Lab6: Virtual Prototype:ARMulator

Speaker: Nelson Chang

Directed by Prof. Tian-Sheuan Chang

April, 2004, NCTU

Goal of This Lab

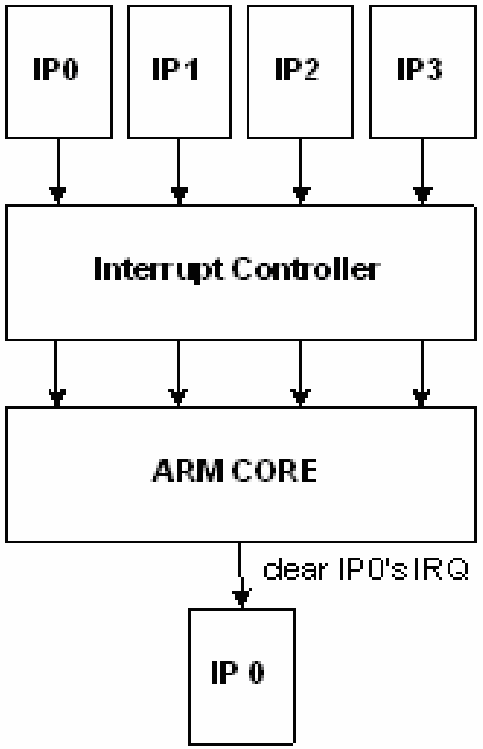
- Be able to write and add ARMulator C hardware model
- Learn to write simple drivers

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.



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

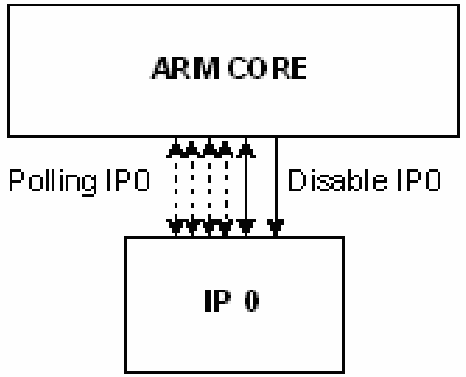
Interrupt controller receives the IRQs and update the IRQ status indicating the IRQ sources.

ARM core receives the IRQs, deteremines 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".



ARM core polls IP0's ready register after IP0 has been enabled.

