!

## ■ Remember to execute X-ming or X-win



IP	Name	Type	CPU	CPU Clock	Memory	OS
140.112.20.59	cad16	IBM X3400	Intel Xeon 64	2.4 GHz * 16	100 G	RHEL 5
140.112.20.60	cad17	IBM X3550	Intel Xeon 64	2.4 GHz * 16	20 G	RHEL 5
140.112.20.61	cad18	SUN Blade 2500	UltraSPARC	1.28 GHz*2	8 G	Solaris 10
140.112.20.62	cad19	SUN Blade 2000	UltraSPARC	1.2 GHz * 2	8 G	Solaris 10
140.112.20.63	cad20	SUN Blade 2000	UltraSPARC	1.2 GHz * 2	8 G	Solaris 10
140.112.20.64	cad21	DELL R620	Intel Xeon 64	2 GHz * 32	48 G	CentOS 5
140.112.20.65	cad22	SUN Blade 2500	UltraSPARC	1.28 GHz * 2	8 G	Solaris 10
140.112.20.66	cad23	DELL R620	Intel Xeon 64	2 GHz * 16	96 G	CentOS 5
140.112.20.67	cad24	SUN Fire 280R	UltraSPARC	1.2 GHz * 2	4 G	Solaris 9
140.112.20.68	cad25	SUN Fire 280R	UltraSPARC	1.2 GHz * 2	4 G	Solaris 9
140.112.20.69	cad26	SUN Fire 280R	UltraSPARC	1.2 GHz * 2	4 G	Solaris 9
140.112.20.70	cad27	IBM x260	Intel Xeon 64	3.2 GHz * 4	8 G	RHEL 4
140.112.20.71	cad28	IBM x260	Intel Xeon 64	3.2 GHz * 4	8 G	RHEL 4
140.112.20.72	cad29	IBM e336	Intel Xeon 64	3.2 GHz	5 G	RHEL 4
140.112.20.73	cad30	IBM X3650	Intel Xeon 64	2 GHz * 16	12 G	SUSE 11
140.112.20.74	cad31	DELL R620	Intel Xeon 64	2.2 GHz * 32	48 G	CentOS 6
140.112.20.75	cad32	SUN Blade 2000	UltraSPARC	1.015 GHz * 2	8 G	Solaris 8
140.112.20.76	cad33	IBM X3500	Intel Xeon 64	2 GHz * 16	20 G	RHEL 4
140.112.20.77	cad34	IBM X3650	Intel Xeon 64	2.4 GHz * 8	12 G	RHEL 5
140.112.20.78	cad35	SUN V20z	AMD Opteron	2.4 GHz * 2	4 G	RHEL 4
140.112.20.79	cad36	SUN V20z	AMD Opteron	2.4 GHz * 2	4 G	RHEL 4
140.112.20.80	cad37	SUN V20z	AMD Opteron	2.4 GHz * 2	4 G	RHEL 4
140.112.20.81	cad38	IBM X3650	Intel Xeon 64	2.4 GHz * 16	96 G	RHEL 5
140.112.20.82	cad39	IBM X3650	Intel Xeon 64	2 GHz * 24	40 G	CentOS 5
140.112.20.83	cad40	IBM X3550	Intel Xeon 64	2 GHz * 24	8 G	CentOS 5
140.112.20.84	cad41	IBM X3550	Intel Xeon 64	2.1 GHz * 24	64 G	CentOS 6
140.112.20.85	cad42	IBM X3500	Intel Xeon 64	2 GHz * 24	32 G	CentOS 5