Advanced VLSI
SOPC design flow

Advisor: 吳安宇
Speaker: 沈文中
Outline

- What’s SOC?
- IP classification
- IP reusable & benefit
- SOPC solution on FPGA
- SOPC design flow
Outline

- What’s SOC?
  - Definition of SOC
  - Advantage of SOC

- IP classification
- IP reusable & benefit
- SOPC solution on FPGA
- SOPC design flow
System-on-a-Chip

1. Use standard bus, to transfer data among blocks.
2. For bus IO, block reusable is benefit.
3. IO pad of chip is the limit of freq. But freq. in the same chip can be higher.
Changes in the Nature of IC Design

- Handcrafted custom chips
- Transistor models
- Cell array
- Library-based chip
- Chip-design tools
- System on a chip
- System-design tools
- Block design

Design efficiency, gates per person-day
Ex: JPEG video system
SOC製程演進階段

第一階段

第二階段

第三階段

Logic

ROM

SRAM

uP Core

ROM

DSP Core

SRAM

PLL

Logic

AD/DA

Soft I/F Core

uP Core

ROM

DSP Core

SRAM

PLL

Application Specific IP Core

AD/DA

Logic

Soft I/F Core

Logic

Analog

2/26/2001 台大SOC課程
System-level IC Development

**System-Level IC Architecture**
- System Architecture
- Chip Architecture
- Technology Selection
- Algorithm Development

**IP Sourcing**
- In-house IP
- 3rd party IP - Selection - Qualification - Licensing

**IP Integration**
- Digital logic +
- Mixed-signal
- Embedded Memory
- Embedded Micro's

**Chip Implementation**
- FPGA
- Gate array
- Standard cells
- Megacell library
- Datapath compiler
- Memory compiler +
- Hand-crafted
- In-house tools

**Chip Fabrication**
- 3rd party foundry services
Outline

- What’s SOC?
- IP classification
  - Soft IP
  - Firm IP
  - Hard IP
- IP reusable & benefit
- SOPC solution on FPGA
- SOPC design flow
Types of IP

1. Soft IP: ("Code")
   1. Synthesizable HDL description at RTL level
   2. Flexible: can be changed to suit an application
   3. Technology independent: may be re-synthesized across process
   4. Significant IP protection risk

```verilog
module wtm(Z,A,B);
output [15:0] Z;
input [7:0] A,B;
wire [7:0] w0,w1,w2,w3,w4,w5,w6,w7,
carry1,sum1,cary2,sum2;
wire carry,sum,ctrl1;
wire [45:0] mcarry,msum;
wire [2:0] carry3,sum3,cary4,sum4;
wire [5:0] z1,z0;

andx7  add0(w0[6:0],A[0],B[6:0]),
add1(w1[6:0],A[1],B[6:0]);
```
Types of IP

1. Firm IP: (“synthesizable netlist + structure”)
   1. Gate-level netlist optimized structurally and topologically for performance and size
   2. Floorplanning or placement without routing
3. Hard IP: ("physical")
   1. Ready for “drop in”
   2. Include layout and timing information
   3. Optimized for performance, size, and power
   4. IP is easily protected
## Categorizing Reusable Blocks

<table>
<thead>
<tr>
<th>Block Type</th>
<th>Flexibility vs. predictability</th>
<th>Design flow</th>
<th>Representation</th>
<th>Libraries</th>
<th>Process technology</th>
<th>Portability</th>
<th>Protecting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft</td>
<td>Very flexible, unpredictable</td>
<td>System design</td>
<td>Behavioral</td>
<td>Not applicable</td>
<td>Independent</td>
<td>Unlimited</td>
<td>Week</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTL design</td>
<td>RTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firm</td>
<td>Flexible, predictable</td>
<td>Floorplanning synthesis</td>
<td>RTL, blocks</td>
<td>Reference</td>
<td>Generic</td>
<td>Library mapping</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Placement</td>
<td>Netlist</td>
<td>Footprint, timing model, wiring model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hard</td>
<td>Inflexible, very predictable</td>
<td>Routing, verification</td>
<td>Polygon data</td>
<td>Process-specific library and design rules</td>
<td>Fixed</td>
<td>Process mapping</td>
<td>Good</td>
</tr>
</tbody>
</table>
Tradeoffs among types of blocks

