Lab3: Core Peripherals

Speaker: Chao-Chung Cheng
Directed by Prof. Tian-Sheuan Chang
March, 2004, NCTU
Goal of This Lab

- Familiarize with ARM Hardware Development Environment
  - ARM Integrator/AP
  - Core Module
  - Logic Module
- How to use Timer/Interrupt
Outline

- **ARM Integrator Core Module (CM) [1]**
- ARM Integrator Logic Module (LM) [2]
- ARM Integrator ASIC Application Platform (AP) [3]
- System Memory Map [1]
- Semihosting
- Lab3 – Core Peripheral
ARM Integrator Core Module/CM

- CM provides **ARM core** personality.
- CM could be used as a standalone development system without AP.
- CM could be mounted onto AP as a system core.
- CM could be integrated into a 3rd-party development or ASIC prototyping system.
Core module
ARM Integrator/CM Feature (CM9TDMI)

- ARM9TDMI microprocessor core
  - ARM940T/ARM920T

- Core module controller FPGA:
  - SDRAM controller
  - System bus bridge
  - Reset controller
  - Interrupt controller

- Supports 16MB~256MB PC66/PC100 168pin SDRAM
- Supports 256/512 KB SSRAM
- Multi-ICE, logic analyzer, and optional trace connectors.
FPGA functional diagram

- SSRAM
- Clock generator
- Status/control registers
- Reset controller
- SDRAM controller
- SDRAM
- ARM core
- System bus bridge
- System bus
- Multi-ICE
- System bus connectors HDRA/HDRB
- FPGA
ARM Integrator Core Module FPGA

- **SDRAM controller**
  - Supports for DIMMs from 16MB to 256MB.

- **Reset controller**
  - Initializes the core.
  - Process resets from different sources.

- **Status and configuration space**
  - Provides processor information.
  - CM oscillator setup.
  - Interrupt control for the processor debug communications channel.

- **System bus bridge**
  - Provides Interface between the memory bus on the CM and the system bus on the AP.
Connecting Multi-ICE with CM

SOC Consortium Course Material
Outline

- ARM Integrator Core Module (CM) [1]
- ARM Integrator Logic Module (LM) [2]
- ARM Integrator ASIC Application Platform (AP) [3]
- System Memory Map [1]
- Semihosting
- Lab3 – Core Peripheral
ARM Integrator/LM Logic Module

- LM is designed as a platform for development AHB/ASB/APB peripherals for use with ARM cores.
- LM could be used as a standalone system.
- LM could be mounted with an Integrator/CM, and an Integrator/AP motherboard.
- LM could be used as a CM with Integrator/AP if a synthesized ARM core, such as ARM9TDMI-S, is programmed into the FPGA.
ARM Integrator/LM Feature (XCV-2000E)

- Altera or Xilinx FPGA
- Configuration PLD and flash memory for storing FPGA configurations
- 1MB SSRAM
- Clock generators and reset resources
- Switches
- LEDs
- Prototyping grid
- JTAG, Trace, and logic analyzer connectors
- System bus connectors to a motherboard or other modules
Using Multi-ICE with LM
Outline

☐ ARM Integrator Core Module (CM) [1]
☐ ARM Integrator Logic Module (LM) [2]
☐ *ARM Integrator ASIC Application Platform (AP)* [3]
☐ System Memory Map [1]
☐ Semihosting
☐ Lab3 – Core Peripheral
About ARM Integrator/AP

- An ATX motherboard which can be used to support the development of applications and hardware with ARM processor.
- Platform board provides the AMBA backbone and system infrastructure required.
- Core Modules (CM) & Logic Modules (LM) could be attached to ASIC Platform.
ARM Integrator/AP Features

- System controller FPGA.
  - System bus to CMs and LMs
  - System bus arbiter
  - Interrupt controller
  - Peripheral I/O controller
  - 3 counter/timers
  - Reset controller
  - System status and control registers
- Clock Generator
- Two serial ports (RS232 DTE)
- PCI bus interface supporting onboard expansion.
- *External Bus Interface* (EBI) supporting external memory expansion.
- 256KB boot ROM
- 32MB flash memory.
- 512K SSRAM.
Integrator/AP

Not to scale

SOC Consortium Course Material 18
ARM Integrator/AP Block Diagram
Assembled Integrator Development System

- Core module
- Logic module
Assembled Integrator/AP system

SOC Consortium Course Material
System Controller FPGA (1/2)

- **System Bus Interface**
  - Supports transfers between system bus and the *Advanced Peripheral Bus* (APB).
  - Supports transfers between system bus and the PCI bus.
  - Supports transfers between system bus and the *External Bus Interface* (EBI).

