ELEC 4200 Lab#11
Interrupting Parallel Load Register for use with a Processor Core

AUBURN
UNIVERSITY

SAMUEL GINN
COLLEGE OF ENGINEERING

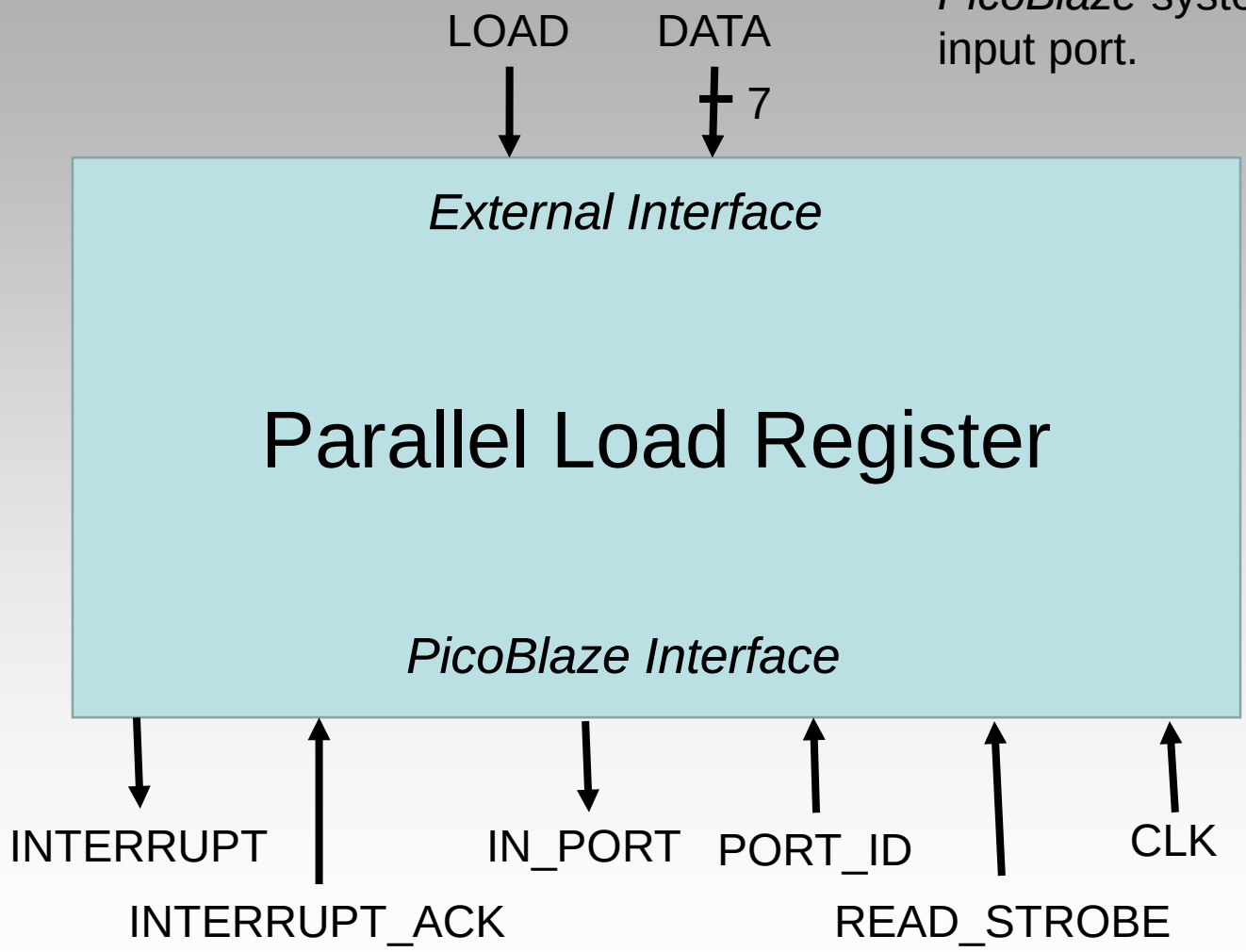
- References you may need:
  - *PicoBlaze KCPSM6 User Manual*
  - *PicoBlaze 8-bit Embedded Microcontroller User Guide*