- **Soft**
- **Firm**
- **Hard**

**Flexibility**

**Predictability, performance, and complexity**

High

Low
Outline

- What’s SOC?
- IP classification
- IP reusable & benefit
- SOPC solution on FPGA
- SOPC design flow
Engineering Productivity Gap

1. Engineering productivity has not been keeping up with silicon gate capacity for several years.

2. Companies have been using larger design teams, making engineers work longer hours, etc., but clearly the limit is being reached.
Why must IP Reuse?

Design productivity crisis:
Divergence of potential design complexity and designer productivity
Outline

- What’s SOC?
- IP classification
- IP reusable & benefit
- SOPC solution on FPGA
- SOPC design flow
What is SOPC?

Complete SOPC Solution (System-On-a-Programmable-Chip)
Introduce to SOPC solution

- Soft embedded processor design & prototyping environment
Introduce to SOPC solution

- Hard embedded processor design & prototyping environment
Excalibur Nios Device Support

Future PLD Architectures

Excalibur MIPS
Excalibur ARM

Mercury™ Devices
APEX™ Devices
FLEX™ 10K Devices
ACEX™ Devices

Performance (MIPS)
Flexibility & Scalability

Low-Cost Embedded Processor

High-Performance Custom DSP

Multi-Processor System

ACEX™ EP1K100

75K Gates Available

Nios

APEG EP20K200E

150K Gates Available

Nios DSP

Nios ESB

Nios ESB

Nios ESB

Nios ESB

Nios ESB

Nios ESB

Nios ESB

Excalibur ARM Stripe

Excalibur XA10

500K Gates Available
Nios Development Kit

- Nios 32-Bit and 16-Bit RISC CPU
- Peripherals
- Development Board
- Development Tools
Nios Embedded Processor Core

- Configurable Soft Core Processor
- 32-Bit Pipelined RISC Architecture
- Large Internal Register File
- Fully synchronous interface
- Configurable Data Path
- 30 to 80 MIPS Performance
- Dynamic Bus Sizing

Your Design Here

12% of EP20K200E

Alterra PLD
Outline

- What’s SOC?
- IP classification
- IP reusable & benefit
- SOPC solution on FPGA
- SOPC design flow
Nios System Overview

- A complete Nios system module contains a Nios embedded processor and its associated system peripherals.

- The SOPC Builder helps you easily specify options for the Nios system module.

- Nios system module contains hardware and software sections.
Nios System Overview (cont.)

- Peripheral components
  - Memory Interface
    - RAM, ROM
  - Serial I/O
    - UART
  - PIO (parallel IO)
    - Seven Segment
    - LED, LCD
    - User defined Interface
  - Timer
Nios System Overview (cont.)

- The SOPC builder let NIOS module easy specified by using wizard

- Almost constraint can be changed to fit our own design, such as peripheral library.
Nios System Overview (cont.)

- Hardware & Software generation

- Nios system module contains:
  - Hardware: (HDL code) describe the hardware module - download by QuartsII
  - Software: (header file) define c program library to develop your own program - download by NIOS SDK
Tool Design Flow

Configure Processor

Select Peripherals

Generate

Peripheral Library
- Processor Design
- Simulation Test Bench

Hardware
- Synthesis
  - Place & Route
- User Design
- Purchased IP

Software
- Cygnus/Red Hat
  - GNUPro
- Users Code
- S/W Libraries
- RTOS

JTAG
- Serial
  - Hardware Configuration File

Download & Debug
- Altera PLD

Download & Debug
Steps

- Create your design
  - Create project file
  - Create *.BDF file
- Add SOPC into your design
- Connect your own block & IO pad to Nios CPU
- Compile & Pin Assignment
- Software development
- Download your design
Create a new project

- Select File => New Project Wizard
  - Working Directory
  - Project name
  - Top-level design name