- **System Bus Arbiter**
  - Provides arbitration for a total of 6 bus masters.
    - Up to 5 masters on CMs or LMs.
    - PCI bus bridge. (the highest priority)

- **Interrupt Controller**
  - Handles IRQs and FIQs for up to 4 ARM processors.
  - IRQs and FIQs originate from the peripheral controllers, PCI bus, and other devices on LMs.
Peripheral I/O Controllers
- 2 ARM PrimeCell UARTs
- ARM PrimeCell Keyboard & Mouse Interface (KMI)
- ARM PrimeCell Real Time Clock (RTC)
- 3 16-bit counter/timers
- GPIO controller
- Alphanumeric display and LED control, and switch reader

Reset Controller
- Initializes the Integrator/AP when the system is reset

System Status & Control Register
- Clock speeds
- Software reset
- Flash memory write protection
System Controller FPGA Diagram
Reset Controller

- A reset controller is incorporated into the system controller FPGA.

- The hardware reset sources are as follows:
  - Push-button **PBRST** and CompactPCI signal **CP_PRST**
  - ATX power OK signal **nPW_OK** and CompactPCI power fail signal **CP_FAL**
  - **FPGADONE** signal (routed through CPCI arbiter to become **nRSTSRC5**)
  - Logic modules using **nEXPRST**
  - Core modules (and Multi-ICE) using **nSRST**
Integrator/AP Reset Control
Interrupt Controller

- The system controller FPGA contains four interrupt controllers.

- The system controller incorporates a separate IRQ and FIQ controller for each core module.

- Interrupts are masked enabled, acknowledged, or cleared via registers in the interrupt controller.

- Main sources of interrupts:
  - System controller’s internal peripherals
  - LM’s devices
  - PCI subsystem
  - Software
Interrupt Controller Architecture
System Bus

- The **HDRA/HDRB** and **EXPA/EXPB** connector pairs are used to connect the system bus between the AP and other modules
  - Core modules on the connectors HDRA and HDRB
  - Logic modules on the connectors EXPA and EXPB

- There are three main system bus (A[31:0], C[31:0], and D[31:0]) and fourth bus B[31:0]
  - A[31:0]: This is the address bus
  - B[31:0]: Only connects HDRA to EXPA and reserved for future use
  - C[31:0]: Used to implement a system control bus
  - D[31:0]: This is the data bus
Peripherals

- The peripheral devices incorporated into the system controller FPGA
  - Counter/timers
  - Real-time clock
  - UARTs
  - Keyboard and mouse interface
  - GPIO
Counter/Timers

- There are 3 counter/timers on an ARM Integrator AP.
- Each counter/timer generates an IRQ when it reaches 0.
- Each counter/timer has:
  - A 16-bit down counter with selectable prescale
  - A load register
  - A control register
These registers control the 3 counter/timers on the Integrator AP board.

Each timer has the following registers.

- **TIMERX_LOAD**: a 16-bit R/W register which is the initial value in free running mode, or reloads each time the counter value reaches 0 in periodic mode.
- **TIMERX_VALUE**: a 16-bit R register which contains the current value of the timer.
- **TIMERX_CTRL**: an 8-bit R/W register that controls the associated counter/timer operations.
- **TIMERX_CLR**: a write only location which clears the timer’s interrupt.
## Counter Timer Registers

