Sundance SMT8096
PCI based DSP/FPGA development board

2009.12.18.

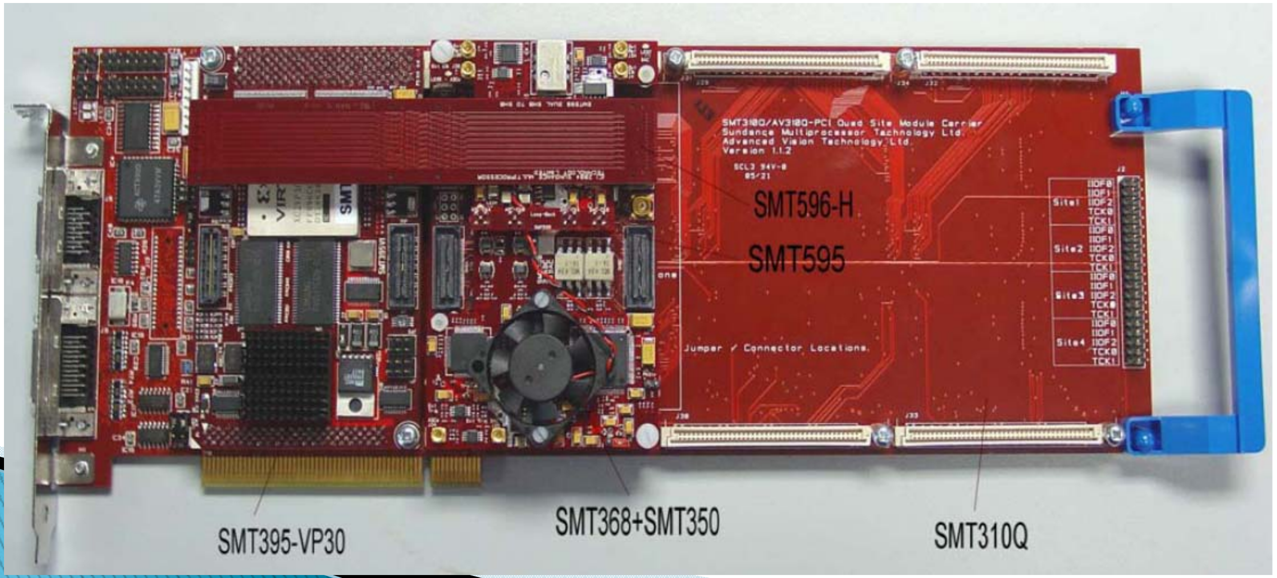
Kang Young Yun
CISL, POSTECH

Outline

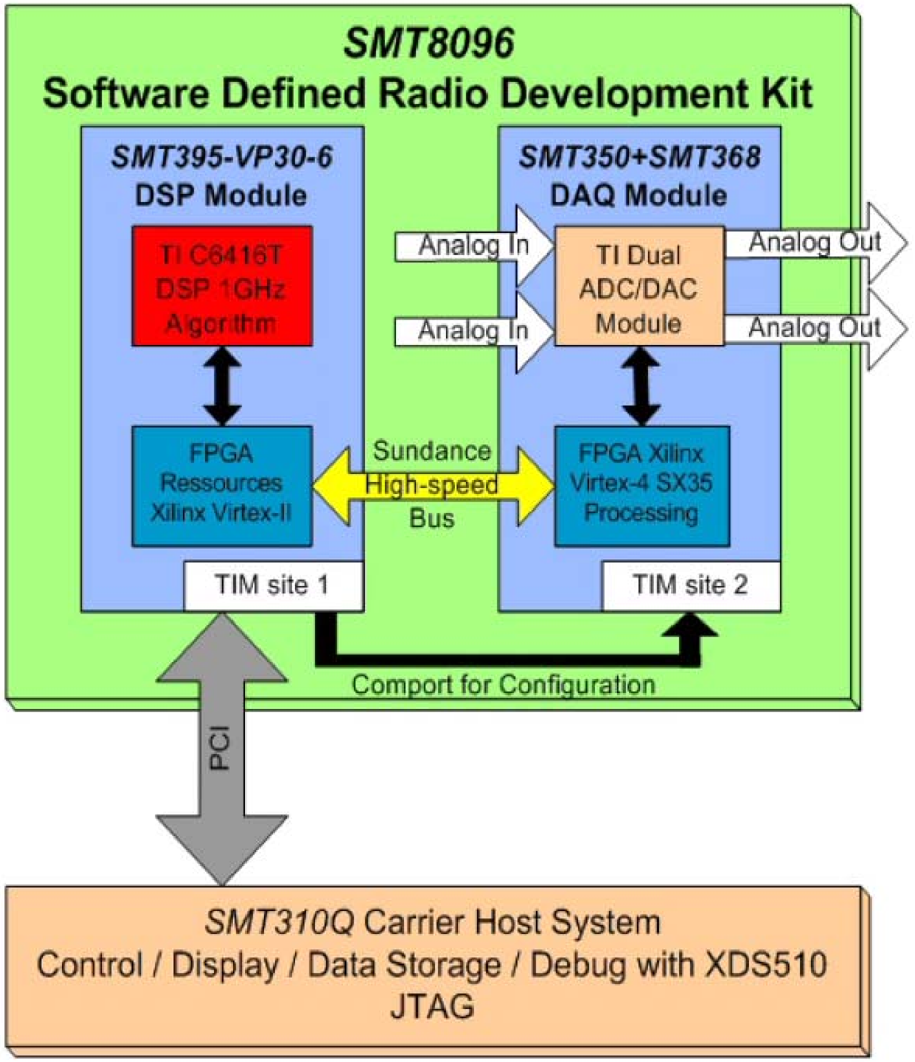
1. Introduction
    1.1 SMT8096
    1.2 DSP/FPGA
2. Hardware
    2.1 DSP module
    2.2 FPGA module
    2.3 ADC/DAC module
3. Software
    3.1 Basic setup
    3.2 3L Diamond
    3.3 Xilinx ISE
    3.4 Modelsim
    3.5 PARS
4. Examples
    4.1 DSP example
    4.2 FPGA example
    4.3 IDE example
    4.4 Xilinx ISE example
    4.5 Modelsim example
    4.6 PARS example

Introduction – SMT8096

- The SMT8096 is a PCI system based on 3 main modules.
  - TI C6416T DSP module (SMT395)
  - Xilinx Virtex-4 FPGA module (SMT368)
  - Dual ADC/DAC module (SMT350)
- All plugged on a PCI carrier board (SMT310Q).



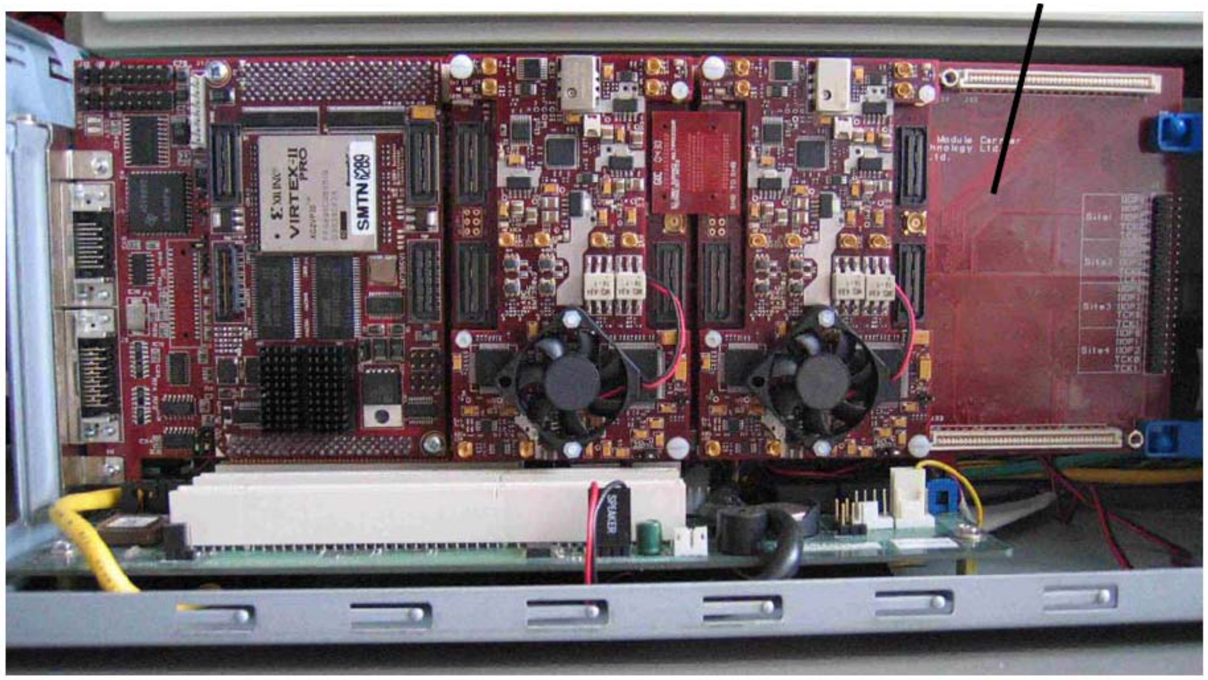
Introduction – SMT8096 (cont.)



- SHB (Sundance High-speed Bus)
  - 32bit, 100MB/s
- Comport (Communication port)
  - 8bit, 20MB/s

Introduction – SMT8096 (cont.)

- 보유 장비
  - 1) SMT8096 board (2X2 시스템)
  - 2) SMT8096 baord + FPGA mod. + ADC/DAC mod.
    (4X4 시스템으로 업그레이드)



Introduction – DSP/FPGA

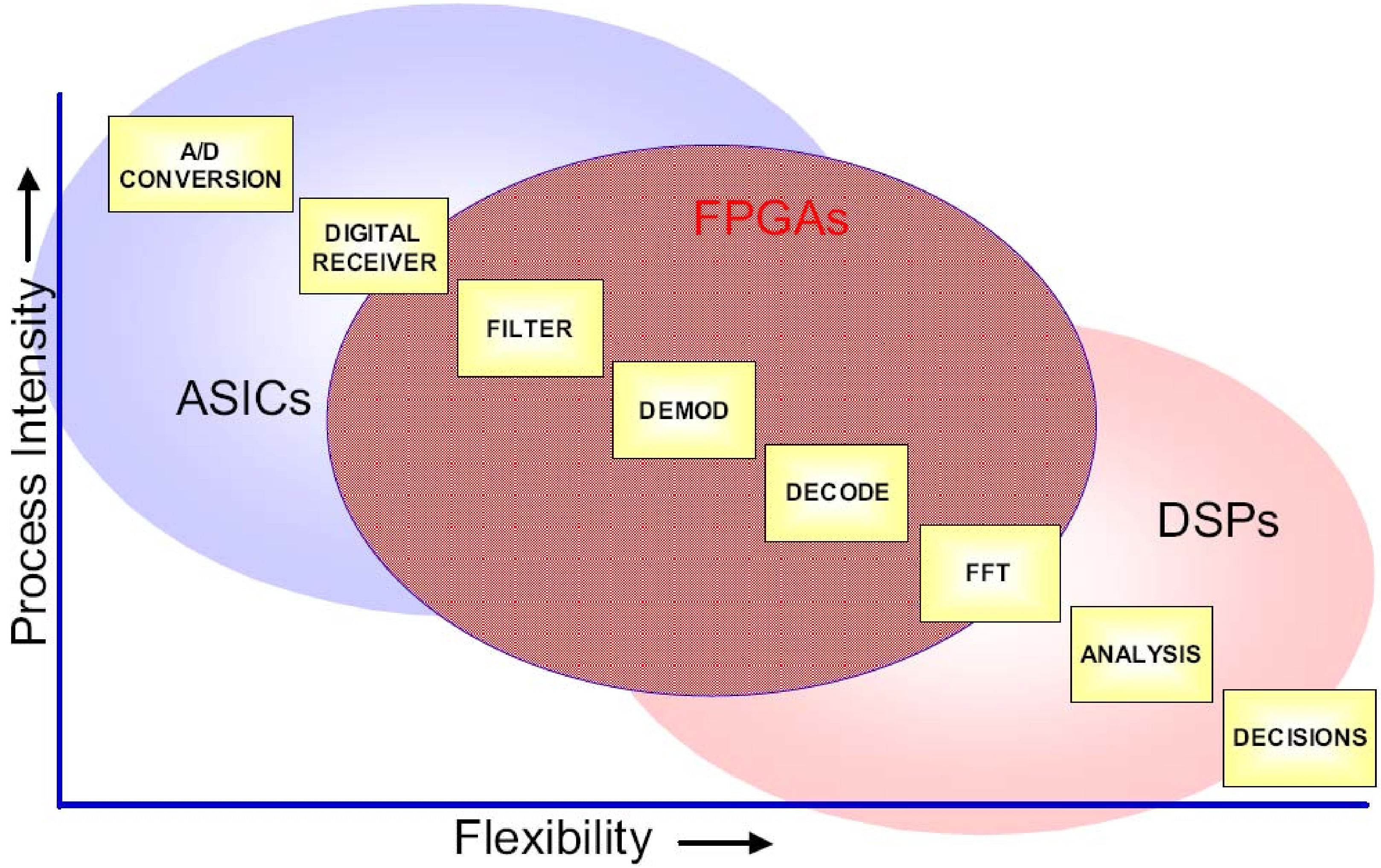
- Digital Signal Processor
  - 프로그램에 의해 명령이 하나씩 patch되어 해석되어 동작함
  - 고정된 하드웨어 아키텍처에 의한 제약
  - 제조사: Texas Instrument, Analog Device

- Field Programmable Gate Array
  - 명령의 해석없이 하드웨어가 직접 동작함
  - 맞춤화된 아키텍처, 버스 구조, 메모리 및 가속기 블록
  - 제조사: Xilinx, Altera, Atmel

Introduction – DSP/FPGA (cont.)

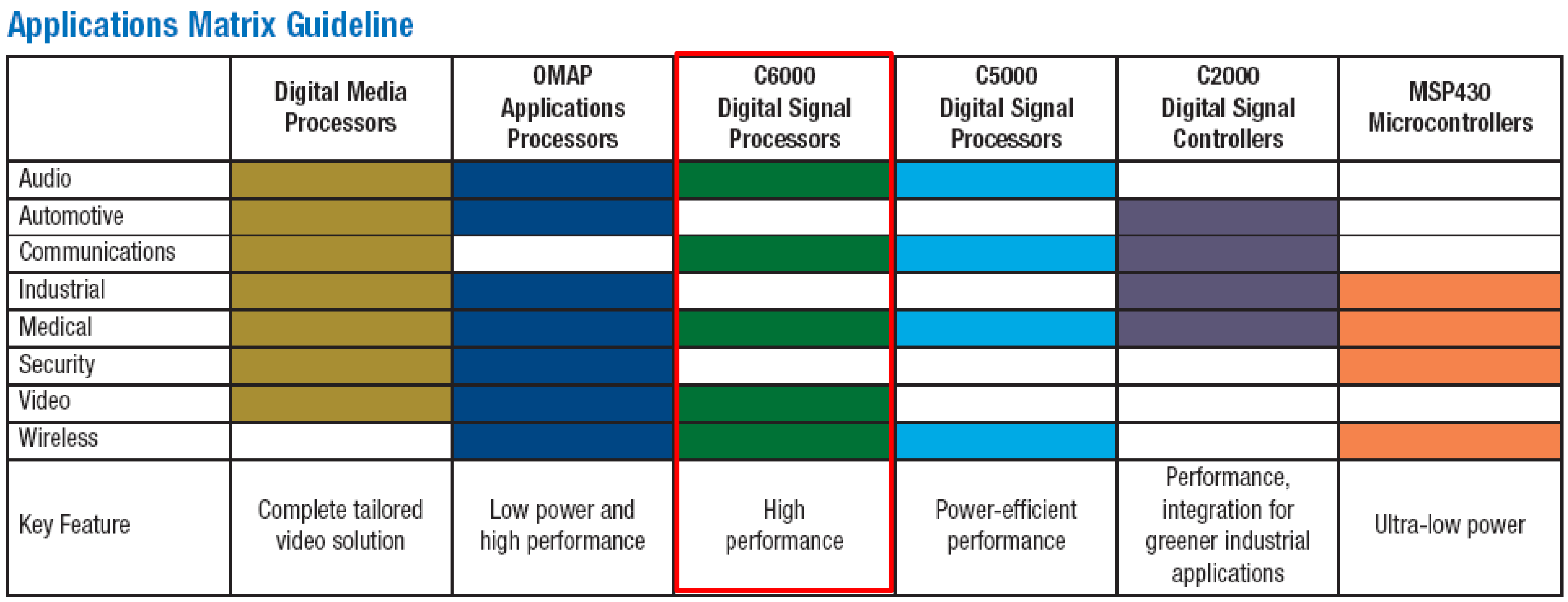
- Reasons to select an FPGA over a DSP microprocessor
  - Performance target not achievable with one–two microproce ssors
    - Properly executed FPGA designs typically outperform a DSP micr oprocessor by a factor of 100:1, and by more than 1000:1 in sp ecial circumstances
  - Power dissipation
    - Power dissipation of an FPGA–DSP design is typically about 20% of a microprocessor based design working at the same sample r ate
  - Programmatic issues can tip the balance
    - Software validation costs are avoided by using hardware
    - Availability of talent/tools
    - Reliability Issues

Introduction – DSP/FPGA (cont.)



Introduction – DSP/FPGA (cont.)