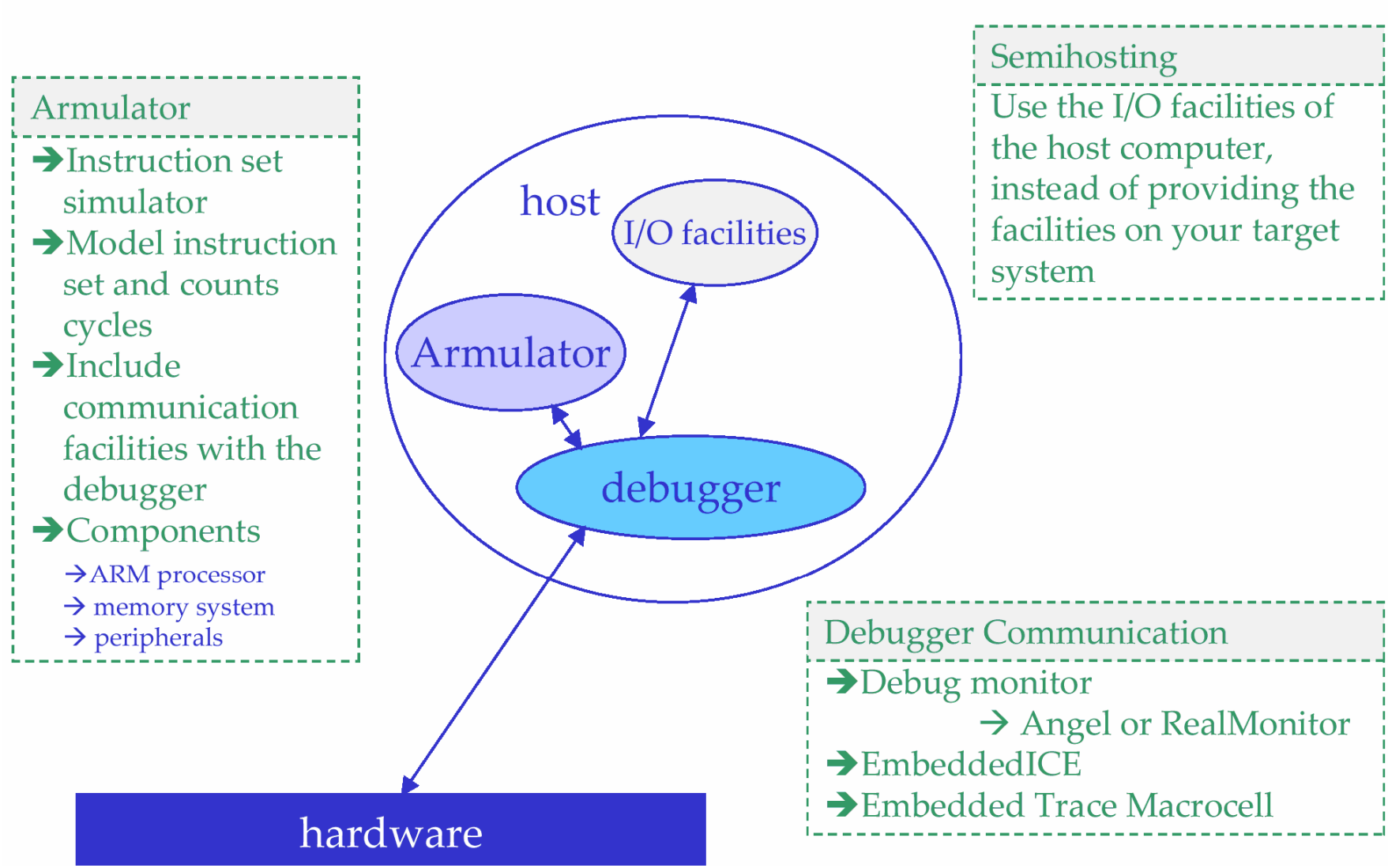
Once IP0 is done with its operation, ARM core will knowfrom the changed value of the ready register.

ARM core will execute the corresponding operations and then disable IP0.

Outline

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

Overview of ARMulator

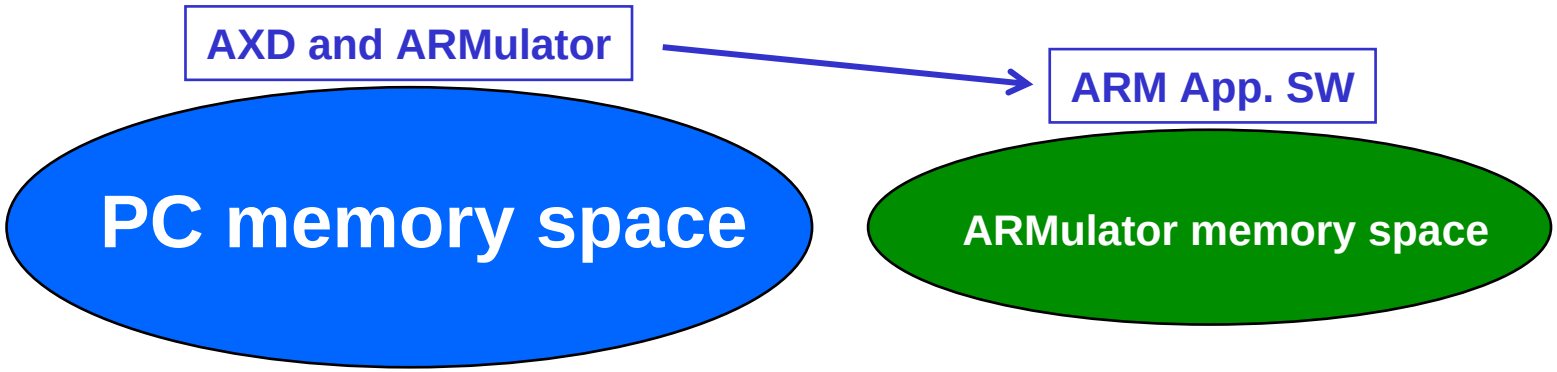


Basic Model Interface

- Extra models can be added without altering existing models.
  - Each model is self contained.
  - Communicates with ARMulator 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 ARMulator closing