- Create a new project in the path
  C:\AVLSI\<your id>\lab2

Click finish
Create a new schematic

- Create a *.bdf file, for placing SOPC system, include NIOS CPU & other block

Block design file
Steps

- Create your design
- Add SOPC into your design
  - Create CPU
  - Add peripherals
  - Generate hardware & software design
- Connect your own block & IO pad to Nios CPU
- Compile & Pin Assignment
- Software development
- Download your design
Create NIOS32 CPU

- Select symbol from tool menu bar,
- And click Mega Wizard Plug-In Manager
Create megafunction

- SOPC is one block of megafunction

- Create a new megafunction, to familiar with its content
Use SOPC Builder

- Select **Altera SOPC Builder 2.5** and
- Choose **Verilog HDL** to perform the soft core CPU
Add Components

- Double click the component list for adding it into your design
Current List of Peripherals

- UART
- PIO
- Timer
- SPI
- PWM
- IDE
- Keyboard
- PS2 Mouse
- VGA
- Ethernet
- PCI
- SRAM
- SDRAM
- Flash
- Compact Flash
- On-chip RAM
- On-chip ROM
- User-Defined
PIO Peripheral

- 1 to 32-bit Parallel I/O Port
  - Input Only
  - Output Only
  - Bi-directional Port
- Edge Detection on Inputs
- Interrupt Generation
  - Mask-able
  - IRQ Source
User-Defined Interface

- Interface to Other Peripherals
- Configures Busses and Timing
- Adds Port Signals to Design
Addon Component

- Peripheral components
  - Memory Interface
    - RAM, ROM, flash
  - Serial I/O
    - UART
  - PIO(parallel IO)
    - Seven Segment
    - LED, LCD
    - User defined Interface
  - Timer

Diagram:
- Nios CPU
- Altera PLD
- Avalon™ Bus
- External Device
- I/O
- User I/F
The demo board
Add CPU

- Choose 32 bit RISC CPU, then click “Finish”

Click “finish”
Steps

- Create your design
- Add SOPC into your design
  - Create CPU
  - Add peripherals
    - Generate hardware & software design
- Connect your own block & IO pad to Nios CPU
- Compile & Pin Assignment
- Software development
- Download your design
Choose On-chip Memory

Attributes: ROM (read-only)

Contents: ROM is blank and read only.
Monitor Program Runs from On-Chip ROM
Debugger Runs on Host PC or UNIX Platform
Basic Development Facilities:
  Download Code
  Burn Flash
  Examine/Modify Memory
  Run Programs
GERMS

- G – Go
- E – Erase Flash
- R – Relocate
- M – Memory
- S – Motorola S record
- : – Intel Hex record
- CR – show next 64 Bytes of memory
- Escape – Restart GERMS monitor
Flash memory setting

Simulate by Quarts II
Steps

- Create your design
- Add SOPC into your design
  - Create CPU
  - Add peripherals
  - Generate hardware & software design
- Connect your own block & IO pad to Nios CPU
- Compile & Pin Assignment
- Software development
- Download your design
SOPC Setup

Choose **verilog** and **APEX 20KE** in HDL generation
Generate System Module

- Software Development Kit (SDK)
- Hardware Design
- Synthesis
What happen after Generate

- **Software**
  - `nios.h` : define the peripheral address
  - : define the memory map

- **Hardware**
  - A NIOS CPU module constructed by verilog code
Software Generation

IO have their addr.

/*
 * File: nios.h
 *
 * This file is a machine generated address map
 * for a Nios cpu named cpu.
 * f:我的文件/avlsi/r91943088/lab2/nios32.ptf
 */

// The Memory Map

#define na_uart_1_debug         ((np_uart *) 0x00000420)
#define na_uart_1_debug_irq                   17
#define na_timer_0              ((np_timer *) 0x00000440)
#define na_timer_0_irq                        18
#define na_led_pio              ((np_pio *)   0x00000460)
#define na_button_pio           ((np_pio *)   0x00000470)
#define na_button_pio_irq                     19
#define na_ext_ram              ((void *)     0x00040000)
#define na_ext_ram_end          ((void *)     0x00080000)
#define na_ext_ram_size         ((void *)     0x00040000)
#define na_ext_flash            ((void *)     0x00100000)
Hardware Generation