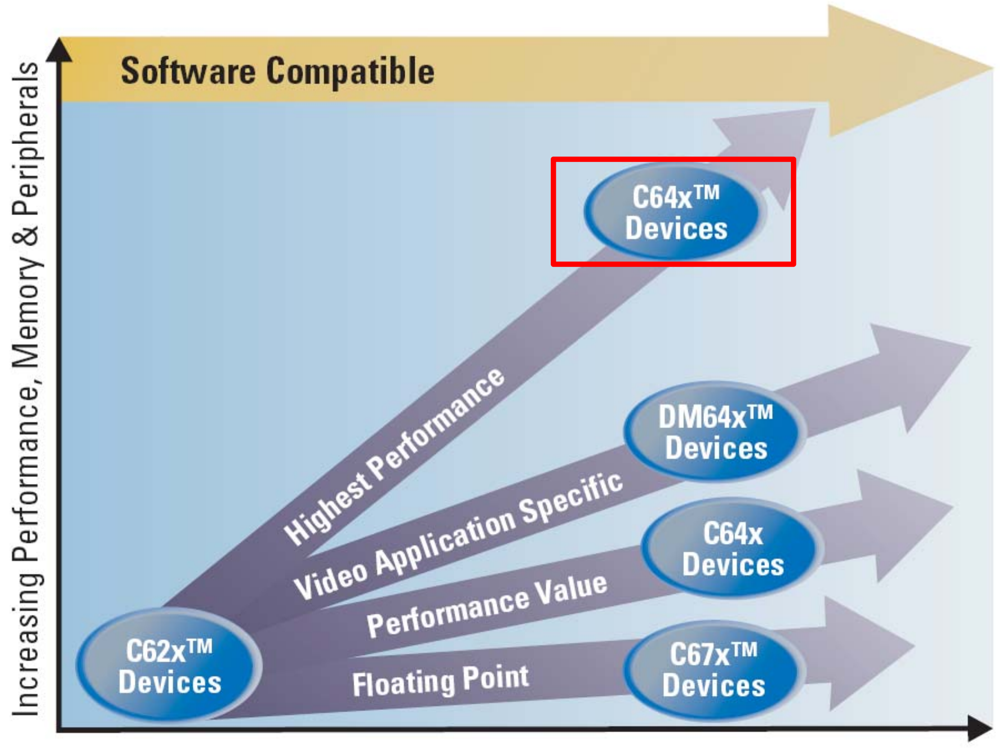
- TI DSPs



Applications Matrix Guideline

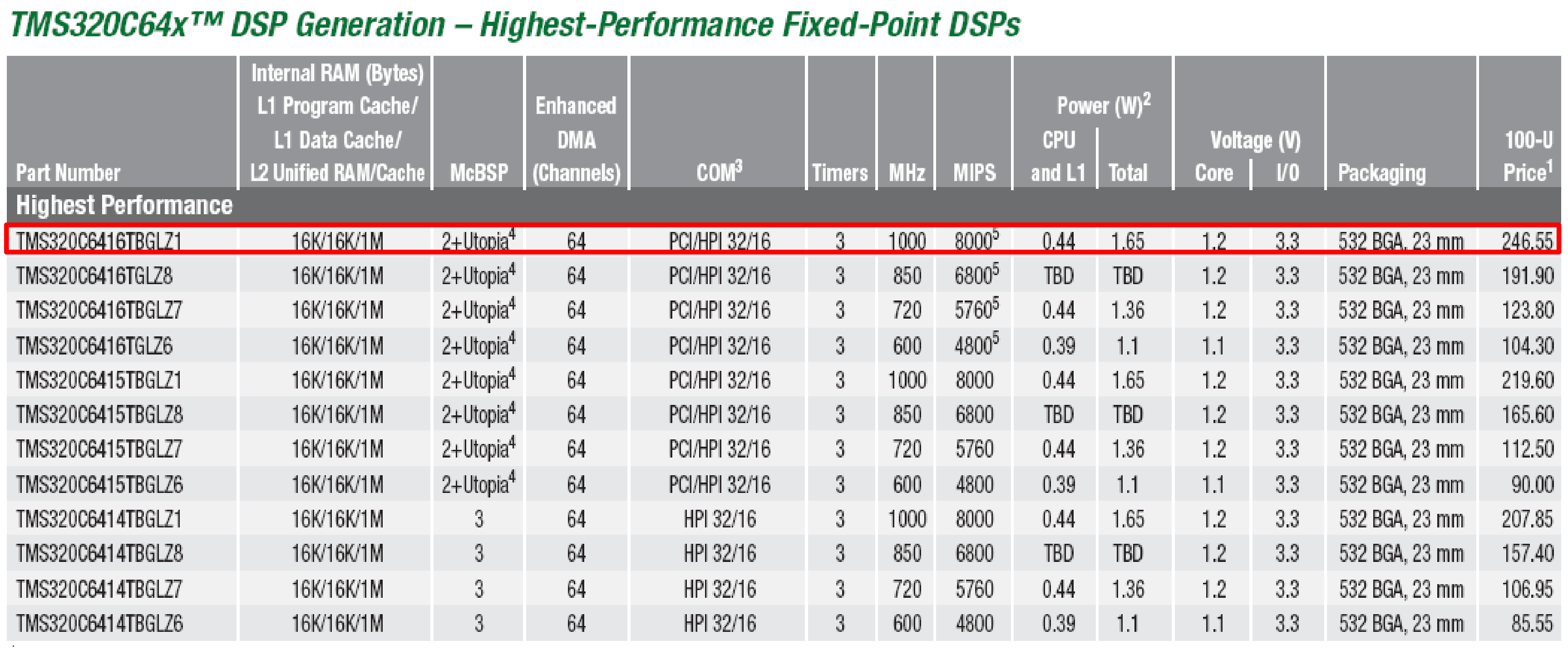
	Digital Media Processors	OMAP Applications Processors	C6000 Digital Signal Processors	C5000 Digital Signal Processors	C2000 Digital Signal Controllers	MSP430 Microcontrollers
Audio	■	■	■	■		
Automotive	■	■			■	
Communications	■		■	■	■	
Industrial	■	■			■	■
Medical	■	■	■	■	■	■
Security	■	■				■
Video	■	■	■			
Wireless		■	■	■		■
Key Feature	Complete tailored video solution	Low power and high performance	High performance	Power-efficient performance	Performance, integration for greener industrial applications	Ultra-low power

Introduction – DSP/FPGA (cont.)



C6000™ DSP Platform Roadmap
The C6000 DSP platform includes a wide range of devices that raise the bar in performance, set new levels of cost efficiency and offer on-chip peripheral integration to enable developers of high-performance systems to choose the device that best suits their specific application.

Introduction – DSP/FPGA (cont.)



TMS320C64x™ DSP Generation – Highest-Performance Fixed-Point DSPs

Part Number	Internal RAM (Bytes) L1 Program Cache/ L1 Data Cache/ L2 Unified RAM/Cache	McBSP	Enhanced DMA (Channels)	COM[3]	Timers	MHz	MIPS	Power (W)[2] CPU and L1	Power (W)[2] Total	Voltage (V) Core	Voltage (V) I/O	Packaging	100-U Price[1]
Highest Performance													
TMS320C6416TBGLZ1	16K/16K/1M	2+Utopia[4]	64	PCI/HPI 32/16	3	1000	8000[5]	0.44	1.65	1.2	3.3	532 BGA, 23 mm	246.55
TMS320C6416TGLZ8	16K/16K/1M	2+Utopia[4]	64	PCI/HPI 32/16	3	850	6800[5]	TBD	TBD	1.2	3.3	532 BGA, 23 mm	191.90
TMS320C6416TBGLZ7	16K/16K/1M	2+Utopia[4]	64	PCI/HPI 32/16	3	720	5760[5]	0.44	1.36	1.2	3.3	532 BGA, 23 mm	123.80
TMS320C6416TGLZ6	16K/16K/1M	2+Utopia[4]	64	PCI/HPI 32/16	3	600	4800[5]	0.39	1.1	1.1	3.3	532 BGA, 23 mm	104.30
TMS320C6415TBGLZ1	16K/16K/1M	2+Utopia[4]	64	PCI/HPI 32/16	3	1000	8000	0.44	1.65	1.2	3.3	532 BGA, 23 mm	219.60
TMS320C6415TBGLZ8	16K/16K/1M	2+Utopia[4]	64	PCI/HPI 32/16	3	850	6800	TBD	TBD	1.2	3.3	532 BGA, 23 mm	165.60
TMS320C6415TBGLZ7	16K/16K/1M	2+Utopia[4]	64	PCI/HPI 32/16	3	720	5760	0.44	1.36	1.2	3.3	532 BGA, 23 mm	112.50
TMS320C6415TBGLZ6	16K/16K/1M	2+Utopia[4]	64	PCI/HPI 32/16	3	600	4800	0.39	1.1	1.1	3.3	532 BGA, 23 mm	90.00
TMS320C6414TBGLZ1	16K/16K/1M	3	64	HPI 32/16	3	1000	8000	0.44	1.65	1.2	3.3	532 BGA, 23 mm	207.85
TMS320C6414TBGLZ8	16K/16K/1M	3	64	HPI 32/16	3	850	6800	TBD	TBD	1.2	3.3	532 BGA, 23 mm	157.40
TMS320C6414TBGLZ7	16K/16K/1M	3	64	HPI 32/16	3	720	5760	0.44	1.36	1.2	3.3	532 BGA, 23 mm	106.95
TMS320C6414TBGLZ6	16K/16K/1M	3	64	HPI 32/16	3	600	4800	0.39	1.1	1.1	3.3	532 BGA, 23 mm	85.55

Introduction – DSP/FPGA (cont.)

- Xilinx FPGAs
  - Virtex series – high performance, high cost
    - $\boxed{\text{Virtex-II pro}}$
    - $\boxed{\text{Virtex-4}}$
    - Virtex-5
    - Virtex-6
  - Spartan series – mid performance, low cost
    - Spartan-II
    - Spartan-3
    - Spartan-6

Introduction – DSP/FPGA (cont.)

FEATURE Comparison Table

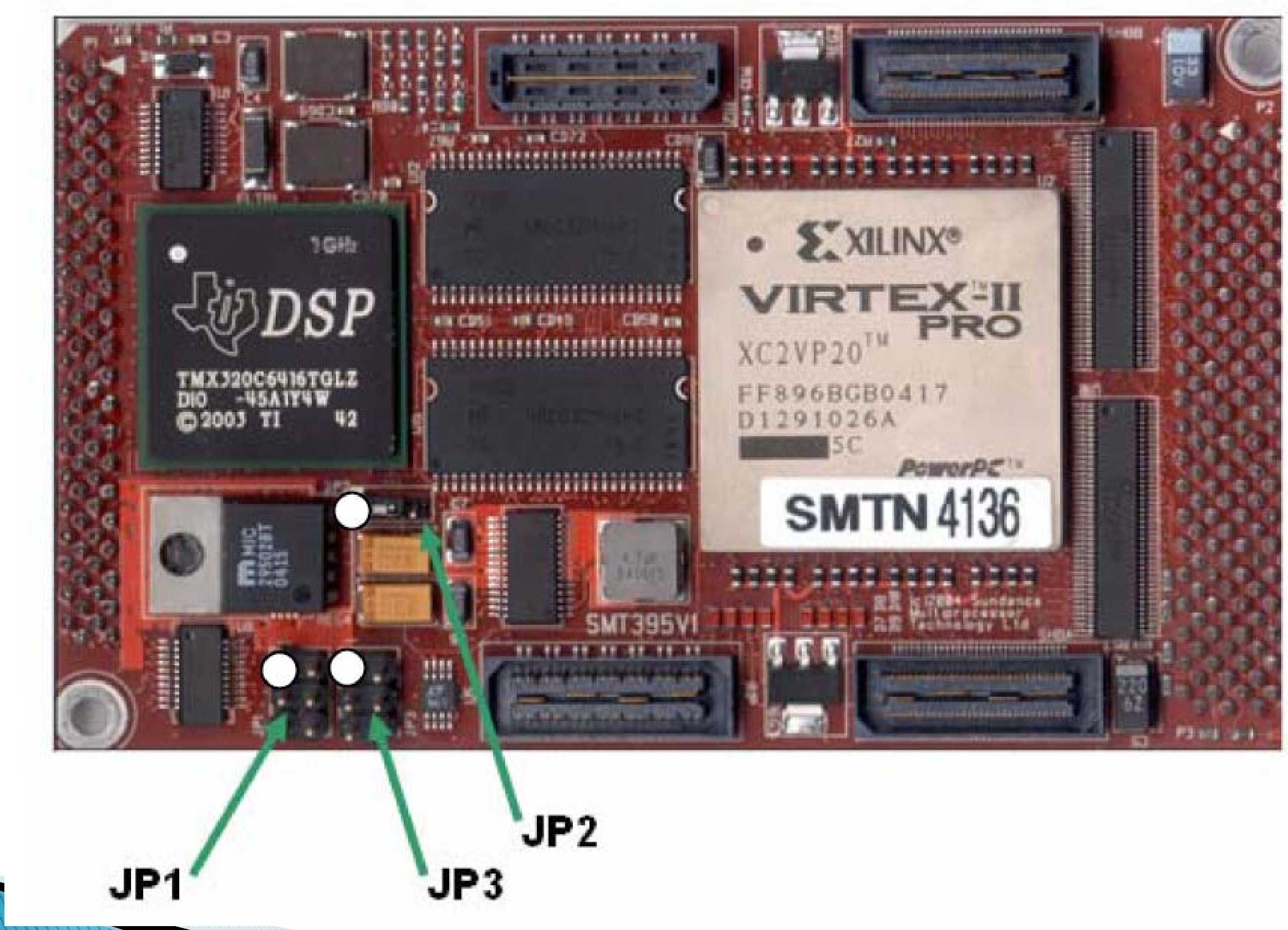
Features	Virtex-5	Virtex-4	Extended Spartan-3A
Logic Cells	Up to 330,000	Up to 200,000	Up to 53,000
User I/Os	Up to 1200	Up to 960	Up to 519 I/O
I/O Standards Supported	Over 40	Over 20	Over 20
Clock Management - DCM	Yes	Yes	Yes
Clock Management - PLL	Yes	No	No
Embedded Block RAM	Up to 18 Mbits	Up to 11Mbits	Up to 1.8 Mbits
Embedded Multipliers for DSP	Yes (25 x 18 MAC)	Yes (18 x 18 MAC)	Yes (18 x 18 MAC)
Multi-Gigabit High Speed Serial	Yes	Yes	No
Soft Processor Support	Yes	Yes	Yes
Embedded PowerPC® Processors	Yes (PowerPC 440 Processor)	Yes (PowerPC 405 Processor)	No

Hardware – DSP module

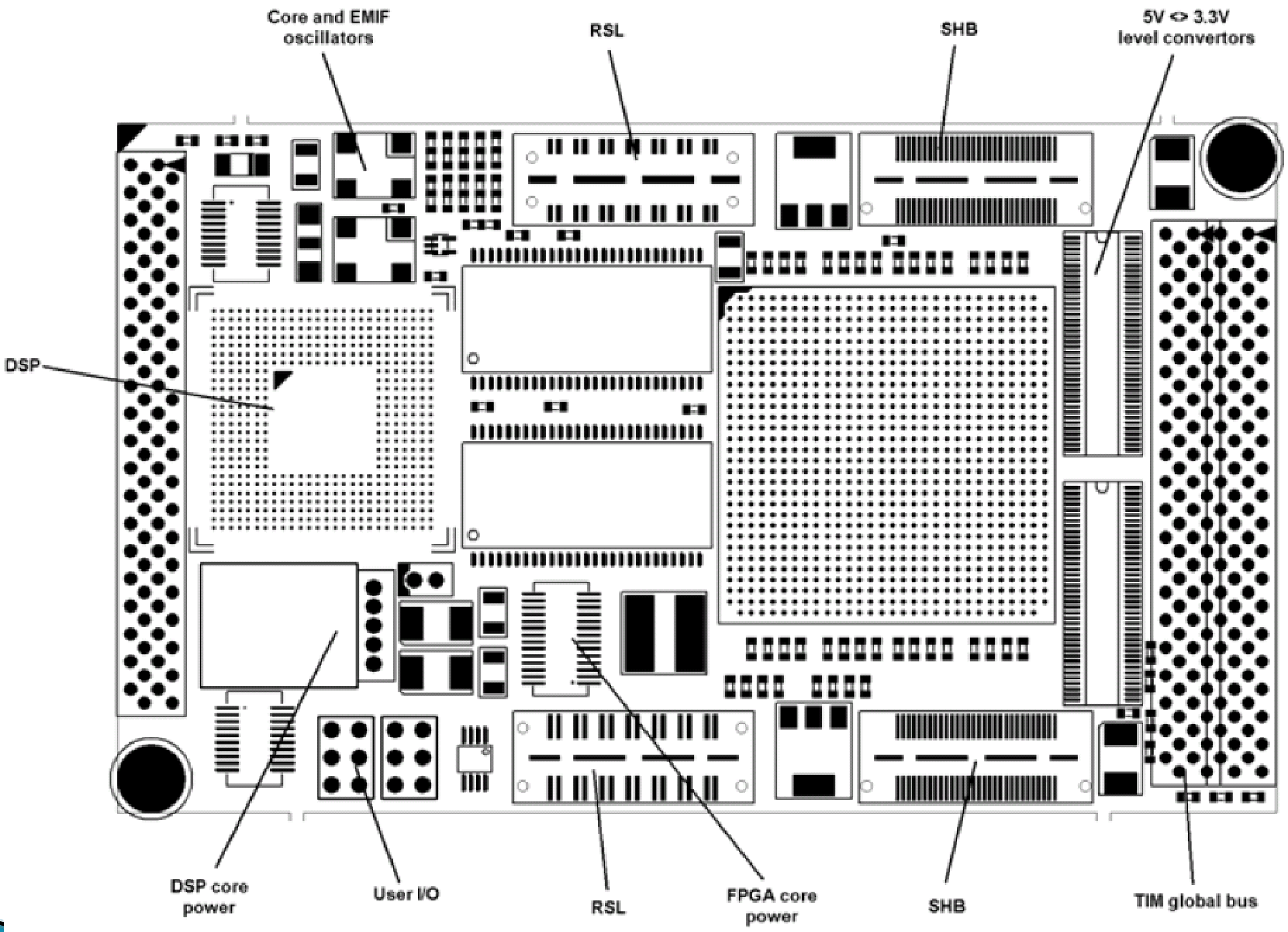
SMT395-VP30-6 characteristics:

- ⇒ 1GHz TMS320C6416T fixed-point DSP,
- ⇒ 8000 MIPS peak performance,
- ⇒ Xilinx Virtex-II Pro VP30-6 FF896 package,
- ⇒ Six 20MB/s communication ports (Comports),
- ⇒ 256MBytes of SDRAM (133MHz),
- ⇒ 8MByte Flash ROM for boot code and FPGA programming,
- ⇒ Global expansion connector,
- ⇒ High bandwidth data I/O via 2 Sundance High-speed Buses (SHB),
- ⇒ JTAG Diagnostics port.

Hardware – DSP module (cont.)