Overview

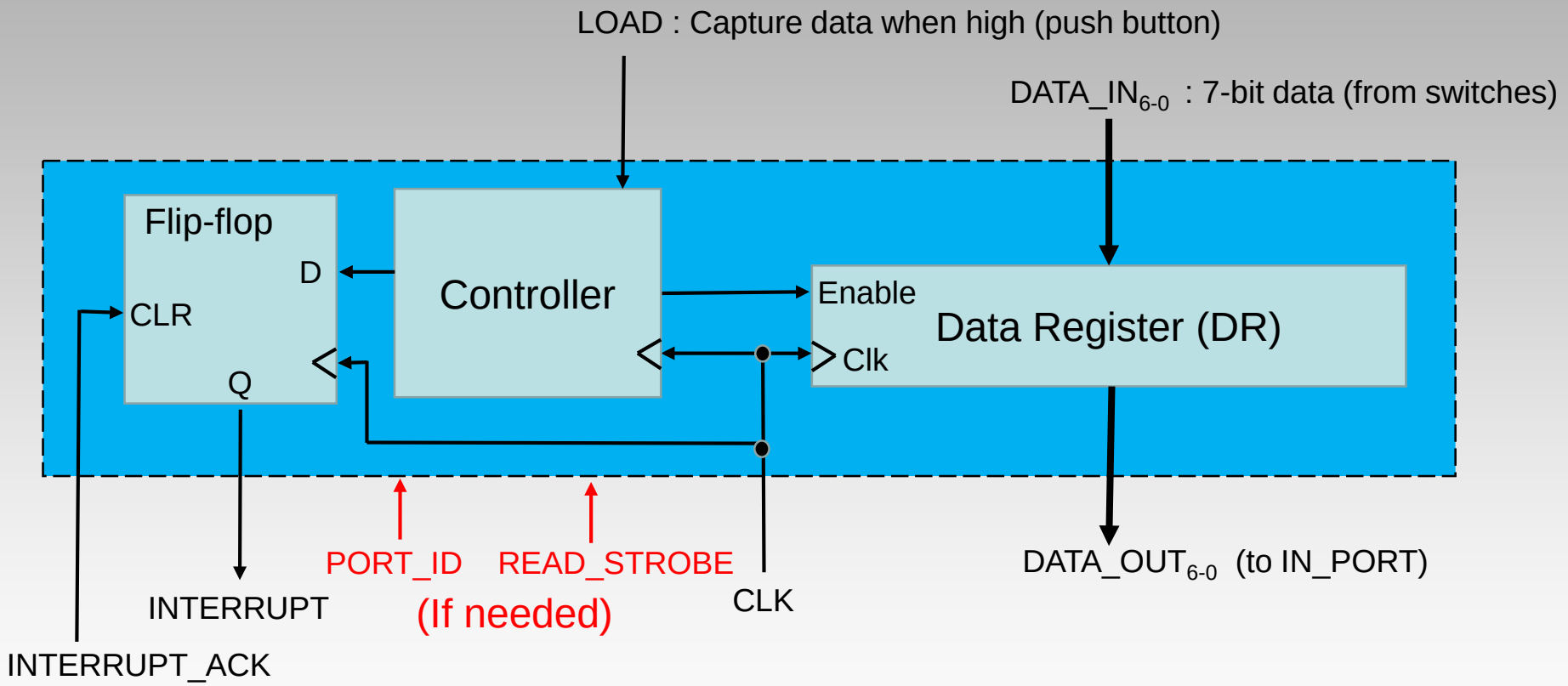
- Design a 7-bit ***Parallel Load Register*** that can be interfaced to *PicoBlaze* as an interrupting input port
  - » Requires understanding of *PicoBlaze* input/output ports and interrupt mechanism
- The Parallel Load Register is to operate as a “slave device” on the NEXYS 4 DDR board
  - – Parallel data is to comprise 7 bits of information
  - – LOAD signal is to be supplied by a push button
  - – DATA is to be supplied by slide switches and/or push buttons
- The Parallel Load Register is to activate an INTERRUPT signal when new data is available for *PicoBlaze* to read
- The Parallel Load Register should operate as a *PicoBlaze* input port, with received data read by an INPUT instruction

Parallel Load Register

To be instantiated in the *PicoBlaze* system as an input port.



Parallel Load Register Structure



Parallel Load Register Operation

- DATA_IN value is to be captured on the rising edge of CLK after LOAD signal is set high.
  - Controller enables register to be loaded
  - Controller prevents new data from being loaded until current data is read
- On the next rising edge of CLK after LOAD signal is set high, activate the INTERRUPT signal (to indicate data available)
  - Controller enables interrupt flip-flop to be set to 1, but not reset to 0.
- When *PicoBlaze* activates INTERRUPT_ACK, reset the INTERRUPT signal
  - Asynchronous reset of the interrupt flip-flop
- *PicoBlaze* should read the data from the register via an INPUT instruction
  - The register operates as an “input port”
  - New data can be loaded into the register after current data is read
  - Review timing of PORT_ID, IN_PORT, READ_STROBE, CLK

Parallel Load Register Design

VHDL Model Specifications

- Entity inputs/outputs as shown on Slide 3
- Behavior as described on Slide 5
- Structure as shown on Slide 4. Write as one VHDL model, using a separate process for each of the major components
  - Data register DR
  - Interrupt flag
  - Controller (you may use more than one process for this, if desired)
  - LOAD debounce
    - Since a push button will be used for LOAD, “debounce” it to ensure that only one bit is captured per button press, perhaps using something like the “one shot” circuit from previous labs
- Use switches and LEDs on the NEXYS 4 DDR board for testing the Parallel Load Register

Pre-lab Assignment

- Review the *PicoBlaze KCPSM6 User Manual* sections on:
  - INPUT/OUTPUT port design
  - INTERRUPT and INTERRUPT_ACK signals
- Write a VHDL model of the Parallel Load Register (PLR)
  - Be prepared to verify this design via simulation
- Integrate the PLR into your *PicoBlaze* system
- Write an assembly language program to verify that you can correctly receive data from the PLR using interrupts. Your program is to control a single output port in the following manner.
  - Your program should continually toggle the MSB of your output port at a rate the human eye can clearly see. (Hint: Use nested delay loops to create a sufficiently long software delay.)
  - When an INTERRUPT signal is supplied by the PLR, the value supplied should be written on the lower 7 bits of your output port before returning to your main operation.
  - The 7-bit value supplied by the user should remain on the lower 7 bits of the output port until a new value is supplied.

Lab Exercise

- Simulate your Parallel Load Register (PLR) VHDL model to verify its correctness.
- Add the PLR to your previous *PicoBlaze* system, and map the PLR input signals to buttons/switches on the NEXYS 4 DDR board.
- Build (synthesize, map, place, route) and download the *PicoBlaze* system to the NEXYS 4 DDR board
- Demonstrate the operation of the *PicoBlaze* system, including the PLR, to the GTA

Report Guidelines

- Be sure to include all sections required by the lab manual guidelines. In addition be sure your report includes the following:
  - SPI receiver VHDL model
  - Simulation procedure and results
  - Steps taken to simulate, synthesize, and download your code
  - Synthesis results (LUTs, FFs, slices, etc)
  - Experimental results - what went right and wrong in your design