Core Peripherals

Speaker: Tzu-Wei Tseng

Adopted from National Chiao-Tung University
IP Core Design

SOC Consortium Course Material
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]
- 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
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
Outline

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

- ZBT SSRAM
- OSC1
- OSC2
- Multi-ICE
- Trace
- Module/motherboard connectors
- Interface module connector
- LA connector
- Push button
- Switches
- LEDs
- Prototyping grid
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]
- 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
ARM Integrator/AP Block Diagram
Assembled Integrator/AP system
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.
System Controller FPGA (2/2)

- 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

- System bus
- External system bus interface
- System bus
- PCI bridge controller
- PCI bridge local bus interface
- Flash, SSRAM and ROM
- PCI Host Bridge
- Arbiter
- Static memory controller
- Status and control registers
- System bus bridge
- Peripheral bus (APB)
- Counter/timers
- Real time clock
- GPIO
- 2xUART
- PS/2 keyboard/mouse interface
- LED/display/switch
- Interrupt controller
A reset controller is incorporated into the system controller FPGA.

The hardware reset sources are as follows:

- Push-button \textbf{PBRST} and CompactPCI signal \textbf{CP\_PRST}
- ATX power OK signal \textbf{nPW\_OK} and CompactPCI power fail signal \textbf{CP\_FAL}
- \textbf{FPGADONE} signal (routed through CPCI arbiter to become \textbf{nRSTSRC5})
- Logic modules using \textbf{nEXPRST}
- Core modules (and Multi-ICE) using \textbf{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

[Diagram showing the architecture of an interrupt controller system, including various components and connections.]

SOC Consortium Course Material
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
System Bus Architecture
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
Counter/Timers Registers (1/2)

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/Timers Registers (2/2)

### 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; 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]
- Lab3 – Core Peripheral
System Memory Map

- LM
- CM alias memory
- PCI
- ROM / RAM and peripherals
- Reserved
- EBI
- Peripheral regs
- CM 0, 1, 3, 3
- 256MB SDRAM (CM 0)
- 256MB SDRAM (CM 1)
- 256MB SDRAM (CM 2)
- 256MB SDRAM (CM 3)
- CS 3 (EXPM)
- SSRAM
- Flash
- Boot ROM
- Spare
- GPIO
- LED/Switch
- Mouse
- Keyboard
- UART 1
- UART 0
- RTC
- Int control
- Counter/Timer
- EBI regs
- Sys control
- CM regs

SOC Consortium Course Material
Core Module Memory Map

Standalone

Abort

0x11000000
SSRAM alias

0x10800000
CM registers

0x10000000
SDRAM

0x00400000
SSRAM

nMBDET=1
REMAP=x

Attached to a motherboard

Motherboard

0x10000000
CM registers

0x10800000
SSRAM alias

0x11000000
Abort

SDRAM

0x00000000
SSRAM

nMBDET=0
REMAP=0

BootROM/flash

nMBDET=0
REMAP=1

Motherboard

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]
- 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

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

Guidance
- Introduction to Important functions used in interrupt handler

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 installs 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