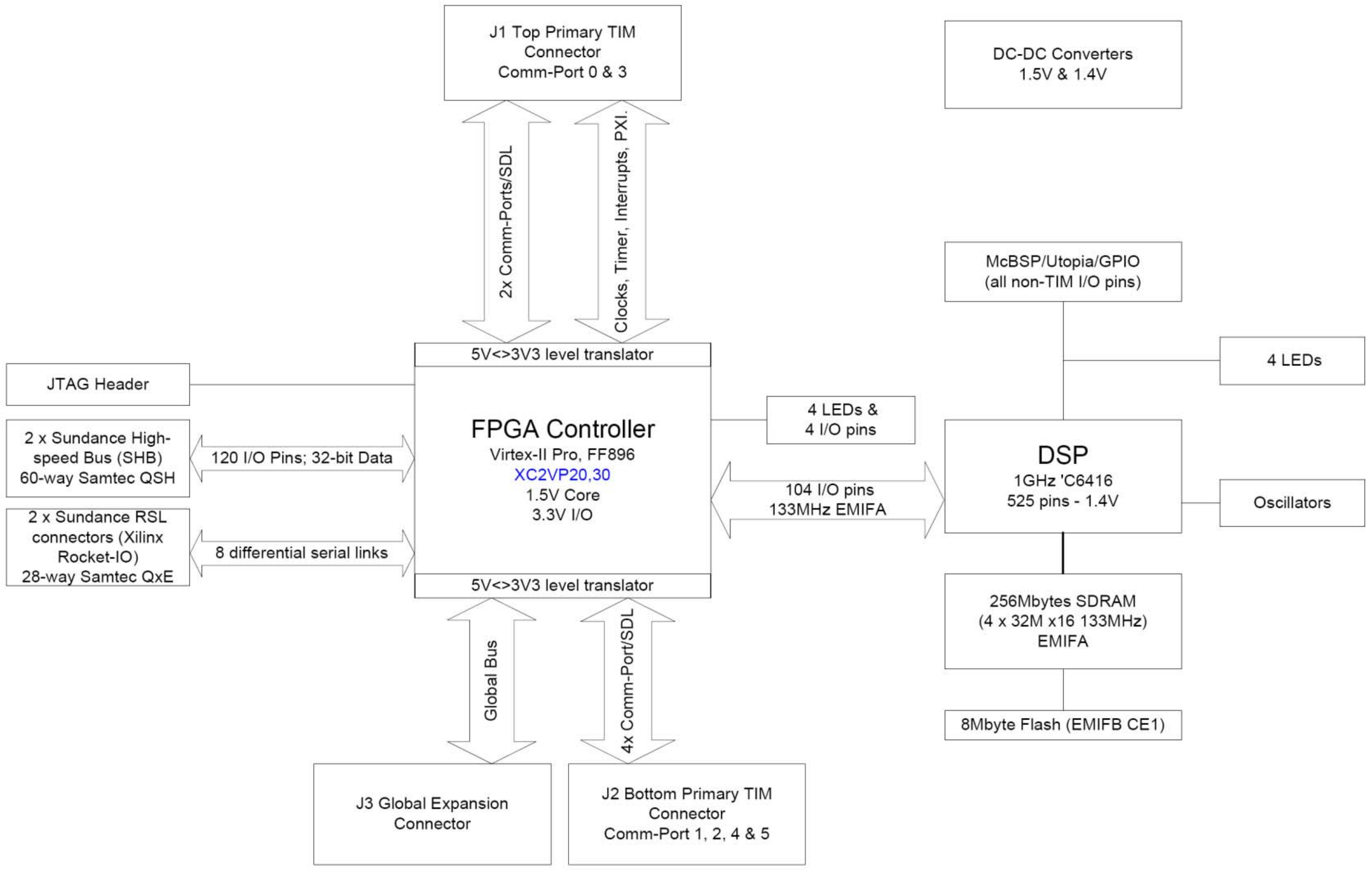


Hardware – DSP module (cont.)

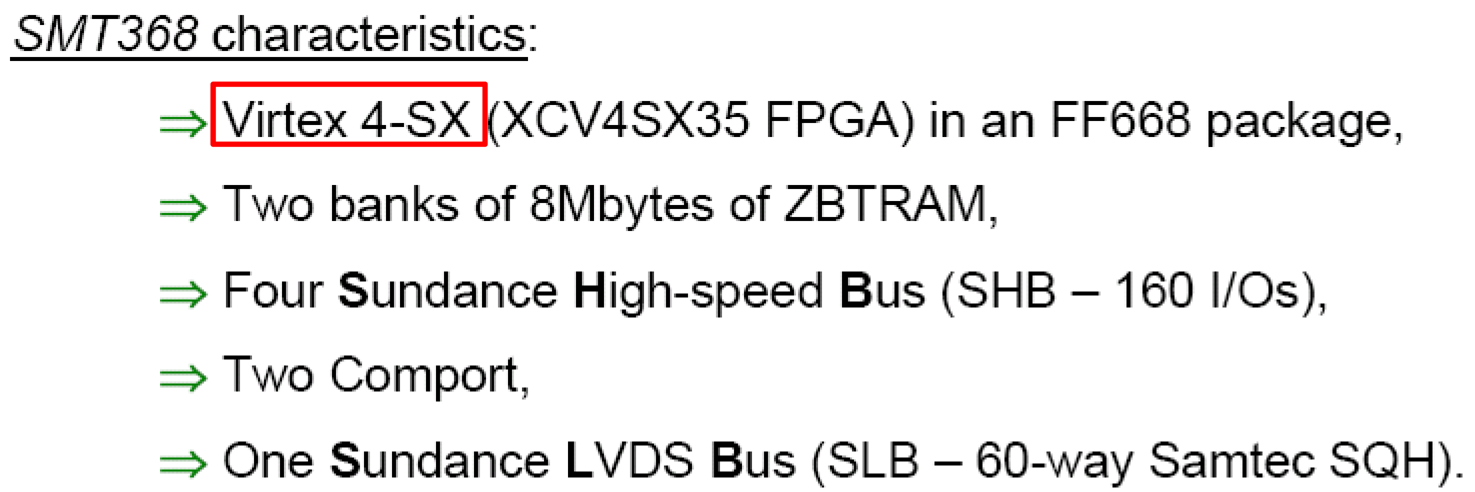


- RSL (Rocket Serial link): 2.5Gbit/s

Hardware – DSP module (cont.)


J1 Top Primary TIM Connector Comm-Port 0 & 3
DC-DC Converters 1.5V & 1.4V
2x Comm-Ports/SDL
Clocks, Timer, Interrupts, PXI.
McBSP/Utopia/GPIO (all non-TIM I/O pins)
4 LEDs
5V<>3V3 level translator
JTAG Header
FPGA Controller
Virtex-II Pro, FF896
XC2VP20,30
1.5V Core
3.3V I/O
4 LEDs & 4 I/O pins
2 x Sundance High-speed Bus (SHB) 60-way Samtec QSH
120 I/O Pins; 32-bit Data
DSP
1GHz 'C6416
525 pins - 1.4V
104 I/O pins 133MHz EMIFA
Oscillators
2 x Sundance RSL connectors (Xilinx Rocket-IO) 28-way Samtec QxE
8 differential serial links
5V<>3V3 level translator
256Mbytes SDRAM (4 x 32M x16 133MHz) EMIFA
Global Bus
4x Comm-Port/SDL
8Mbyte Flash (EMIFB CE1)
J3 Global Expansion Connector
J2 Bottom Primary TIM Connector Comm-Port 1, 2, 4 & 5

Hardware – FPGA module



SMT368 characteristics:

⇒ Virtex 4-SX (XCV4SX35 FPGA) in an FF668 package,

⇒ Two banks of 8Mbytes of ZBTRAM,

⇒ Four **S**undance **H**igh-speed **B**us (SHB – 160 I/Os),

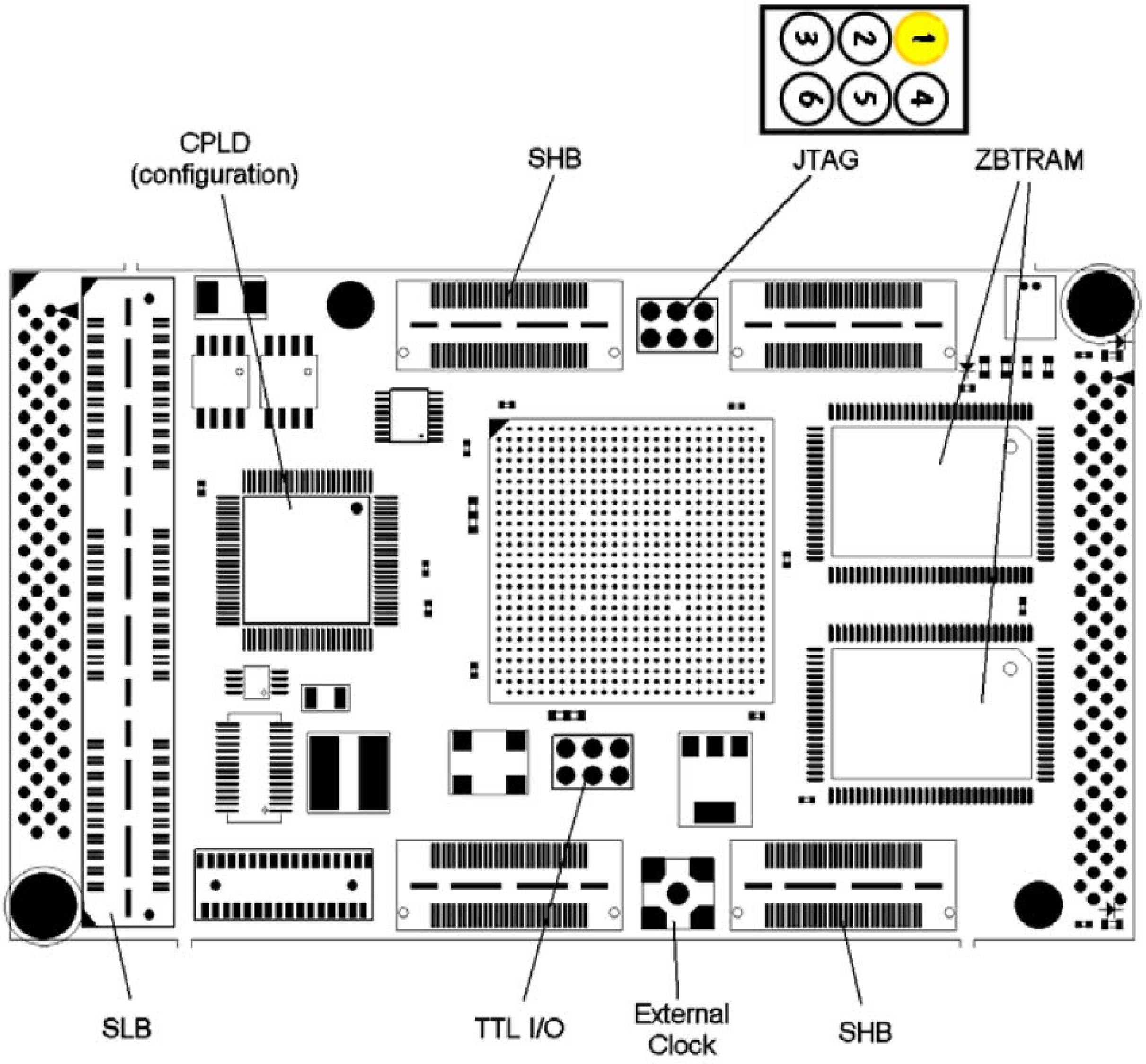
⇒ Two Comport,

⇒ One **S**undance **L**VDS **B**us (SLB – 60-way Samtec SQH).

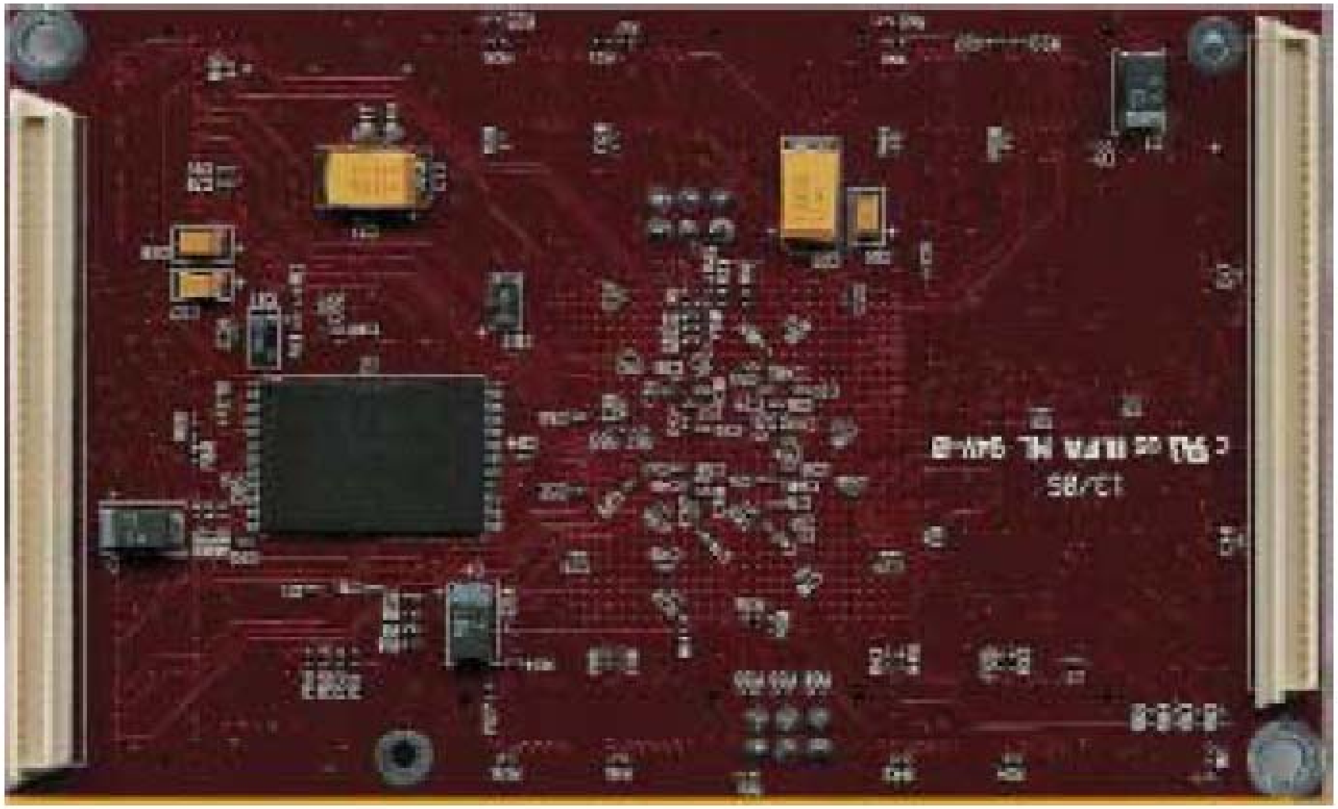
- ZBTRAM (Zero Bus Turnaround Random Access Memory): designed to sustain 100% bus bandwidth by eliminating turnaround cycle when there is transition form Read to Write, or vice-versa.
- LVDS (Low Voltage Differential Signaling): an electrical signaling system that can run at very high speeds over inexpensive twisted-pair copper cables

Hardware – FPGA module (cont.)

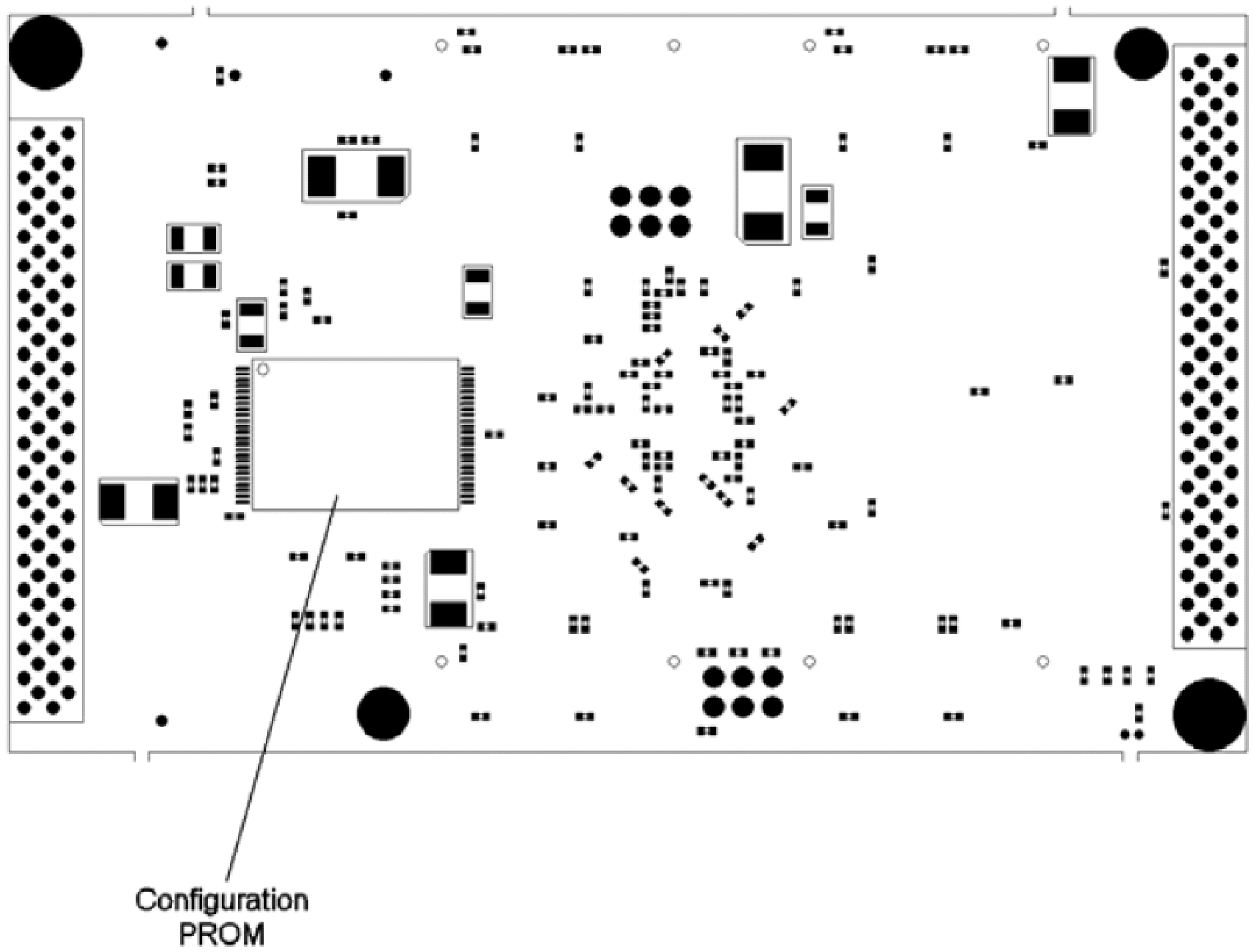
Hardware – FPGA module (cont.)



