LAB 3 – Synchronous Serial Port Design Using Verilog

Fall 2011
TA : Junyoung Park
VLSI Lab Overview

- **Verilog**
  - Hardware Description Language (HDL)
  - Behavioral Description
  - Structural Description
  - Synthesizable code
VLSI Lab Overview (Con’t)

- RTL level system design using Verilog (Behavioral modeling) to design
  - Part A: Synchronous Serial Port (SSP)
  - Part B: Integration of SSP with ARM processor using a SoC bus interface called “WISHBONE”
- Verilog Simulators
  - Simulator – VCS (Synopsys), ncverilog (Cadence)
- Tools
  - Simulation:
    - VCS – Verilog simulation tool
    - VirSim – GUI to control simulation and view waveforms
  - Synthesis:
    - Design Vision – Logic Synthesis
Lab3A – SSP Design

■ SSP Perform
  - Parallel-to-serial conversion on data received from a processor
  - Serial-to-parallel conversion on data received from a peripheral device

■ Your design
  - Your SSP provides buffering capability on both the TX and RX logic using FIFOs to allow up to four 8-bit words to be stored in independently in both TX and RX module
Lab3A – SSP Block Diagram

- PWDATA[7:0]
  - PSEL
  - PWRITE

- PRDATA[7:0]
  - PCLK
  - CLEAR_B

- TxFIFO
  8 bits wide, 4 locations deep
  - TxDATA[7:0]
  - signals*

- RxFIFO
  8 bits wide, 4 locations deep
  - RxDATA[7:0]
  - signals*

- Transmit/Receive Logic

- SSPTXINTR
- SSPOE_B
- SSPTXD
- SSPFSSOUT
- SSPCLKOUT
- SSPCLKIN
- SSPFSSIN
- SSRXD
- SSPRXINTR
Lab3A – SSP Specifications

- Transmit FIFO (TxFIFO)/ Receive FIFO (RxFIFO)
  - 8-bit wide, 4-location deep FIFO memory buffer
  - If FIFO is full, generate SSPTXINTR/SSPRXINTR interrupt signal and refuse to accept any additional data

- Transmit and Receive Logic
  - TX Logic:
    - read from TxFIFO and perform parallel-to-serial conversion
    - Send the serial data synchronously
  - RX Logic:
    - perform serial-to-parallel conversion on the incoming data synchronously
    - Send data into RxFIFO
Lab3A – Port Descriptions

- Name your top module as “ssp”
- You should use the exact names shown below
- SSP module port description
  - Input port: PCLK, CLEAR_B, PSEL, PWRITE, PWDATA[7:0], SSPCLKin, SSPFSSin, SSPRXD
  - Output port: PRDATA[7:0], SSPOE_B, SSPTXD, SSPCLKOut, SSPFSSOut, SSPTXINTR, SSPRXINTR
Lab3A – Signal Description

- Please refer to the lab website for details
- Signal descriptions
  - CLEAR_B: Low active clear signal
  - SSPCLKIN(RxLogic), SSPCLKOUT(TxLogic)
    - Synchronization clock for data transfer
    - 2 times slower than PCLK (clock for SSP)
  - PSEL: Chip select signal for SSP
  - PWRITE: High – Write to SSP, Low – Read from SSP
  - SSPFSSIN(RxLogic), SSPFSSOUT(TxLogic)
    - Indicate starting of data transfer
  - SSPOE_B: Low active output enable signal, indicating when SSPTXD is valid (Used only for transmission)
  - Assume perfect synchronization between SSPCLKIN and SSPCLKOUT - testing is done with external loop back
    - Dataflow: (Written by the processor) → SSP TxFIFO → SSPTXD → SSPRXD → SSP RxFIFO → (Read by the processor)
Lab3A – Timing Diagram (single transfer)

Frame format of SSP (for single transfer)
Lab3A – Timing Diagram (continuous transfer)
Lab3A – Logic Synthesis

- Regularly synthesize your code by reading in your design to make sure your design is synthesizable
Lab3B – WISHBONE SoC interconnection interface

- **Objective**
  - To design WISHBONE bus interface module to integrate heterogeneous modules
Lab3 – What is given to you

ARM Processor (main.v)

ARM Core (arm.v)

Memory (mem2.v)

/images/mem.data

asm_arm
yourtest >
mem.data

MOV R1, R2
MOV R2, R3
MOV R3, R4
...


Lab 3 – What you should do

ARM Processor (main.v)

ARM Core (arm.v)

Wishbone Bus Interface
w_master.v
w_slave.v

Memory (mem2.v)

Clock Management Unit (cmu.v)

SSP (ssp.v)
Lab3B – Module Descriptions

- ARM processor (Given)
  - A master core of your system
- Memory module (Given)
  - Simple memory block used as an instruction memory
- Synchronous Serial Port (Designed in Lab3A)
  - Slave core of your system
- WISHBONE & Clock Management Unit (Need to design in Lab3B)
  - WISHBONE master: communicate with the ARM core
  - WISHBONE slave: communicate with the SSP and memory module
  - Clock Management Unit: manage all the clocks and interrupts
Master:
- When the ARM core gets the data transfer activities, WISHBONE master has to transfer the data from/to the WISHBONE slave

Slave:
- Transfer the data to/from the memory of SSP according to request from the ARM core, and then report the result of the transfer to the WISHBONE master

Should be done with WISHBONE’s timing specifications
Clock Management Unit
- for ARM processor (phi1, phi2)
- for Memory (phi1, phi2)
- for WISHBONE and SSP (clk_o)

If interrupts are occurred, Clock Management Unit should stop the clock for ARM processor.
Lab3B – WISHBONE (Con’t)

- **Instruction Read**
  - When ARM accesses h’00000000 – h’000FFFFF
  - Generate memory access signals
  - Deliver instructions from memory to ARM
  - Mem → Slave → Master → ARM
    - Path: Read signal (ARM) → master → slave → memory → data from memory → slave → master → ARM (instruction transferred to ARM)

- **Data Read from SSP**
  - When ARM access h’0010001
  - Generate SSP access signals
  - Deliver data from SSP (RxFIFO) to ARM
  - SSP → Slave → Master → ARM
    - Path: Read signal (ARM) → master → slave → SSP → data from SSP → slave → master → ARM (data transferred to ARM)
Lab3B – WISHBONE (Con’t)

- **Data Write to SSP**
  - When ARM access h’0010000
  - Generate SSP access signals
  - Deliver data from ARM to SSP (TxFIFO)
  - ARM → Master → Slave → SSP
    - Path: Read signal and data (from ARM) → master → slave → SSP (data written to SSP)

- **Interrupt Handling**
  - Stop ARM by holding phi1 and phi2 (Clock Management Unit)
    - E.g., SSP Overflow Interrupt
    - SSP → Clock Management Unit → ARM (holds clocks)
Lab3B – Simulation

- Write your own assembly test program using ARM instructions
  - Read ARM manual given to you

- Compile the assembly code and save it to /image/mem.dat (actual memory image)
  - `asm_arm my_assembly_code > mem.data`
  - Template mem.data can be found in Lab3 website – you may choose to create your own

- Run VCS