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x13000000</td>
<td>TIMER0_LOAD</td>
<td>R/W</td>
<td>16</td>
<td>Timer0 load register</td>
</tr>
<tr>
<td>0x13000004</td>
<td>TIMER0_VALUE</td>
<td>R</td>
<td>16</td>
<td>Timer0 current value reg</td>
</tr>
<tr>
<td>0x13000008</td>
<td>TIMER0_CTRL</td>
<td>R/W</td>
<td>8</td>
<td>Timer0 control register</td>
</tr>
<tr>
<td>0x1300000C</td>
<td>TIMER0_CLR</td>
<td>W</td>
<td>1</td>
<td>Timer0 clear register</td>
</tr>
</tbody>
</table>

## Timer Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>ENABLE</td>
<td>Timer enable: 0=disable; 1=enable.</td>
</tr>
<tr>
<td>6</td>
<td>MODE</td>
<td>Timer mode: 0=free running; 1=periodic</td>
</tr>
<tr>
<td>5:4</td>
<td>unused</td>
<td>Unused, always 0</td>
</tr>
<tr>
<td>3:2</td>
<td>PRESCALE</td>
<td>Prescale divisor: 00=none; 01 = div by 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10=div by 256; 11 = undefined</td>
</tr>
<tr>
<td>1:0</td>
<td>Unused</td>
<td>Unused, always 0</td>
</tr>
</tbody>
</table>
The IRQ and FIQ Control Registers

- Implemented in the system controller FPGA.
- Provides interrupt handling for up to 4 processors.
- There’s a 22-bit IRQ and FIQ controller for each processor.
The registers control each processor’s interrupt handler on the Integrator AP board.

Each IRQ has following registers:

- **IRQX_STATUS**: a 22-bit register representing the current masked IRQ status.
- **IRQX_RAWSTAT**: a 22-bit register representing the raw IRQ status.
- **IRQX_ENABLESET**: a 22-bit location used to set bits in the enable register.
- **IRQX_ENABLECLR**: a 22-bit location used to clear bits in the enable register.
IRQ Registers (2/2)

- **IRQ Registers**

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x14000000</td>
<td>IRQ0_STATUS</td>
<td>R</td>
<td>22</td>
<td>IRQ0 status</td>
</tr>
<tr>
<td>0x14000004</td>
<td>IRQ0_RAWSTAT</td>
<td>R</td>
<td>22</td>
<td>IRQ0 IRQ status</td>
</tr>
<tr>
<td>0x14000008</td>
<td>IRQ0_ENABLESET</td>
<td>R/W</td>
<td>22</td>
<td>IRQ0 enable set</td>
</tr>
<tr>
<td>0x1400000C</td>
<td>IRQ0_ENABLECLR</td>
<td>W</td>
<td>22</td>
<td>IRQ0 enable clear</td>
</tr>
</tbody>
</table>

- **IRQ Registers bit assignments**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SOFTINT</td>
<td>Software interrupt</td>
</tr>
<tr>
<td>5</td>
<td>TIMERINT0</td>
<td>Counter/Timer interrupt</td>
</tr>
<tr>
<td>6</td>
<td>TIMERINT1</td>
<td>Counter/Timer interrupt</td>
</tr>
<tr>
<td>7</td>
<td>TIMERINT2</td>
<td>Counter/Timer interrupt</td>
</tr>
</tbody>
</table>
Outline

- ARM Integrator Core Module (CM) [1]
- ARM Integrator Logic Module (LM) [2]
- ARM Integrator ASIC Application Platform (AP) [3]
- System Memory Map [1]
- Semihosting
- Lab3 – Core Peripheral
System Memory Map
Core Module Memory Map

Standalone vs Attached to a motherboard

<table>
<thead>
<tr>
<th>Memory Location</th>
<th>Standalone</th>
<th>Attached to motherboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x11000000</td>
<td>Abort</td>
<td>Motherboard</td>
</tr>
<tr>
<td>0x10800000</td>
<td>SSRAM alias</td>
<td>SSRAM alias</td>
</tr>
<tr>
<td>0x10000000</td>
<td>CM registers</td>
<td>CM registers</td>
</tr>
<tr>
<td>0x00040000</td>
<td>SDRAM</td>
<td>SDRAM</td>
</tr>
<tr>
<td>0x00000000</td>
<td>SSRAM</td>
<td>BootROM/flash</td>
</tr>
</tbody>
</table>

nMBDET=1
REMAP=x

nMBDET=0
REMAP=0

nMBDET=0
REMAP=1
Outline

- ARM Integrator Core Module (CM) [1]
- ARM Integrator Logic Module (LM) [2]
- ARM Integrator ASIC Application Platform (AP) [3]
- System Memory Map [1]
- Semihosting
- Lab3 – Core Peripheral
What is Semihosting?
- A mechanism whereby the target communicates I/O requests made in the application code to the host system, rather than attempting to support the I/O itself.