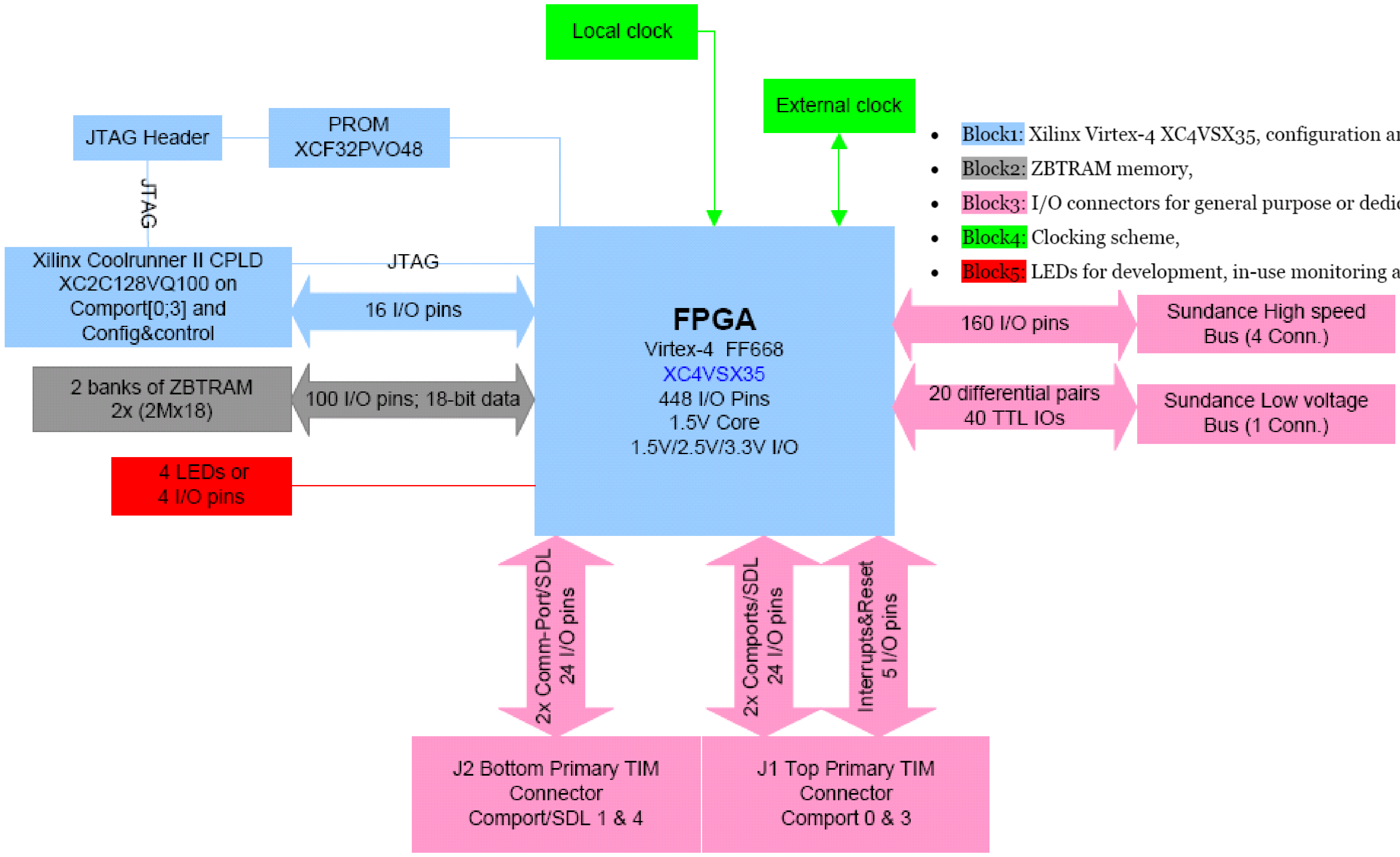
Hardware – FPGA module (cont.)



Hardware – FPGA module (cont.)



Hardware – FPGA module (cont.)



Local clock
External clock
JTAG Header
PROM XCF32PVO48
JTAG
Xilinx Coolrunner II CPLD XC2C128VQ100 on Comport[0;3] and Config&control
JTAG
16 I/O pins
2 banks of ZBTRAM 2x (2Mx18)
100 I/O pins; 18-bit data
4 LEDs or 4 I/O pins
FPGA
Virtex-4 FF668
XC4VSX35
448 I/O Pins
1.5V Core
1.5V/2.5V/3.3V I/O
160 I/O pins
Sundance High speed Bus (4 Conn.)
20 differential pairs
40 TTL IOs
Sundance Low voltage Bus (1 Conn.)
2x Comm-Port/SDL 24 I/O pins
2x Comports/SDL 24 I/O pins
Interrupts&Reset 5 I/O pins
J2 Bottom Primary TIM Connector Comport/SDL 1 & 4
J1 Top Primary TIM Connector Comport 0 & 3

- Block1: Xilinx Virtex-4 XC4VSX35, configuration and reset schemes,
- Block2: ZBTRAM memory,
- Block3: I/O connectors for general purpose or dedicated interfaces,
- Block4: Clocking scheme,
- Block5: LEDs for development, in-use monitoring and general purpose use.

Hardware – ADC/DAC module

SMT350 characteristics:

⇒ Two 14-bit ADCs (TI – ADS5500) sampling at up to 105MHz,

⇒ Dual 16-bit DAC (DAC5686) sampling at up to 500MHz (interpolation),

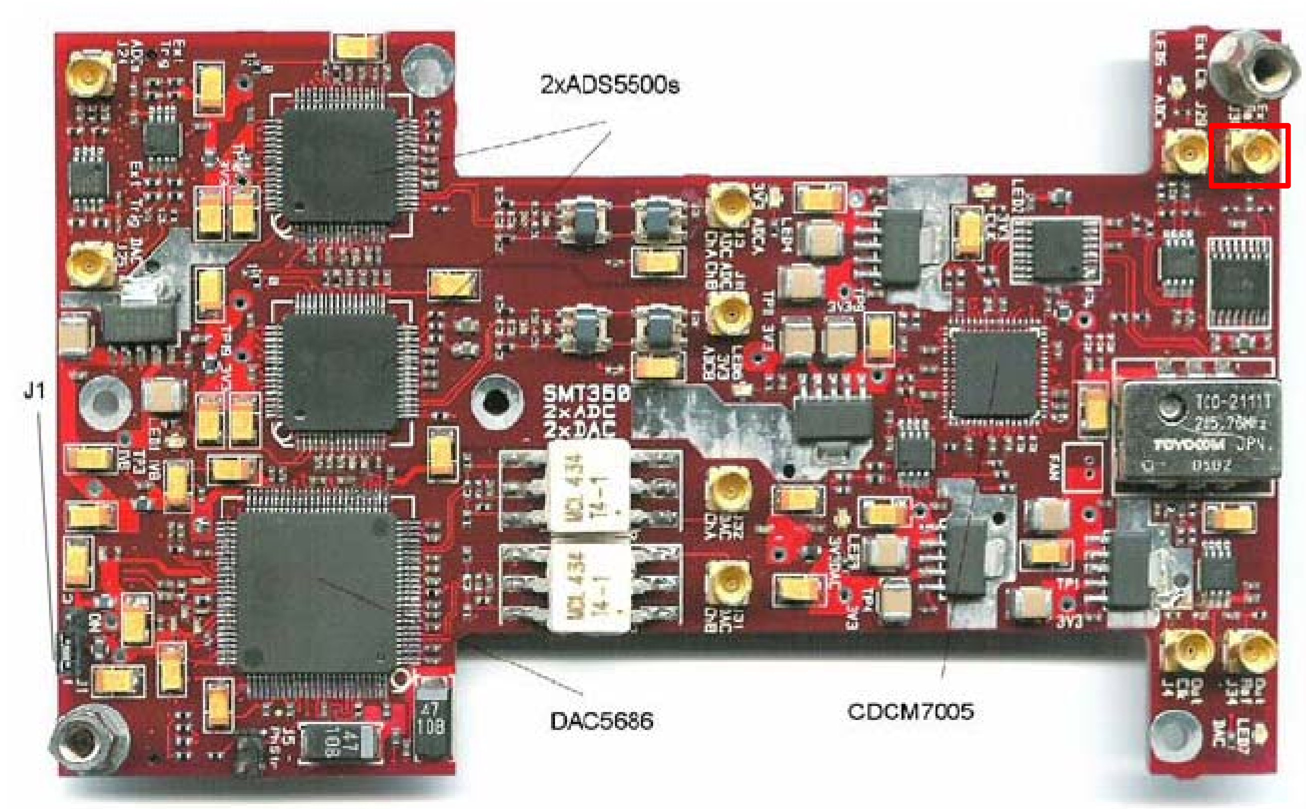
⇒ One **S**undance **L**VDS **B**us (SLB – 60-way Samtec SQH),

⇒ Low-jitter on-board system clock based around the combination of a VCXO and the TI - CDCM7005,

⇒ 50-Ohm terminated analogue inputs and outputs, external triggers and clocks via MMCX (Huber and Suhner) connectors.

Hardware – ADC/DAC module (cont.)

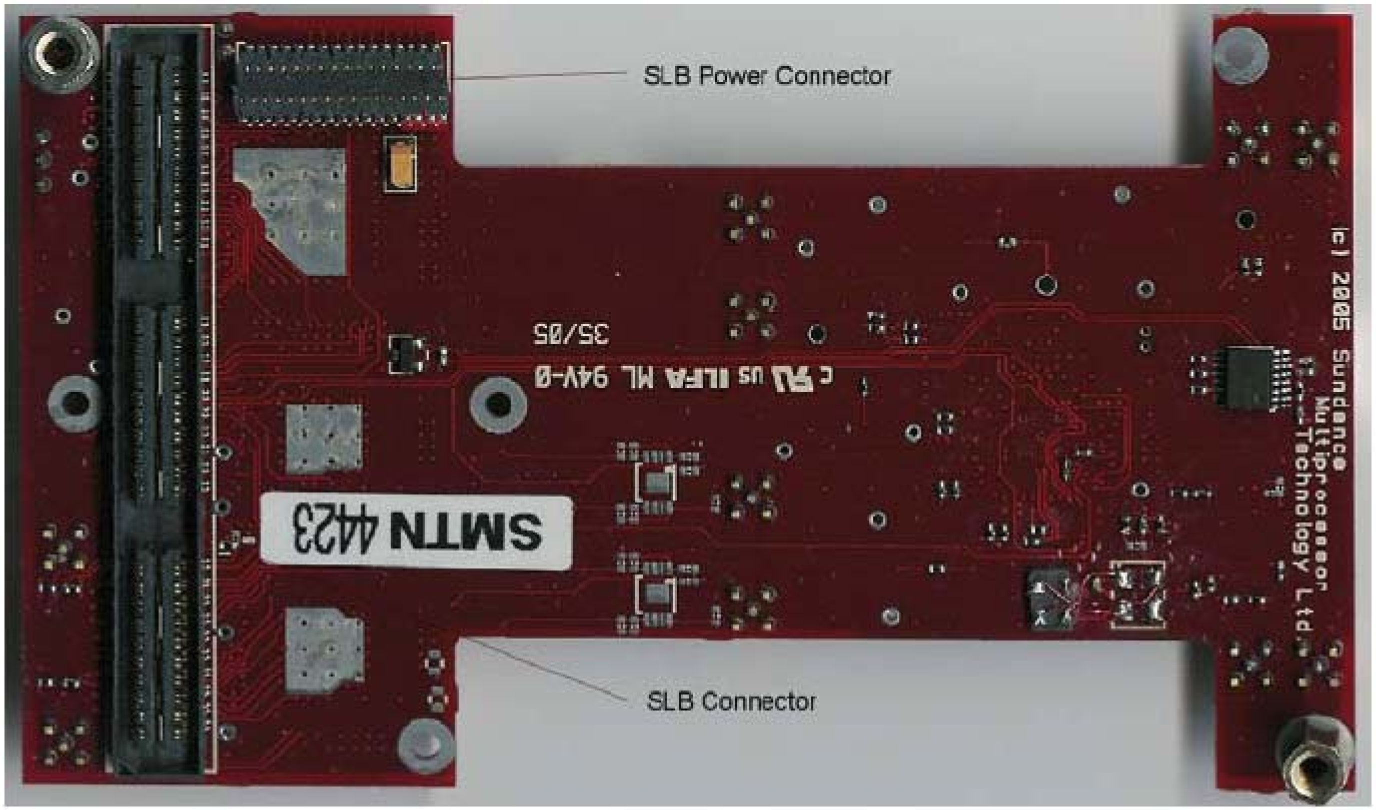
- Daughter module component side



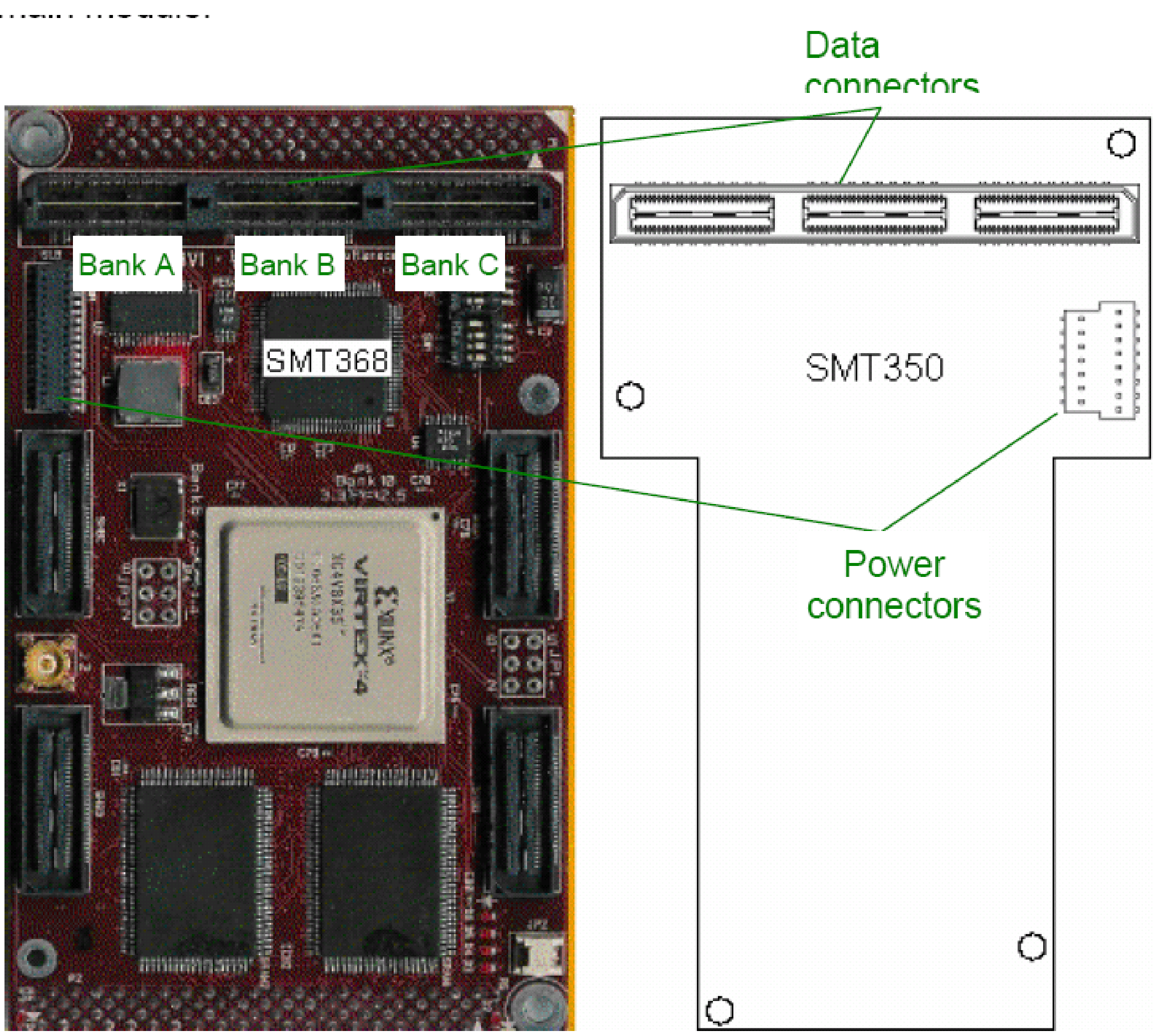
mmcx type connector

Hardware – ADC/DAC module (cont.)

- Daughter module solder side

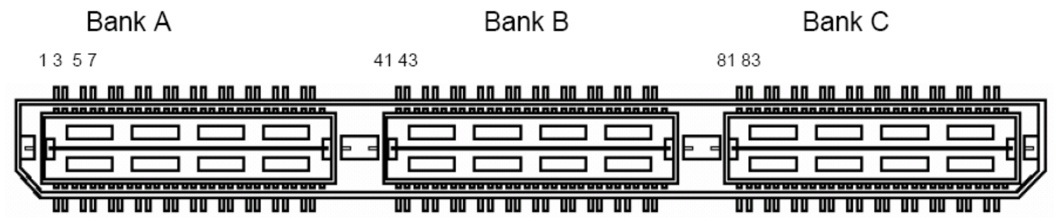


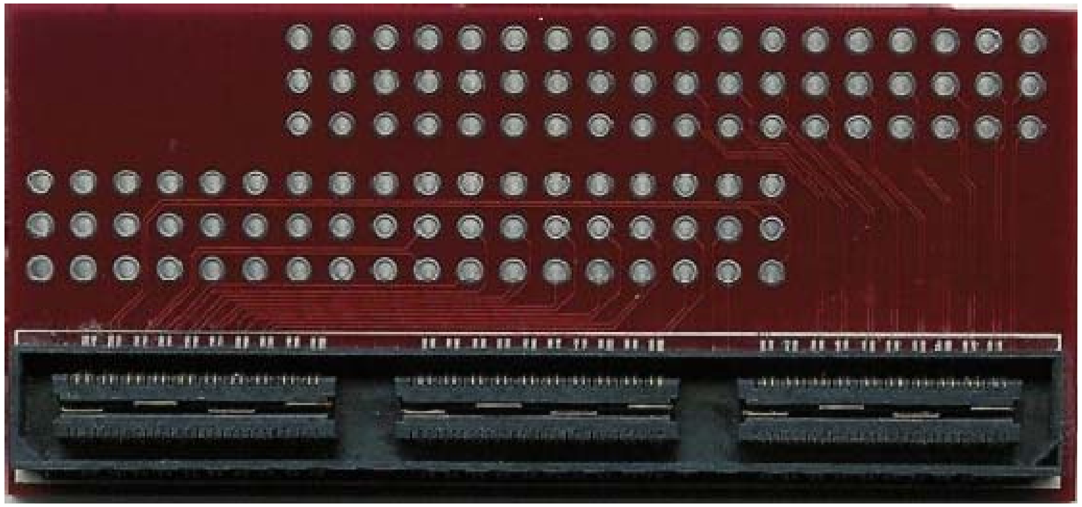
Hardware – ADC/DAC module (cont.)



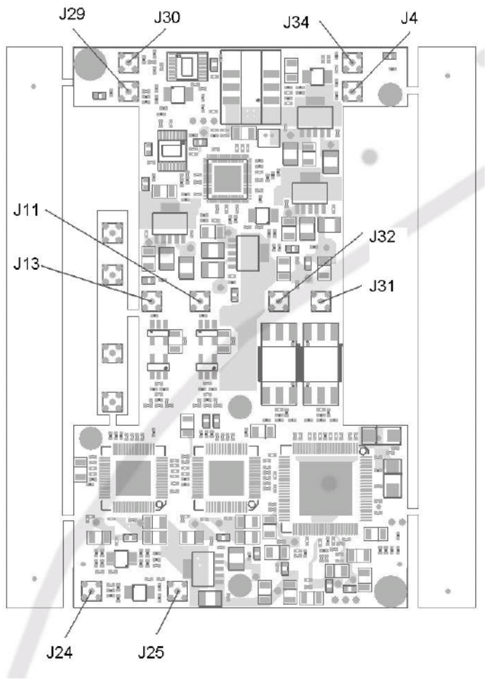
Hardware – ADC/DAC module (cont.)