Operating memory space of ARMulator and ARM application SW

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



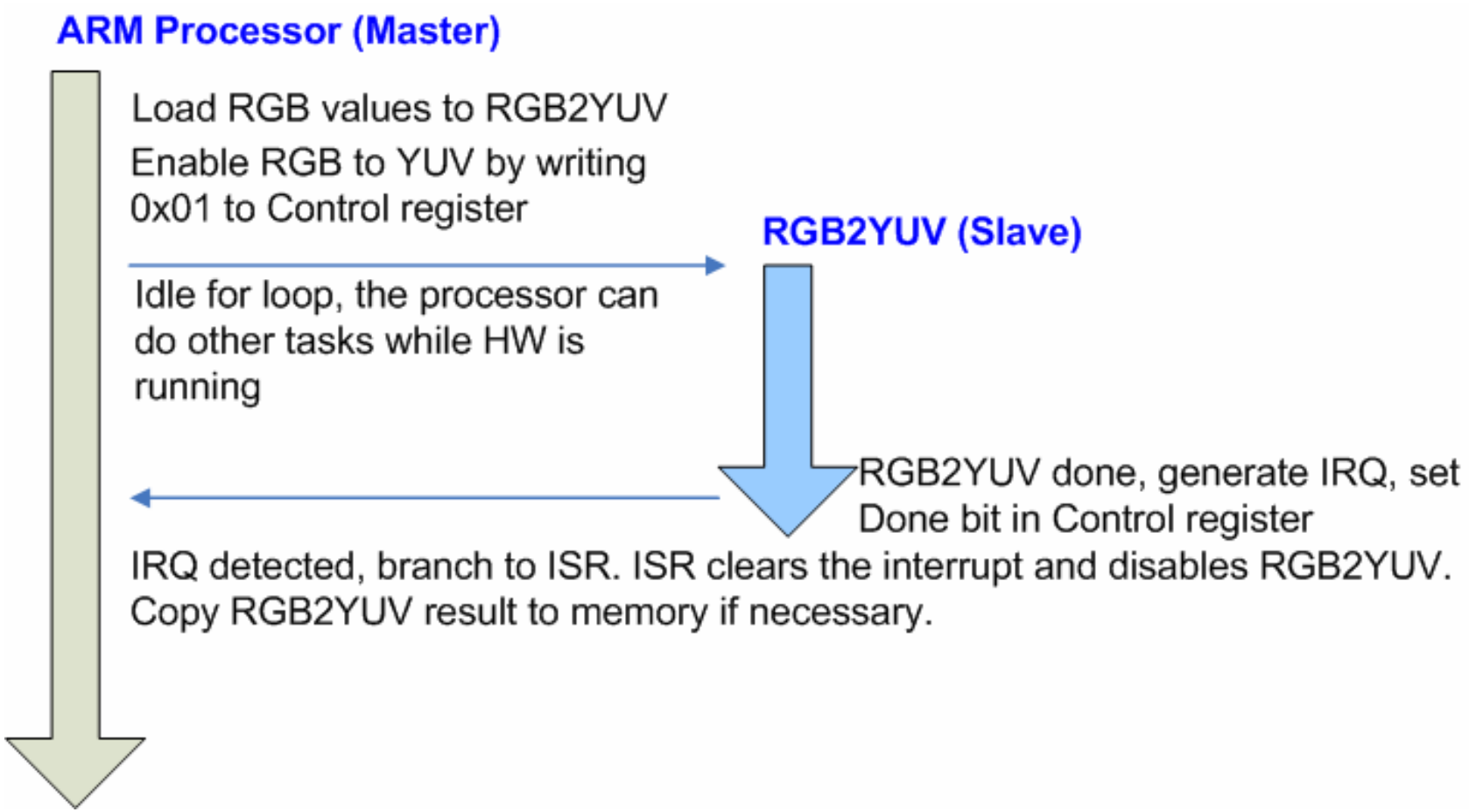
Outline

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

Lab 6: Virtual Prototype: ARMulator

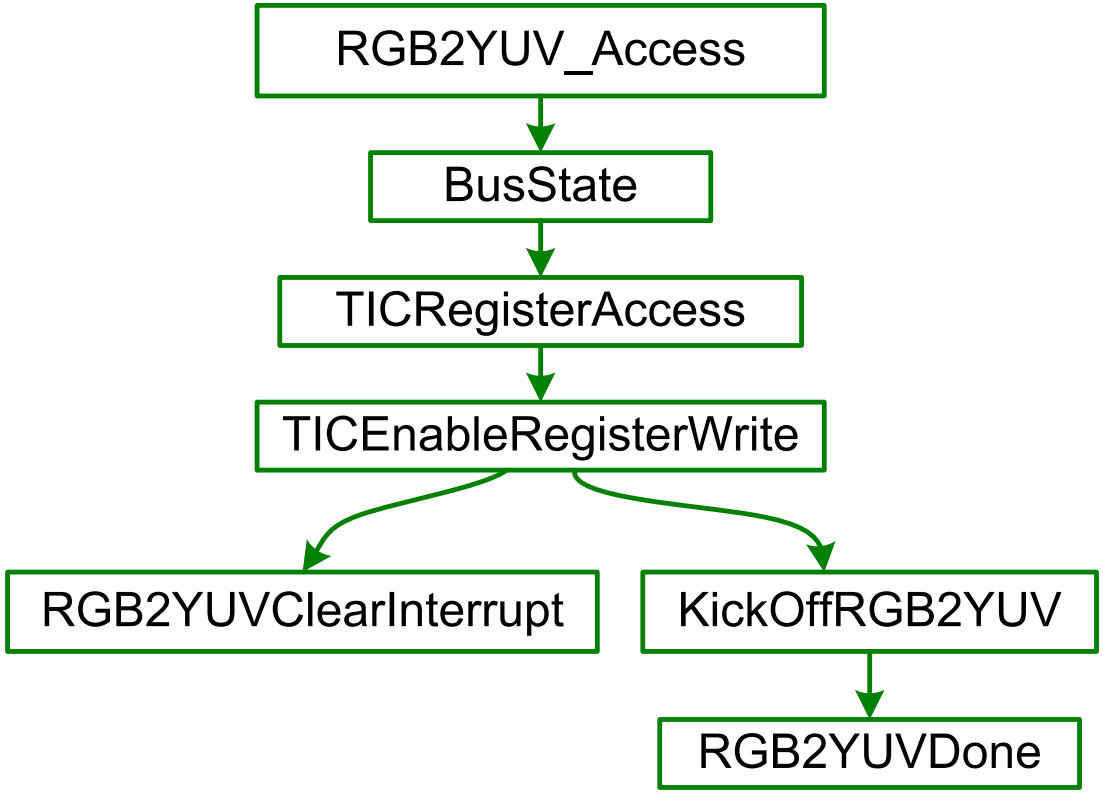
- Goal
  - Be able to write and add ARMulator C hardware model
  - Learn to write simple drivers
- Principles
  - ARMulator hardware models
  - Memory mapped register
- Guidance
  - Observer how hardware model works
- Steps
  - Use *nmake* to build the hardware model, and run the demo program to test 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



Call-graph in RGB2YUV

- RGB2YUV_Access is called upon a reference to its address range



References

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

[2] ARM Debug Target Guide [DUI0058D].