- After generation, a NIOS module appears in Symbol.
Steps

- Create your design
- Add SOPC into your design
- Connect your own block & IO pad to Nios CPU
- Compile & Pin Assignment
- Software development
- Download your design
Add IO Pad

- Use add symbol to add IO pad to schematic.
Steps

- Create your design
- Add SOPC into your design
- Connect your own block & IO pad to Nios CPU

✓ Pin Assignment & Compile

- Download your design
Assign Pins

- Compile first

- Assign the IO pins to the I/O Interface on develop board

- Compile then
Steps

- Create your design
- Add SOPC into your design
- Connect your own block & IO pad to Nios CPU
- Compile & Pin Assignment
- **Software development**
- Download your design
Software Develop

- CPU contains **hardware** part and **software** part

- **Hardware** part was created above

- **Software** part is written in **c or c++** language, then compiled into **assembly code**

- Take an example for writing **c++ program** to control CPU change light when press switch
How to write code

#define SW7PRESSED 0xff7

int main(void)
{
    int buttons;
    int led = 0;

    np_pio *buttonpio = na_button_pio;
    np_pio *ledpio = na_led_pio;

#if NIOS_GDB
    nios_gdb_install(1);
    nios_gdb_breakpoint();
#endif

    buttonpio->np_piodirection = 0;
    ledpio->np_piodirection = 1;

Define memory map & peripheral addr.

0111

button

led

0: input
1: output
2: bidirection
How to write code (cont.)

```c
while(1)
{
    buttons = buttonpio->np_piodata;
    buttons = buttons | 0xff0;
    if (buttons == SW7PRESSED)
    {
        if (led == 0) // toggle LED 0
            led = 1;
        else
            led = 0;
        ledpio->np_piodata = led;
        while (buttons == SW7PRESSED)
            buttons = buttonpio->np_piodata | 0xff0;
        nr_delay (30); // debounce delay
    }
    // That's all folks
}
```
Steps

- Create your design
- Add SOPC into your design
- Connect your own block & IO pad to Nios CPU
- Compile & Pin Assignment
- Software development

✓ Download your design
Memory Usage

- *.srec is downloaded into SRAM, can’t still exist after reset

- Flash memory store the data executed automatically when boot
NIOS SDK

- UNIX like environment

- Open from "Start Menu/program files/Altera/Excalibur NIOS 2.0/NIOS SDK Shell"
NIOS Command

- **nios-build**: Compile, Assemble & Link
  - Transpose c program into `<program>.srec`

- **nios-run**: Download Executable & Run
  - burn flash with `<project>.hexout` (hardware) or `<program>.flash` (software)
  - Download *.srec to SRAM

- **srec2flash**: Create Flash-bootable Code
  - Convert `<program>.srec` to `<program>.flash`
Nios Routines

- nios-build hello.c (nb hello.c)
  - Builds program including compile, link and convert to SREC
- nios-run hello.srec (nr hello.srec)
  - Downloads using com1 and enters terminal mode
- nr –p com2 hello.srec
  - Downloads using com2
- nr –x hello.srec
  - Downloads without entering terminal mode
- nr –t
  - Enters terminal mode without download (Ctrl + C to exit)
Download routine

- nr <project>.hexout -download hardware design
- nr <program>.srec -download software design
LAB

- NIOS Tutorial
- Have a C code to generate sine wave and display to 7-segment LED. Program should read data from DIP SW as a value of delay loop. (pi / 10)
Homework

- Use Mega Plug-In Manager to generate NIOS with DIP SW and necessary peripheral device.
- Use Mega Plug-In Manager to generate a 4 by 4 multiplier.
- Have a verilog code to decode 8 bit binary to 7-segment.
- Have a C code to read two 4-bit data from DIP SW and send data to 4 by 4 multiplier. The multiplier output should connect to decoder directly to display answer.
- Use LA to read answer-data to verify, too.