- PCB for probing purpose – SMT598





Hardware – ADC/DAC module (cont.)



Connector name and (silkscreen schematics)	Description	Location on the board
J13	ADCA Analog Input	Middle / Left
J11	ADCB Analog Input	Middle / Left
J32	DACA Analog Output	Middle / Right
J31	DACB Analog Output	Middle / Right
J30	External Reference Input	Top / Left
J29	External Clock Input	Top / Left
J34	External Reference Output	Top / Right
J4	External Clock Output	Top / Right
J24	External Trigger ADCs	Bottom / Left
J25	External Trigger DAC	Bottom / Left

Software – Basic setup

- TI CCS (Code Composer Studio) 3.3 or higher
  - TI development tool suite (compiler, linker, etc)
- Xilinx ISE 9.2i (ISE 10.1 is not compatible)
  - Xilinx development tool suite (synthesis, simulation)
- Sundance driver
- 3L Diamond
  - A set of tools to create efficient applications using multiprocessor hardware made from DSPs and FPGAs
  - Provides an integrated development environment
  - Diamond server/ Diamond IDE

Software – Basic setup (cont.)

Why do I need Diamond?

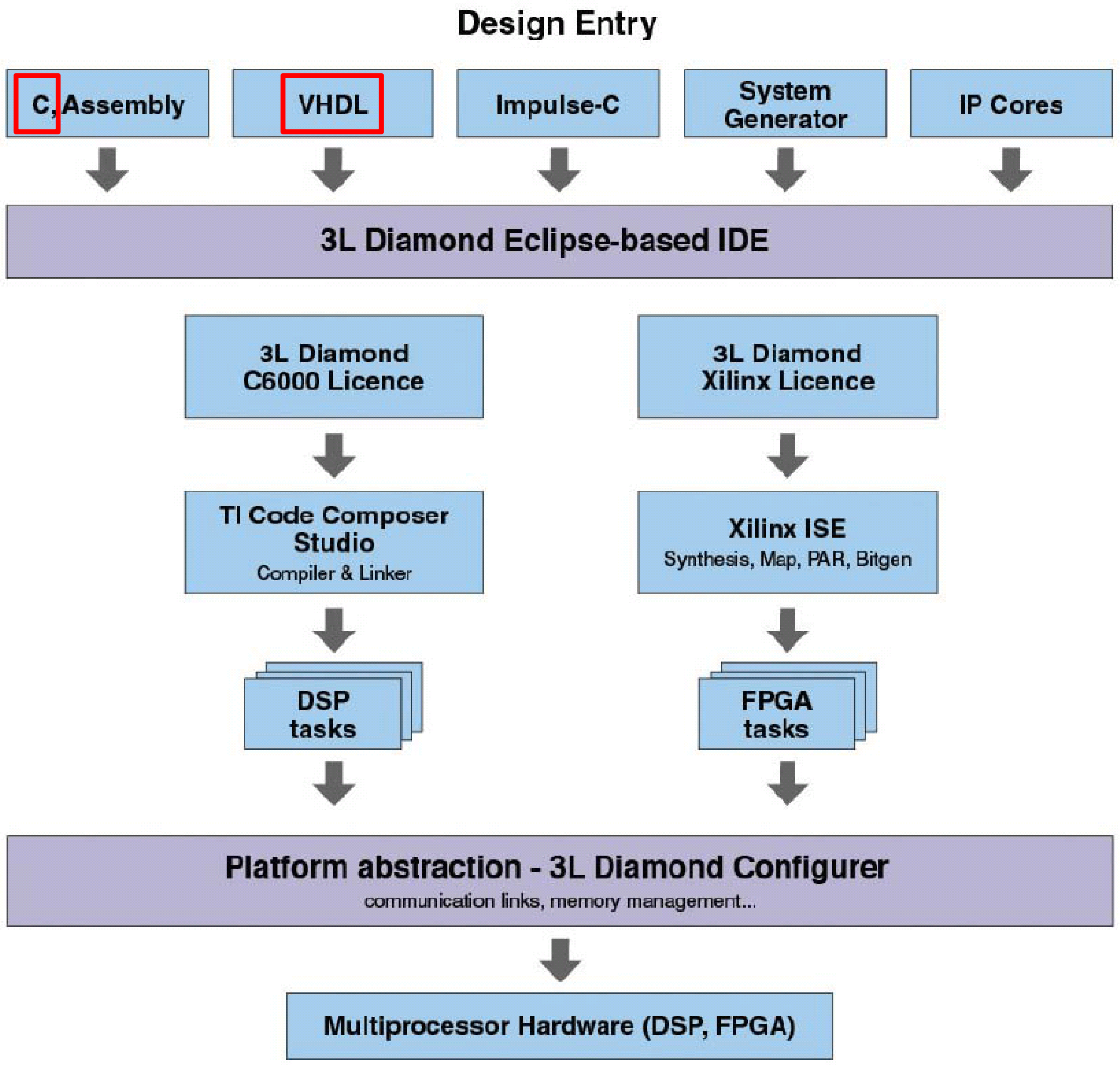
- This is rather like asking, why do I need a high-level language? The strict answer is that you can make do without tools to help you build applications, but you will be making your task an order of magnitude harder.

- If you decide not to use Diamond to build a multiprocessor application you will have to do all the work yourself, including things like:

  - loading all the processors, including ones remote from the host PC
  - keeping control of all the separate modules needed to load your application
  - starting your application in a synchronised way
  - managing communications, possibly including deadlock-free message routing
  - writing your own device drivers
  - explicitly managing all memory allocation
  - writing your own multithreading support or using something that is likely to be less efficient than Diamond's
  - inventing a host communication mechanism
  - being prepared to rewrite your source if the configuration changes
  - being prepared to make major changes if the underlying hardware changes
  - supporting multiple source versions for all hardware and configuration variations
  - handling all of the underlying hardware peculiarities (unexpected cache behaviour, for example)
  - ... and many more.

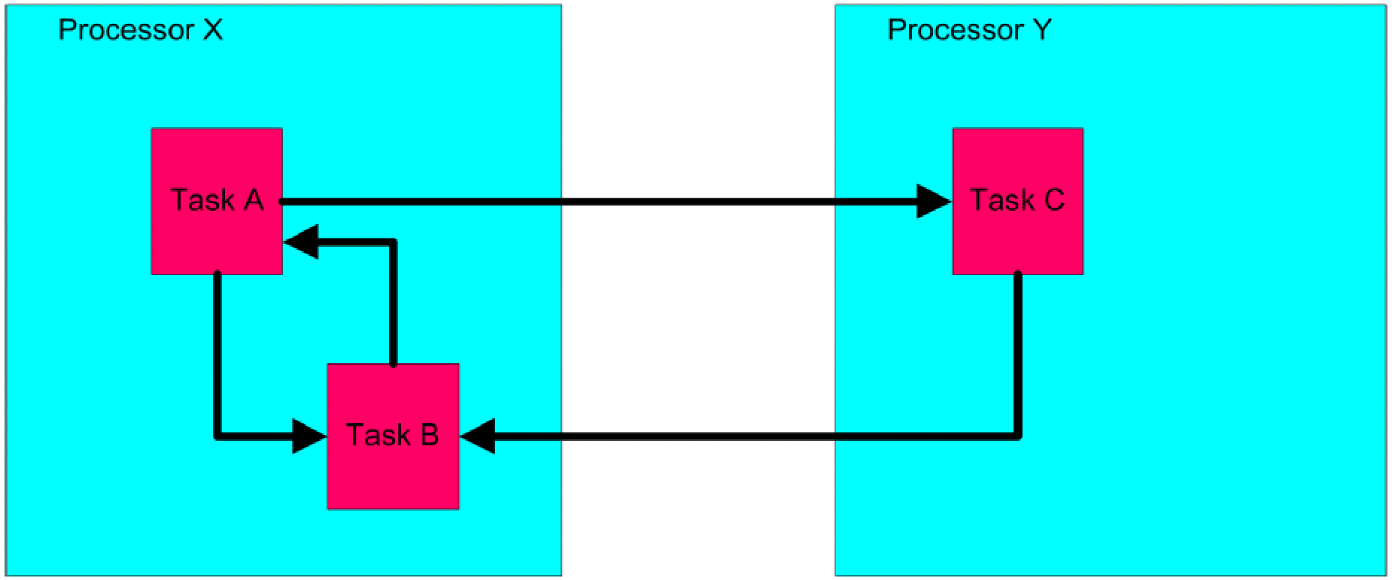
- Diamond can do all of this for you. 3L works closely with hardware vendors and puts a great deal of time, effort, and experience into optimising all aspects of the system to give you the best results.

Software – 3L Diamond


Design Entry
C, Assembly
VHDL
Impulse-C
System Generator
IP Cores
3L Diamond Eclipse-based IDE
3L Diamond C6000 Licence
3L Diamond Xilinx Licence
TI Code Composer Studio
Compiler & Linker
Xilinx ISE
Synthesis, Map, PAR, Bitgen
DSP tasks
FPGA tasks
Platform abstraction - 3L Diamond Configurer
communication links, memory management...
Multiprocessor Hardware (DSP, FPGA)

Software – 3L Diamond (cont.)

- The Diamond Model
  - In this model, a computing system is a collection of concurrently active sequential processes that can only communicate with each other over channels.
  - DSP tasks are implemented in C while FPGA tasks may be implemented directly in VHDL.



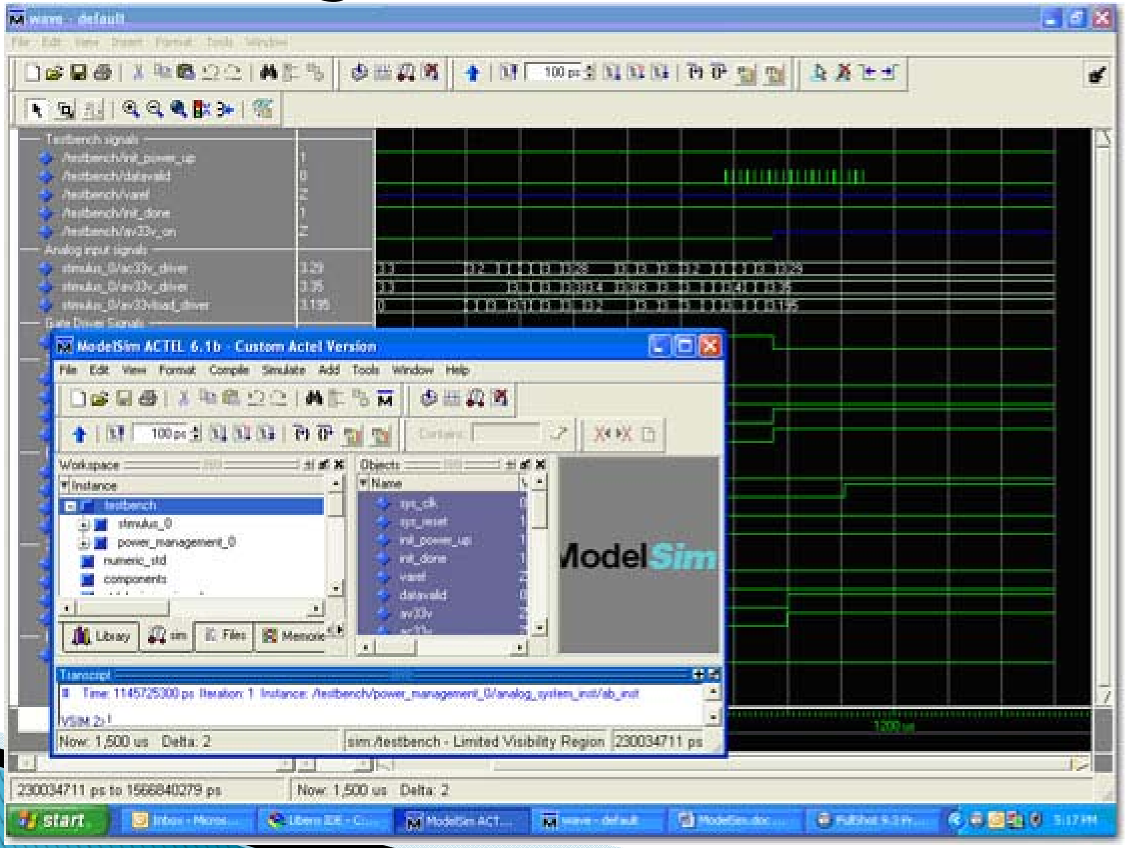
Software – Xilinx ISE

- Xilinx ISE (Integrated Software Environment)
  - Synthesis: HDL code로부터 netlist 파일(*.ngc) 생성
  - Implement
    - Translate: 여러 디자인을 하나의 netlist로 구성
    - Map: netlist로부터 physical component 구성 (slices and IOBs)
    - Place and route: 칩위에 구성물을 구성하고, 그 구성물을 연결하며 타이밍 데이터를 리포트함
  - Configuration: generate PROM files and download to devices using iMPACT

  - ISIM (ISE Simulator): PC 환경에서 로직 검증

Software – *Modelsim*

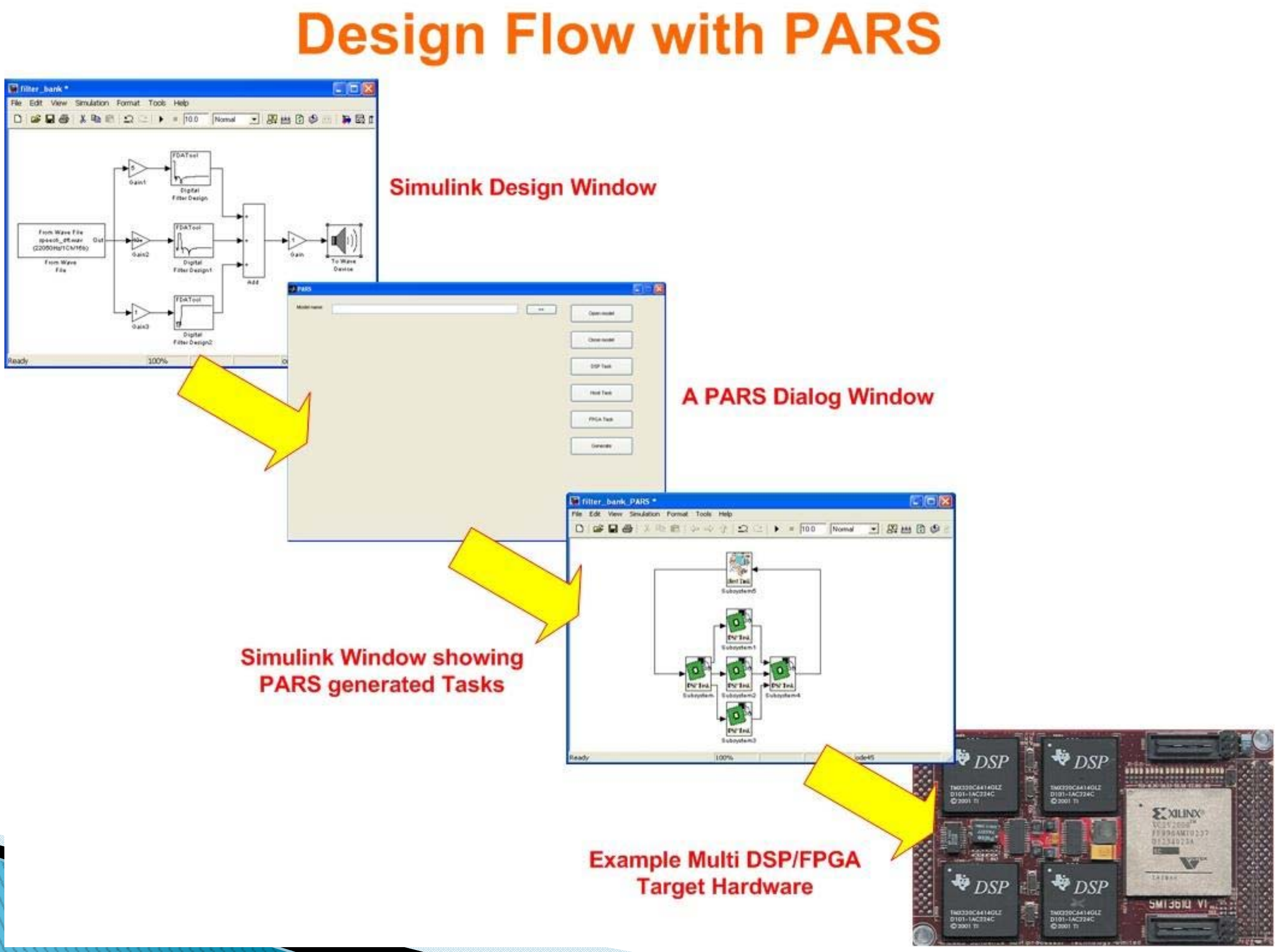
- *ModelSim* provides a comprehensive simulation and debug environment for complex ASIC and FPGA designs.



```
# Time: 1145725300 ps Iteration: 1 Instance: /testbench/power_management_0/analog_system_init/ab_init
VSIM 2>
Now: 1,500 us   Delta: 2   sim:/testbench - Limited Visibility Region   230034711 ps
230034711 ps to 1566840279 ps   Now: 1,500 us   Delta: 2
```