How does it work?
- The debugger running on the host will handle the communications with the target application hardware.
- The I/O request from the target application hardware will be handled and display by the host’s debugger.

Advantage
- Enables the developers to perform the system I/O through the host’s debugger.
- The time and efforts for the developer to support system I/O request by writing hardware drivers is not required.
Semihosting (2/2)

```
printf("hello\n");
```

Application Code → SWI → Library Code

SWI handled by debug agent

Host computer → Communication with debugger running on host

hello → Text displayed on host screen
Semihosting operations are requested using a semihosted SWI numbers:
- **0x123456** in ARM state.
- **0xAB** in Thumb state.

Software Interrupt (SWI) Interface
- A Software Interrupt (SWI) is requested with an SWI number.
- Different operations in the SWI are identified using value of r0.
- Other parameters are passed in a block that is pointed by r1.
- The result is returned in r0. It could be an immediate value or a pointer.

3. Semihosting SWIs functions
- Semihosting operations used by C library functions such as printf(), scanf() uses semihosting SWIs.
### Software interrupt (SWI) in ARM

<table>
<thead>
<tr>
<th>SWI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_OPEN (0x01) on page 5-12</td>
<td>Open a file on the host</td>
</tr>
<tr>
<td>SYS_CLOSE (0x02) on page 5-14</td>
<td>Close a file on the host</td>
</tr>
<tr>
<td>SYS_WRITEC (0x03) on page 5-14</td>
<td>Write a character to the console</td>
</tr>
<tr>
<td>SYS_WRITENO (0x04) on page 5-14</td>
<td>Write a null-terminated string to the console</td>
</tr>
<tr>
<td>SYS_WRITE (0x05) on page 5-15</td>
<td>Write to a file on the host</td>
</tr>
<tr>
<td>SYS_READ (0x06) on page 5-16</td>
<td>Read the contents of a file into a buffer</td>
</tr>
<tr>
<td>SYS_READONLY (0x07) on page 5-17</td>
<td>Read a byte from the console</td>
</tr>
<tr>
<td>SYS_ISERROR (0x08) on page 5-17</td>
<td>Determine if a return code is an error</td>
</tr>
</tbody>
</table>
Outline

- ARM Integrator Core Module (CM) [1]
- ARM Integrator Logic Module (LM) [2]
- ARM Integrator ASIC Application Platform (AP) [3]
- System Memory Map [1]
- Semihosting

- *Lab3 – Core Peripheral*
Lab 3: Core Peripherals

- **Goal**
  - Understand available resource of ARM Integrator
    - Integrator/AP
    - Core Module (CM)
    - Logic Module (LM)
    - Memory-mapped device
    - Timer/Interrupt

- **Guidance**
  - Introduction to Important functions used in interrupt handler

- **Principles**
  - ARM ASIC Platform Resources
  - Semihosting
  - Interrupt handler
  - Architecture of Timer and Interrupter controller

- **Steps**
  - The same to that of code development

- **Requirements and Exercises**
  - Modified the C program. We use Real-Time Clock instead of timer to show our IRQ0 values.

- **Discussion**
  - How to use multi-timer/interrupt.
Several important functions are used in this example:

- **Install_Handler**: This function install the IRQ handler at the branch vector table at 0x18.

- **myIRQHandler**: This is the user’s IRQ handler. It performs the timer ISR in this example.

- **enableIRQ**: The IRQ enable bit in the CPSR is set to enable IRQ.

- **LoadTimer, WriteTimerCtrl, ReadTimer, ClearTimer**: Timer related functions.
References

[2] DUI0126B_CM7TDMI_UG.pdf