Lab6: Virtual Prototype: ARMulator

Speaker: Nelson Chang
Directed by Prof. Tian-Sheuan Chang
April, 2004, NCTU
Goal of This Lab

- Be able to write and add ARMutator C hardware model
- Learn to write simple drivers
Outline

- *System synchronization*
- ARMulator C hardware model
- Lab6 – Virtual Prototype: ARMulator
Interrupt

- An IP device signals an interrupt when it completes its tasks enabled by ARM core. We say that the IP "raised an interrupt request (IRQ)". This IRQ tells the ARM core that it has finished its task, and requests to be handled.

Diagram:

- IP0, IP1, IP2, and IP3 raised interrupt request (IRQ) at the same time. The IRQs are sent to the interrupt controller.

- Interrupt controller receives the IRQs and updates the IRQ status indicating the IRQ sources.

- ARM core receives the IRQs, determines which IRQ should be handled according to programmed priorities, and then executes the corresponding interrupt service routine (ISR).

- The ISR performs its operations and clears the IP0's interrupt.
Polling

- The ARM core keeps accessing a certain register in the IP which indicates whether it has completed its task enabled by the ARM core for a certain time interval. Once the IP has done its task, the register changes its value, so the ARM core could know the IP is ready and the IP requires to be handled. The action of continuous accessing and checking the register with a certain time interval is called "polling".
Outline

- System synchronization
- ARMulator C hardware model
- Lab6 – Virtual Prototype: ARMulator
Overview of ARMulator

Armulator
- Instruction set simulator
- Model instruction set and counts cycles
- Include communication facilities with the debugger
- Components
  - ARM processor
  - memory system
  - peripherals

Semihosting
Use the I/O facilities of the host computer, instead of providing the facilities on your target system

Debugger Communication
- Debug monitor
  - Angel or RealMonitor
- EmbeddedICE
- Embedded Trace Macrocell

host
I/O facilities
Armulator
debugger
hardware
Basic Model Interface

- Extra models can be added without altering existing models.
  - Each model is self contained.
  - Communicates with ARMuulator through defined interfaces.

- Parts in basic model interface
  - Data structure declaration
    - Declares private data structure
  - Initialization
    - Initialize private variables
    - Install callbacks
  - Finalization
    - Uninstall callbacks
    - Called upon ARMuulator closing
Operating memory space of ARMulator and ARM application SW

- ARMulator runs on **PC or Workstation**
  - Pointers in C hardware models point to memory space on **PC**
- ARM application runs on **ARMulator**
  - Pointers in application software point to **ARMulator** memory space
  - ARM application software **cannot** access the memory space of **PC**
Outline

- System synchronization
- ARMulator C hardware model
- Lab6 – Virtual Prototype: ARMulator
Lab 6: Virtual Prototype: ARMulator

- **Goal**
  - Be able to write and add ARMulator C hardware model
  - Learn to write simple drivers

- **Principles**
  - ARMulator hardware models
  - Memory mapped register

- **Guidance**
  - Observer how hardware model works

- **Steps**
  - Use *nmake* to build the hardware model, and run the demo program to test the hardware.
  - Observe how hardware model works using VC++ 6.0

- **Requirements and Exercises**
  - Add a new matrix transpose hardware
  - Test the newly add hardware

- **Discussion**
  - Compare the hardware performance with pure software implementation
Interaction between ARM and RGB2YUV

ARM Processor (Master)
- Load RGB values to RGB2YUV
- Enable RGB to YUV by writing 0x01 to Control register
  
  Idle for loop, the processor can do other tasks while HW is running

RGB2YUV (Slave)
- RGB2YUV done, generate IRQ, set Done bit in Control register
  
  IRQ detected, branch to ISR. ISR clears the interrupt and disables RGB2YUV. Copy RGB2YUV result to memory if necessary.
Call-graph in RGB2YUV

RGB2YUV_Access is called upon a reference to its address range
References

[1] ARM Application Note 32: The ARMulator [DAI0032E].