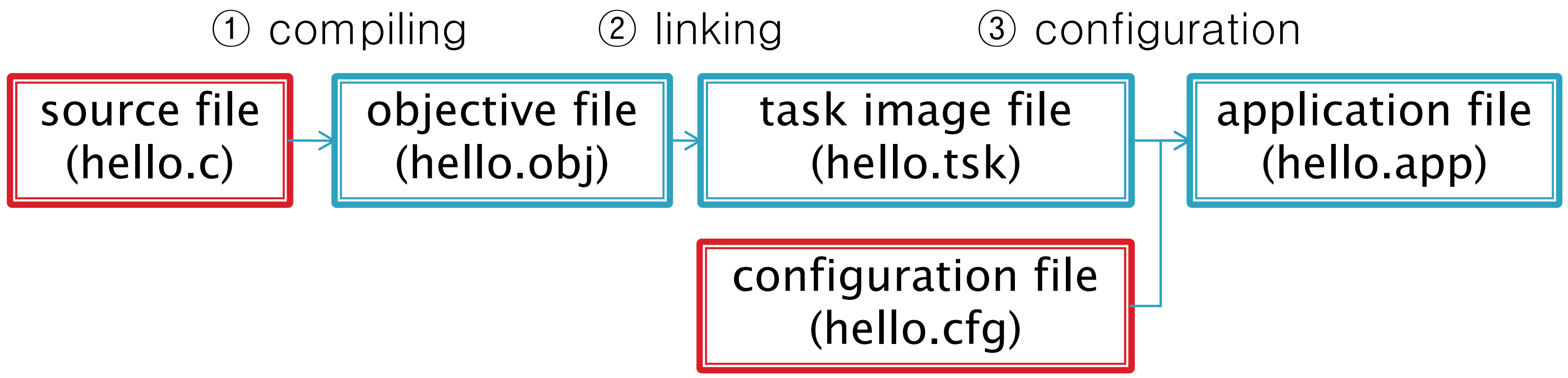
Software – PARS

- Parallel Application from Rapid Simulation

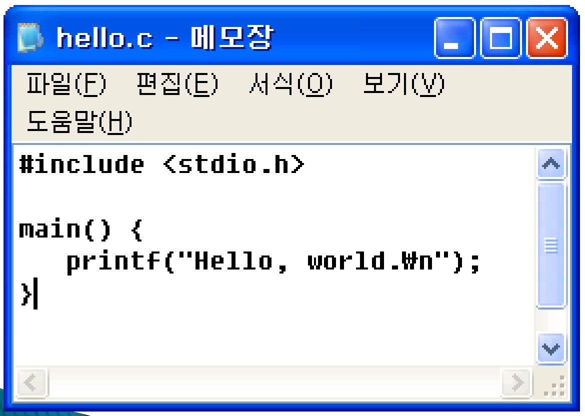


DSP example 1 – Hello.c

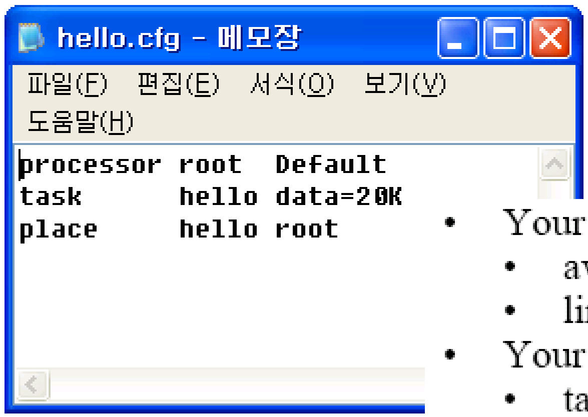
- Application file 생성 과정



① compiling
② linking
③ configuration
source file (hello.c)
objective file (hello.obj)
task image file (hello.tsk)
application file (hello.app)
configuration file (hello.cfg)



```
#include <stdio.h>

main() {
   printf("Hello, world.₩n");
}
```

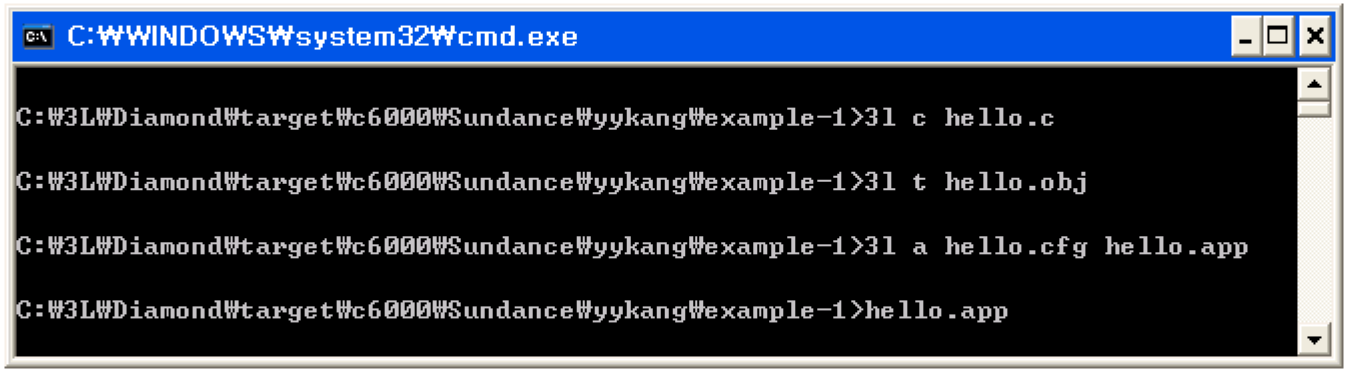


```
processor root  Default
task      hello data=20K
place     hello root
```

- Your hardware structure:
  - available processors
  - links connecting them
- Your software structure:
  - tasks to be included
  - channel connections between them
- How to map the software onto the hardware.

DSP example 1 – Hello.c (cont.)

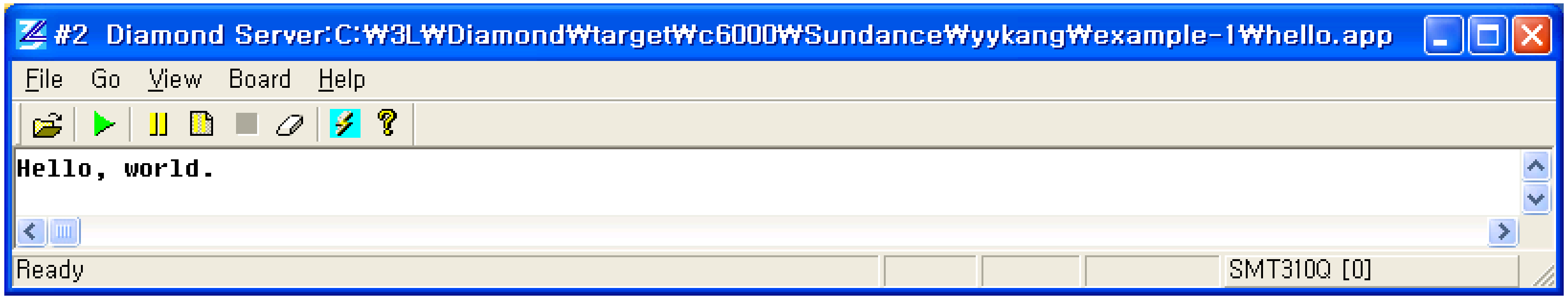
- Application file 생성 과정



C:₩WINDOWS₩system32₩cmd.exe

```
C:₩3L₩Diamond₩target₩c6000₩Sundance₩yykang₩example-1>3l c hello.c

C:₩3L₩Diamond₩target₩c6000₩Sundance₩yykang₩example-1>3l t hello.obj

C:₩3L₩Diamond₩target₩c6000₩Sundance₩yykang₩example-1>3l a hello.cfg hello.app

C:₩3L₩Diamond₩target₩c6000₩Sundance₩yykang₩example-1>hello.app
```

- Application 실행 화면

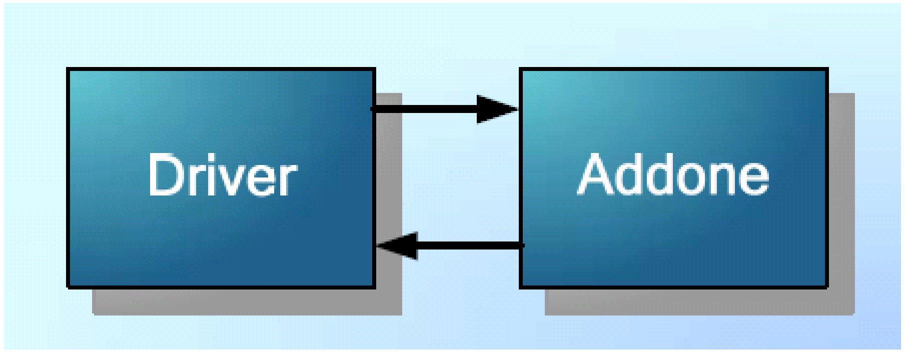


#2 Diamond Server:C:₩3L₩Diamond₩target₩c6000₩Sundance₩yykang₩example-1₩hello.app

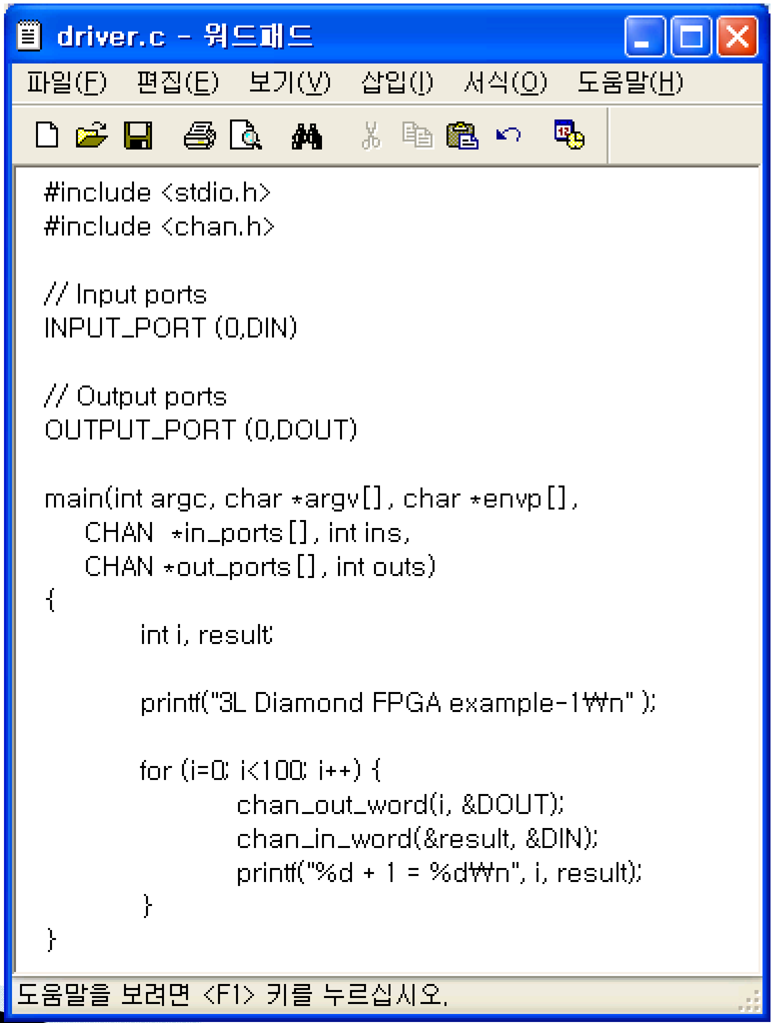
```
Hello, world.
```

FPGA example 1 – addone

- Overview
  - In this example the DSP generates data that are sent to a task in the FPGA that increments them by one. The result is sent to the DSP which displays it on the host PC.

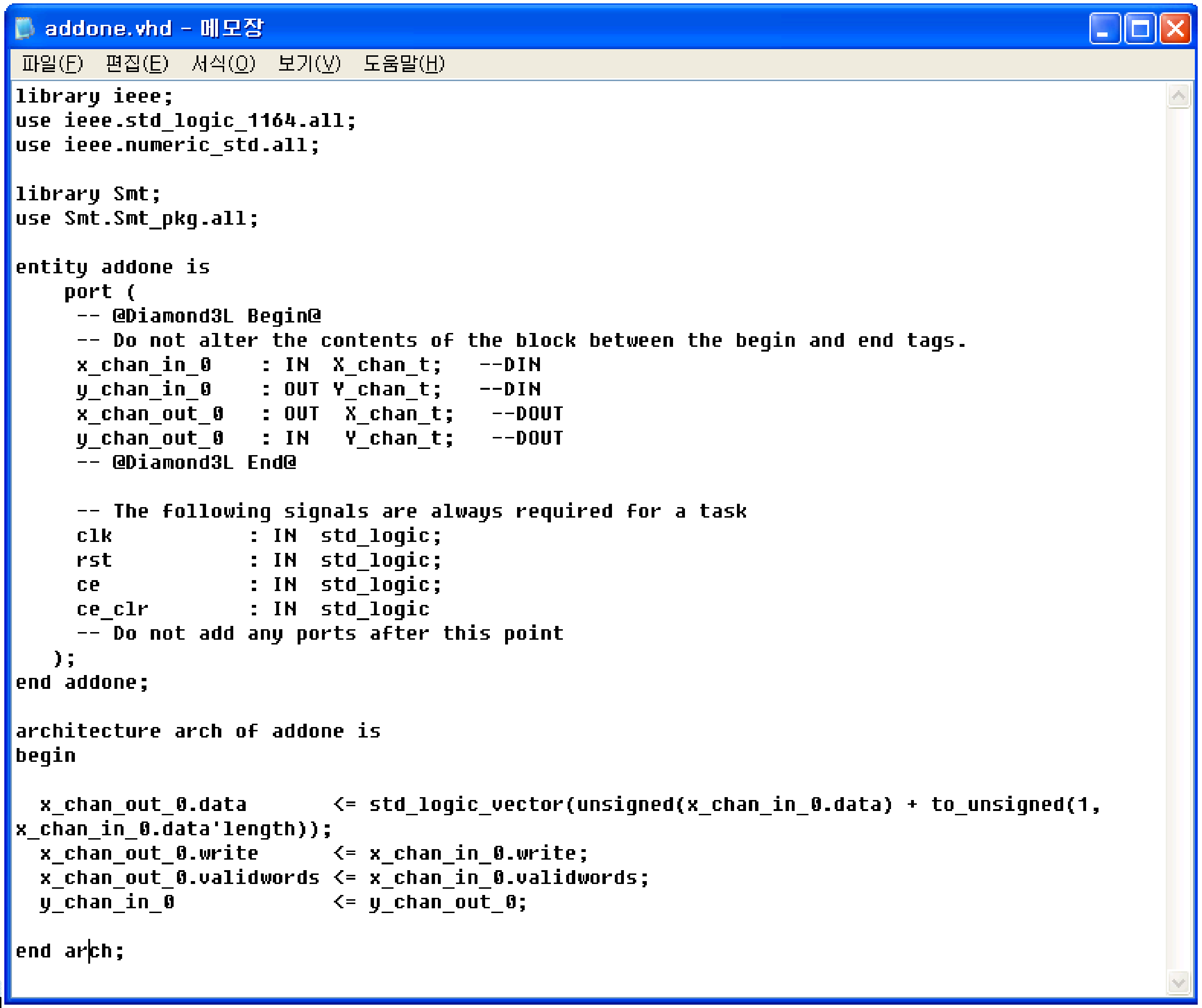


FPGA example 1 – addone (cont.)


driver.c - 워드패드
파일(F) 편집(E) 보기(V) 삽입(I) 서식(O) 도움말(H)

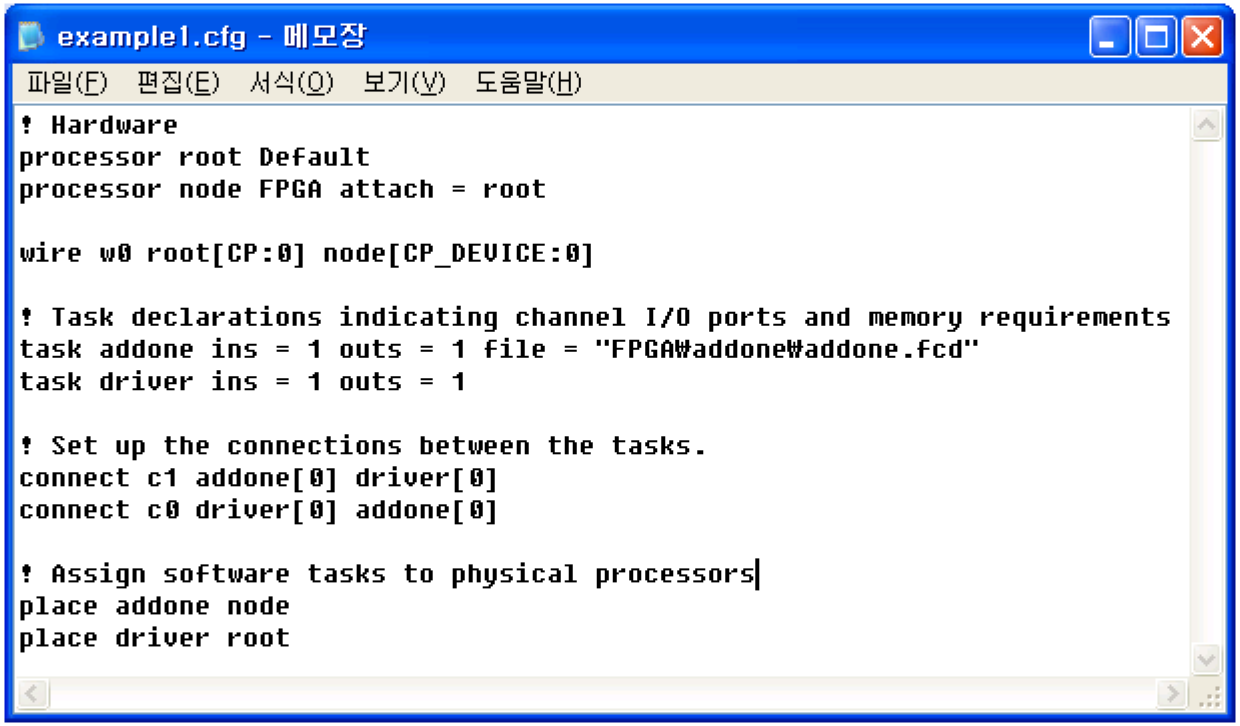
```
#include <stdio.h>
#include <chan.h>

// Input ports
INPUT_PORT (0,DIN)

// Output ports
OUTPUT_PORT (0,DOUT)

main(int argc, char *argv[], char *envp[],
    CHAN  *in_ports[], int ins,
    CHAN *out_ports[], int outs)
{
        int i, result;

        printf("3L Diamond FPGA example-1₩n" );

        for (i=0; i<100; i++) {
                chan_out_word(i, &DOUT);
                chan_in_word(&result, &DIN);
                printf("%d + 1 = %d₩n", i, result);
        }
}
```

FPGA example 1 – addone (cont.)



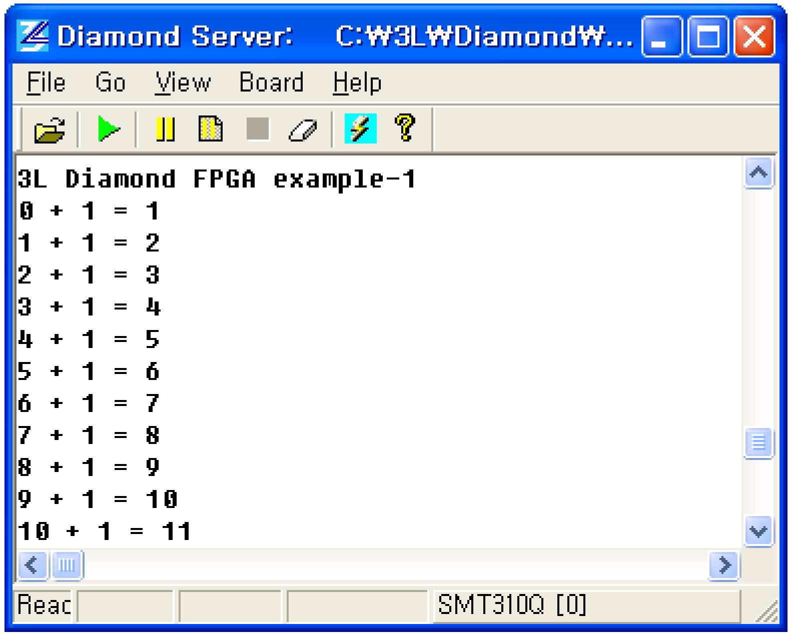
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

library Smt;
use Smt.Smt_pkg.all;

entity addone is
    port (
     -- @Diamond3L Begin@
     -- Do not alter the contents of the block between the begin and end tags.
     x_chan_in_0     : IN  X_chan_t;   --DIN
     y_chan_in_0     : OUT Y_chan_t;   --DIN
     x_chan_out_0    : OUT  X_chan_t;   --DOUT
     y_chan_out_0    : IN   Y_chan_t;   --DOUT
     -- @Diamond3L End@

     -- The following signals are always required for a task
     clk            : IN  std_logic;
     rst            : IN  std_logic;
     ce             : IN  std_logic;
     ce_clr         : IN  std_logic
     -- Do not add any ports after this point
   );
end addone;

architecture arch of addone is
begin

  x_chan_out_0.data       <= std_logic_vector(unsigned(x_chan_in_0.data) + to_unsigned(1,
x_chan_in_0.data'length));
  x_chan_out_0.write      <= x_chan_in_0.write;
  x_chan_out_0.validwords <= x_chan_in_0.validwords;
  y_chan_in_0             <= y_chan_out_0;

end arch;
```

FPGA example 1 – addone (cont.)



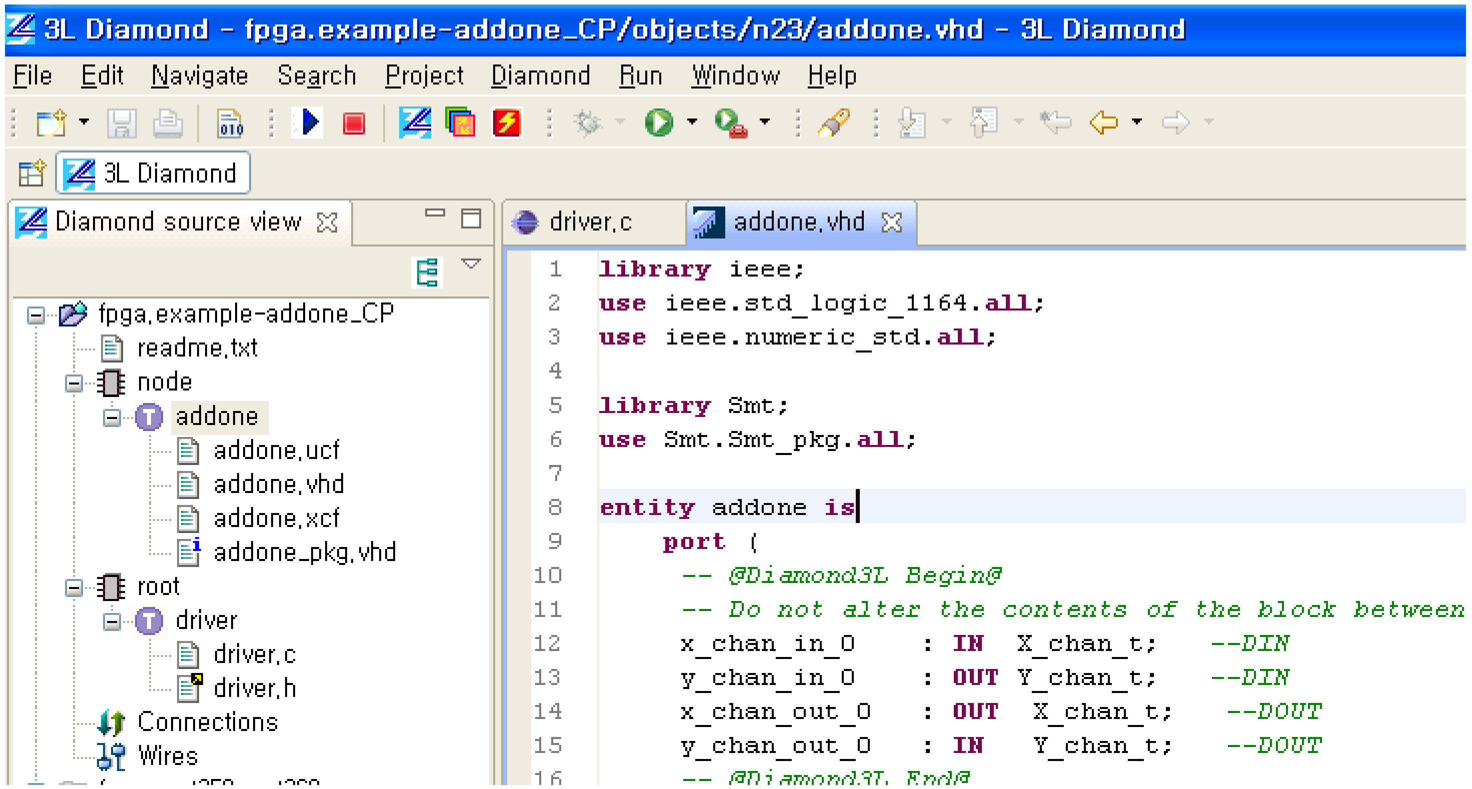
```
! Hardware
processor root Default
processor node FPGA attach = root

wire w0 root[CP:0] node[CP_DEVICE:0]

! Task declarations indicating channel I/O ports and memory requirements
task addone ins = 1 outs = 1 file = "FPGA₩addone₩addone.fcd"
task driver ins = 1 outs = 1

! Set up the connections between the tasks.
connect c1 addone[0] driver[0]
connect c0 driver[0] addone[0]

! Assign software tasks to physical processors
place addone node
place driver root
```

FPGA example 1 – addone (cont.)


Diamond Server: C:₩3L₩Diamond₩...
File Go View Board Help
Reac
SMT310Q [0]

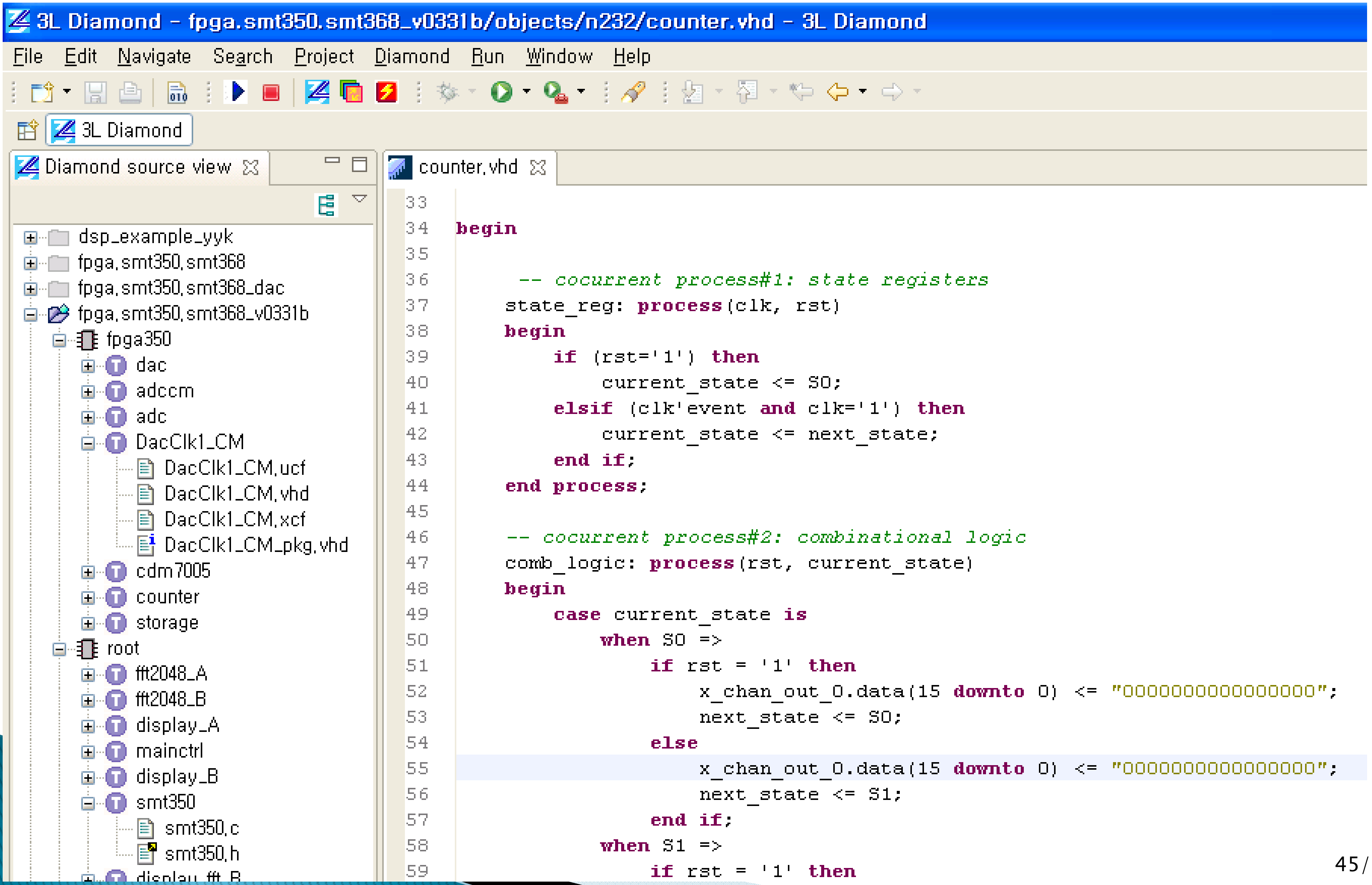
```
3L Diamond FPGA example-1
0 + 1 = 1
1 + 1 = 2
2 + 1 = 3
3 + 1 = 4
4 + 1 = 5
5 + 1 = 6
6 + 1 = 7
7 + 1 = 8
8 + 1 = 9
9 + 1 = 10
10 + 1 = 11
```

IDE example – addone



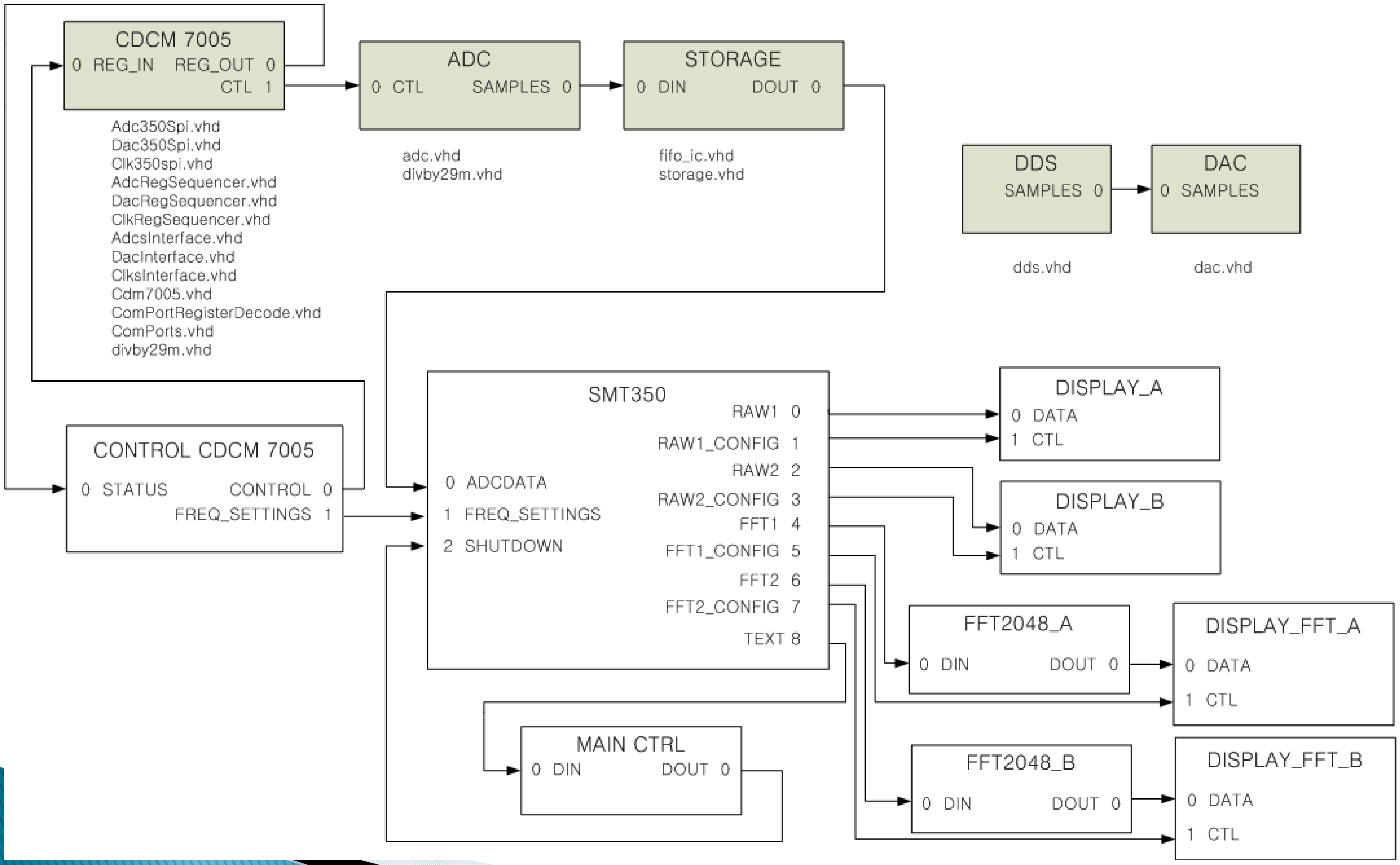
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

library Smt;
use Smt.Smt_pkg.all;

entity addone is
    port (
     -- @Diamond3L Begin@
     -- Do not alter the contents of the block between
     x_chan_in_0    : IN  X_chan_t;   --DIN
     y_chan_in_0    : OUT Y_chan_t;   --DIN
     x_chan_out_0   : OUT  X_chan_t;   --DOUT
     y_chan_out_0   : IN   Y_chan_t;   --DOUT
     -- @Diamond3L End@
```

IDE example – smt350

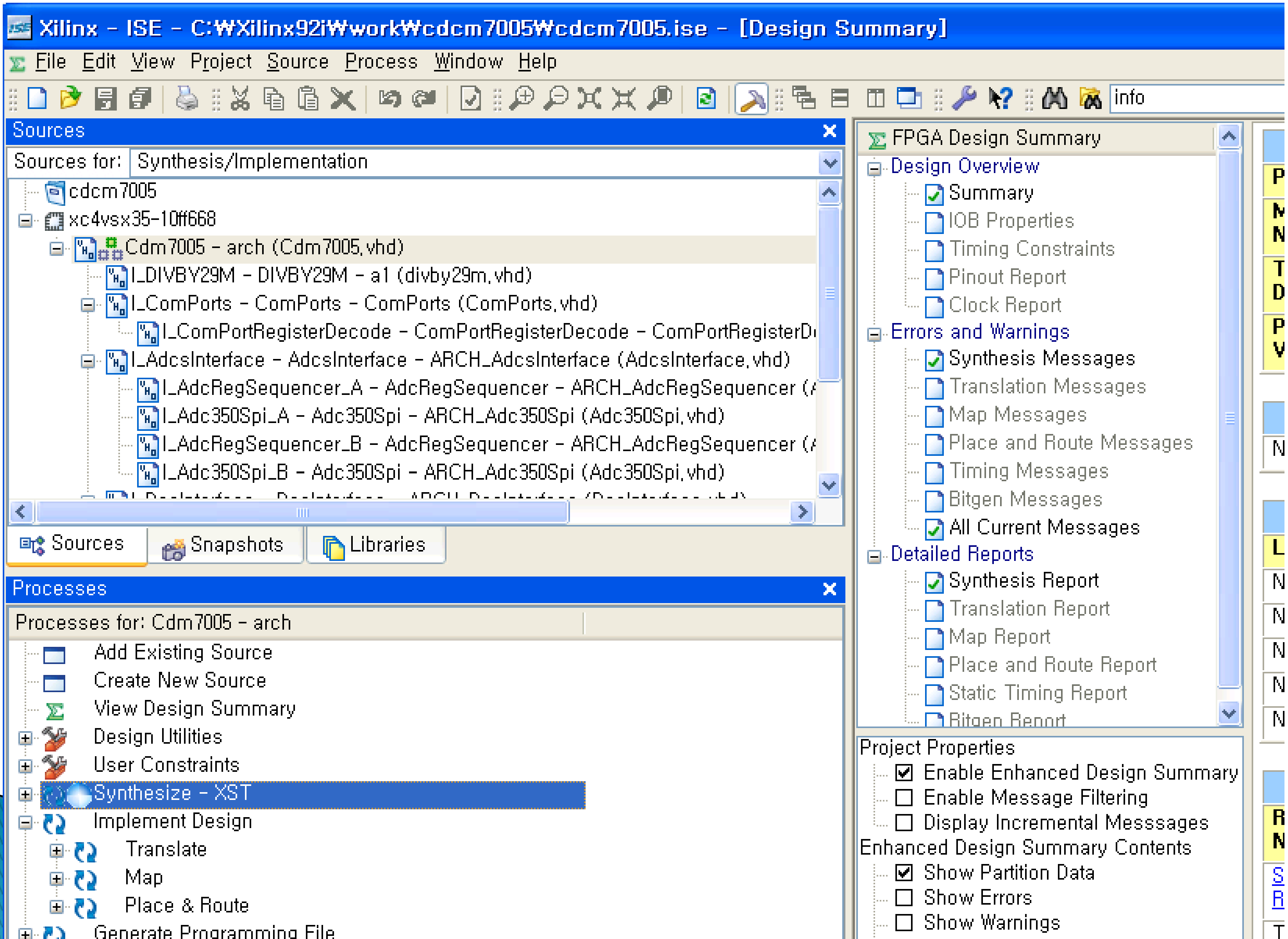


```

begin

     -- cocurrent process#1: state registers
    state_reg: process(clk, rst)
    begin
        if (rst='1') then
            current_state <= S0;
        elsif (clk'event and clk='1') then
            current_state <= next_state;
        end if;
    end process;

    -- cocurrent process#2: combinational logic
    comb_logic: process(rst, current_state)
    begin
        case current_state is
            when S0 =>
                if rst = '1' then
                    x_chan_out_0.data(15 downto 0) <= "0000000000000000";
                    next_state <= S0;
                else
                    x_chan_out_0.data(15 downto 0) <= "0000000000000000";
                    next_state <= S1;
                end if;
            when S1 =>
                if rst = '1' then
```

IDE example – smt350 (cont.)



Xilinx ISE example (cont.)



Xilinx - ISE - C:₩Xilinx92i₩work₩cdcm7005₩cdcm7005.ise - [Design Summary]

File Edit View Project Source Process Window Help

Sources

Sources for: Synthesis/Implementation

```
cdcm7005
xc4vsx35-10ff668
  Cdm7005 - arch (Cdm7005.vhd)
    I_DIVBY29M - DIVBY29M - a1 (divby29m.vhd)
    I_ComPorts - ComPorts - ComPorts (ComPorts.vhd)
      I_ComPortRegisterDecode - ComPortRegisterDecode - ComPortRegisterD
    I_AdcsInterface - AdcsInterface - ARCH_AdcsInterface (AdcsInterface.vhd)
      I_AdcRegSequencer_A - AdcRegSequencer - ARCH_AdcRegSequencer (
      I_Adc350Spi_A - Adc350Spi - ARCH_Adc350Spi (Adc350Spi.vhd)
      I_AdcRegSequencer_B - AdcRegSequencer - ARCH_AdcRegSequencer (
      I_Adc350Spi_B - Adc350Spi - ARCH_Adc350Spi (Adc350Spi.vhd)
```

Sources | Snapshots | Libraries

Processes

Processes for: Cdm7005 - arch

```
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
Implement Design
  Translate
  Map
  Place & Route
Generate Programming File
```

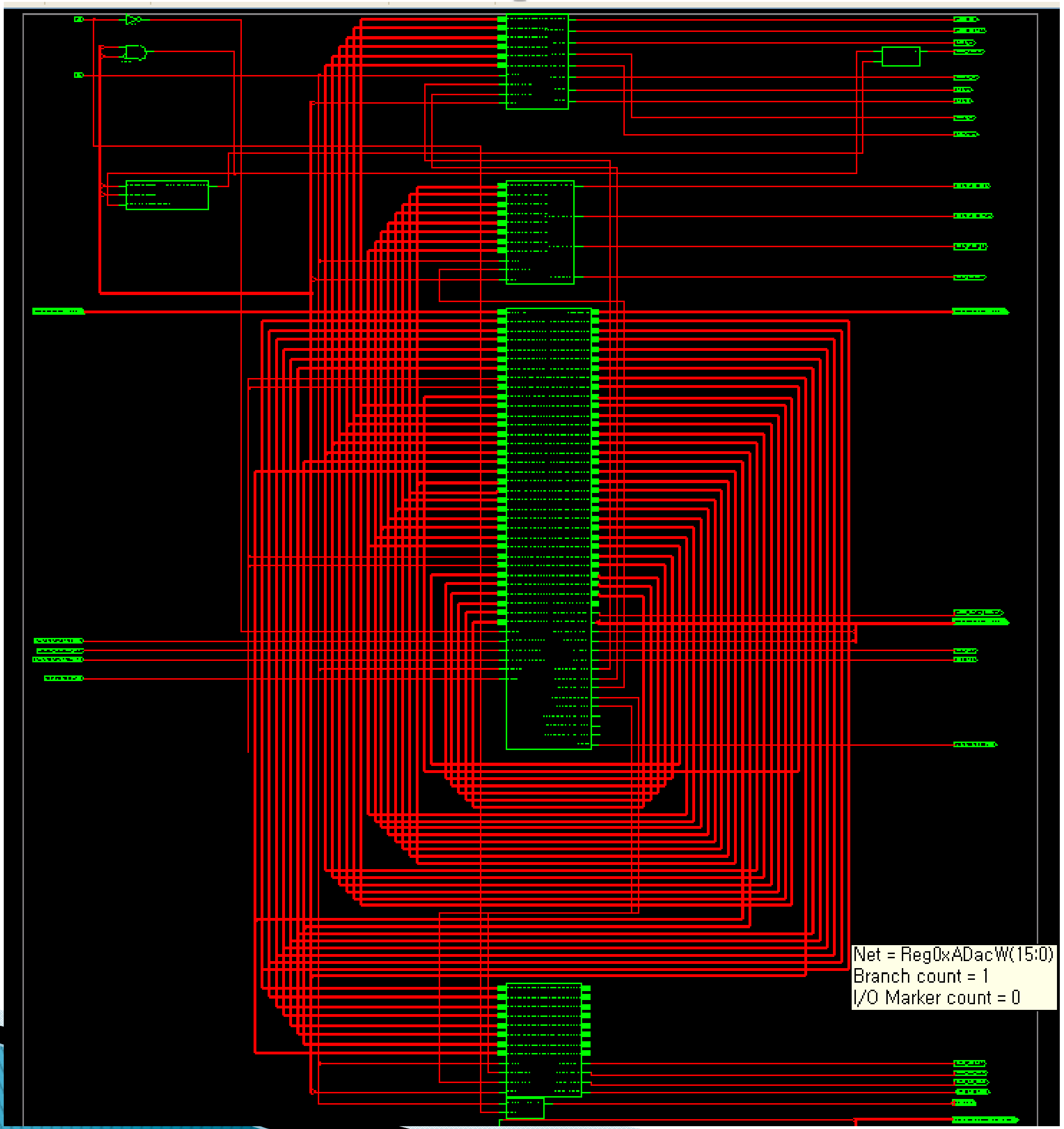
FPGA Design Summary

```
Design Overview
  Summary
  IOB Properties
  Timing Constraints
  Pinout Report
  Clock Report
Errors and Warnings
  Synthesis Messages
  Translation Messages
  Map Messages
  Place and Route Messages
  Timing Messages
  Bitgen Messages
  All Current Messages
Detailed Reports
  Synthesis Report
  Translation Report
  Map Report
  Place and Route Report
  Static Timing Report
  Bitgen Report
```

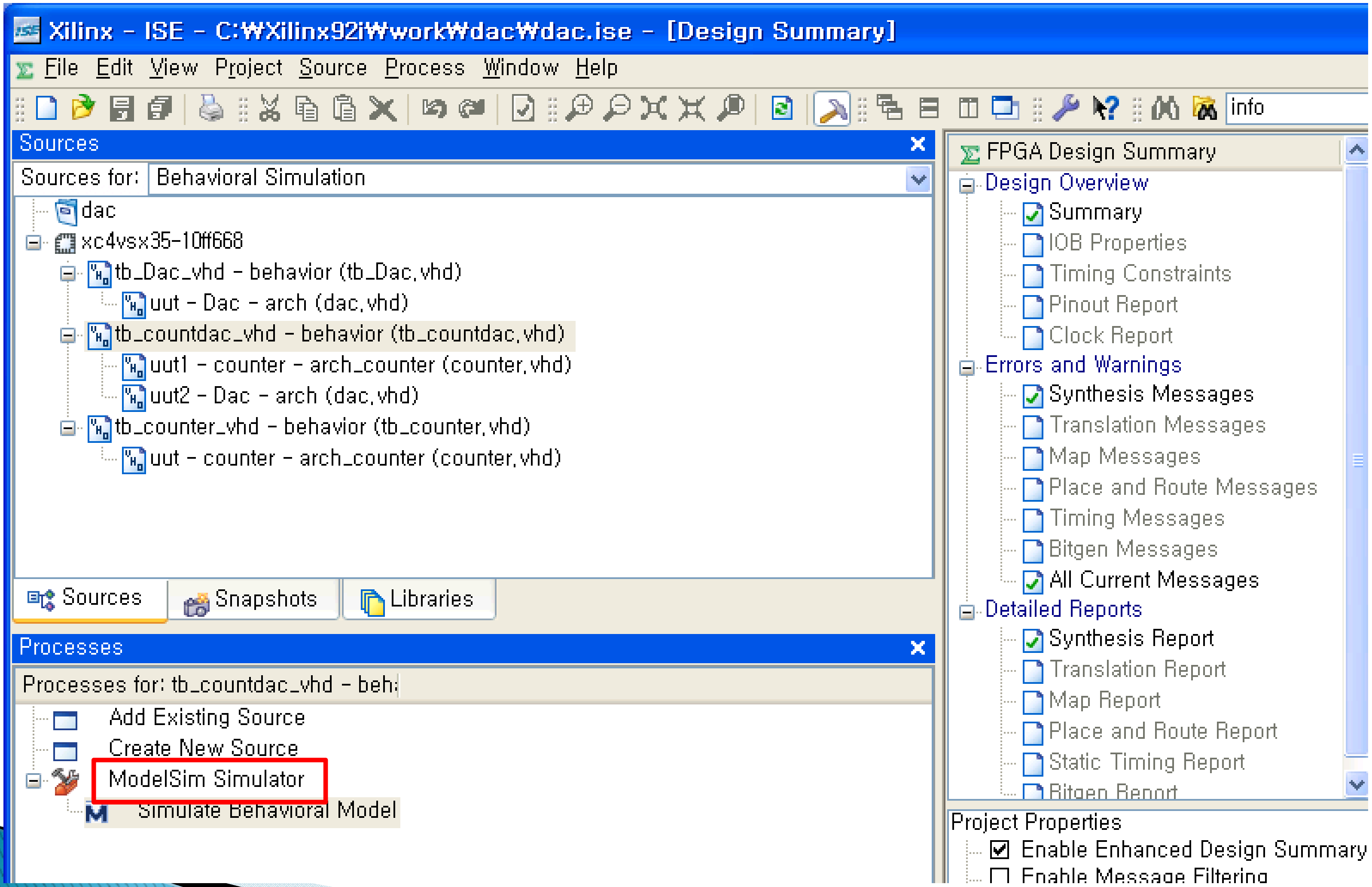
Project Properties
- ☑ Enable Enhanced Design Summary
- ☐ Enable Message Filtering
- ☐ Display Incremental Messsages

Enhanced Design Summary Contents
- ☑ Show Partition Data
- ☐ Show Errors
- ☐ Show Warnings

Xilinx ISE example



Modelsim example


Xilinx - ISE - C:₩Xilinx92i₩work₩dac₩dac.ise - [Design Summary]

Sources for: Behavioral Simulation

- dac
- xc4vsx35-10ff668
  - tb_Dac_vhd - behavior (tb_Dac.vhd)
    - uut - Dac - arch (dac.vhd)
  - tb_countdac_vhd - behavior (tb_countdac.vhd)
    - uut1 - counter - arch_counter (counter.vhd)
    - uut2 - Dac - arch (dac.vhd)
  - tb_counter_vhd - behavior (tb_counter.vhd)
    - uut - counter - arch_counter (counter.vhd)

Processes for: tb_countdac_vhd - beh:

- Add Existing Source
- Create New Source
- ModelSim Simulator
  - Simulate Behavioral Model

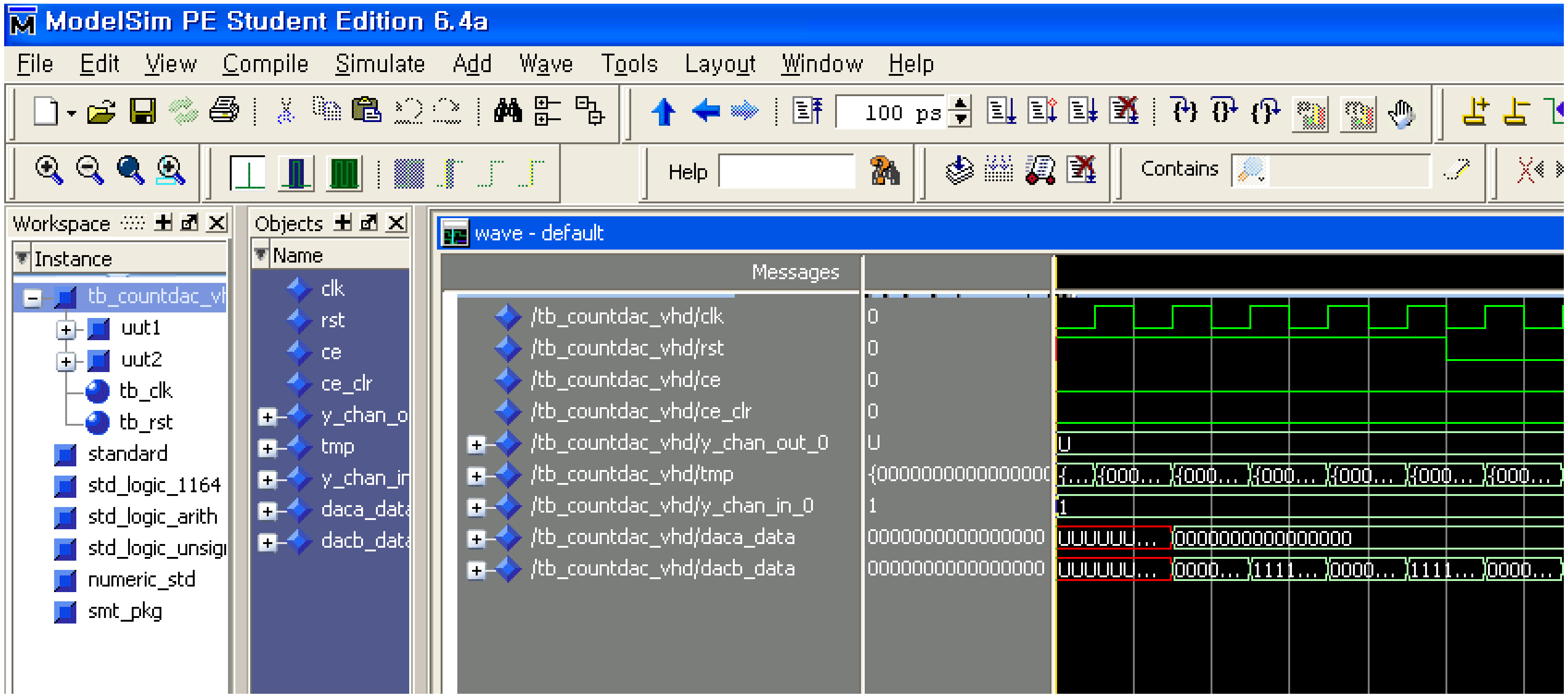
FPGA Design Summary

- Design Overview
  - ✓ Summary
  - IOB Properties
  - Timing Constraints
  - Pinout Report
  - Clock Report
- Errors and Warnings
  - ✓ Synthesis Messages
  - Translation Messages
  - Map Messages
  - Place and Route Messages
  - Timing Messages
  - Bitgen Messages
  - ✓ All Current Messages
- Detailed Reports
  - ✓ Synthesis Report
  - Translation Report
  - Map Report
  - Place and Route Report
  - Static Timing Report
  - Bitgen Report

Project Properties

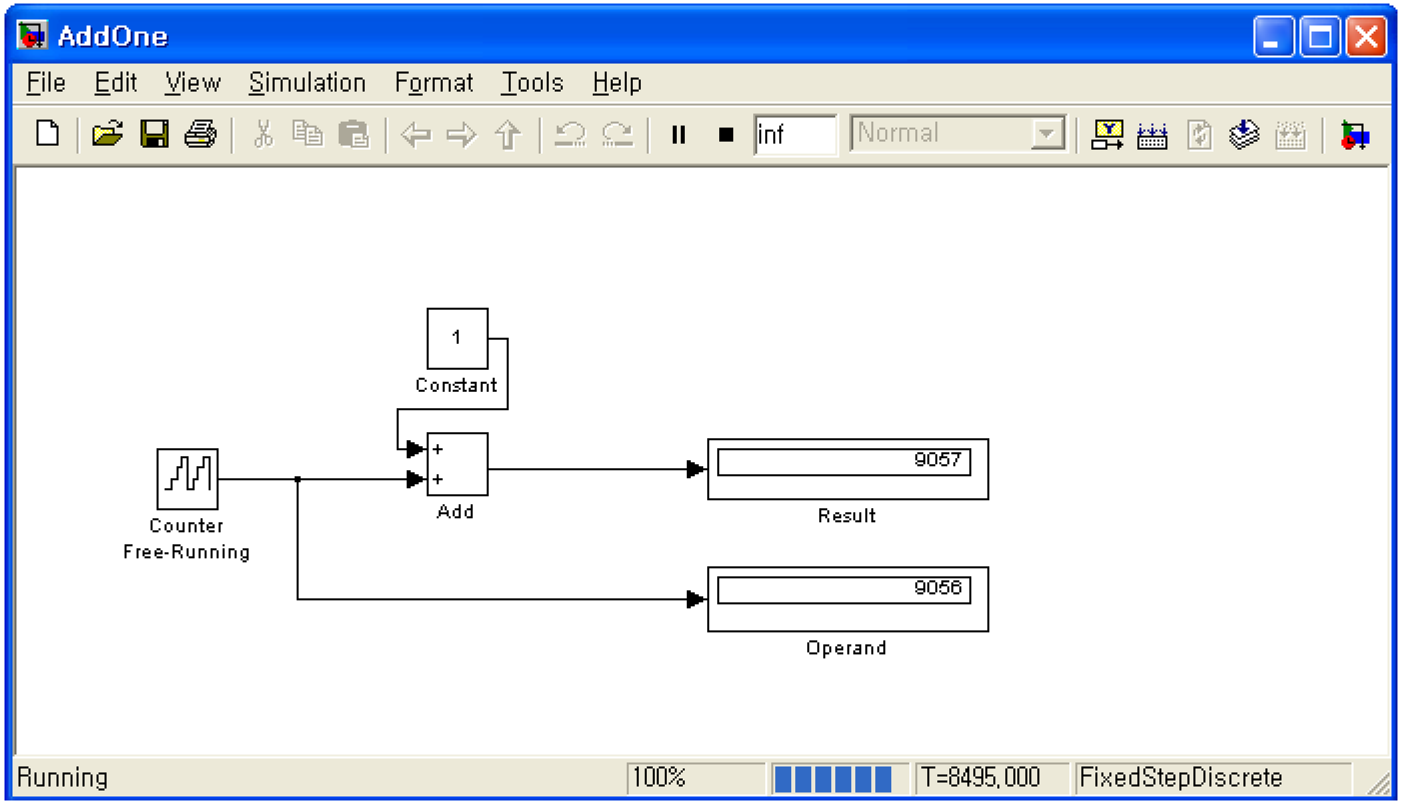
- [✓] Enable Enhanced Design Summary
- [ ] Enable Message Filtering

Modelsim example (cont.)



```
/tb_countdac_vhd/clk           0
/tb_countdac_vhd/rst           0
/tb_countdac_vhd/ce            0
/tb_countdac_vhd/ce_clr        0
/tb_countdac_vhd/y_chan_out_0  U                   U
/tb_countdac_vhd/tmp           {0000000000000000   {...}{000...}{000...}{000...}{000...}{000...}{000...}
/tb_countdac_vhd/y_chan_in_0   1                   1
/tb_countdac_vhd/daca_data     0000000000000000    UUUUUU... 0000000000000000
/tb_countdac_vhd/dacb_data     0000000000000000    UUUUUU... 0000... 1111... 0000... 1111... 0000...
```

PARS example – add one



Future works

- ITRC Forum 전시 예정 (2010.05.25)
  - Full-Duplex AF Relay 구현
  - Relay: EM-level self-interference cancellation
  - Destination: real-time or image processing
- IDE 예제 코드 완벽 분석
- PARS programming
  - FPGA module
  - ADC/DAC module

References

- http://www.sundance.com/
- http://support.sundance.com