Interrupt Controller …

- Interrupt Controller overview?
  - In any system, processor is the master, peripheral controllers are slaves. Slaves can't initiate a transaction to processor, because transactions can only be initiated by master.
- What is Interrupt?
  - Interrupt is a mechanism used by peripheral controllers to get processor attention. Peripheral controller ex: USB controller, PCIe, DDR, KBD, SPI, etc
- Analogy: Trainer, students, admin
  - admin is the interrupt controller(INTC)
    - INTC will interface with processor and peripheral controllers.
  - Peripheral controller will raise interrupts(~student raising hand), these interrupts goes to INTC(admin will make note of questions and students who asked questions). INTC will forward these interrupts one-by-one to processor on priority basis. (admin will give these questions to trainer one-by-one).
  - Processor will service the request(~trainer answering student question)
  - Processor will indicate to INTC that interrupt has been serviced.
  - That interrupt will be dropped (~student dropping his hand)
  - INTC will now get next highest priority active interrupt, above process will repeat till all active interrupts are serviced. (till all questions are answered)
  - Once all interrupts are serviced, processor will resume its regular task.(trainer starts teaching the subject)

Interrupt Controller …

- Analogy: Trainer, students, admin
  - Trainer => processor
  - admin => interrupt controller(INTC)
  - students => peripheral controller
- As part of regular flow, trainer will teaching
  - when student has a doubt, he raises the hand(~generating an interrupt)
  - Interrupt is collected by the admin(sitting next to trainer)
  - admin collects all the questions(~interrupts), figures out which is the highest priority among all the questions.
  - admin gives this number(student name/number) to the trainer
  - trainer will address the question
  - question has been answered.
  - student will drop the hand(dropping the interrupt)
  - this process continues till all questions(interrupts) have been answered(serviced)
  - once all questions are answered, trainer will continue with the normal flow(processor will go with its normal instruction flow)
  - INTC will interface with processor and peripheral controllers.

Interrupt Controller …

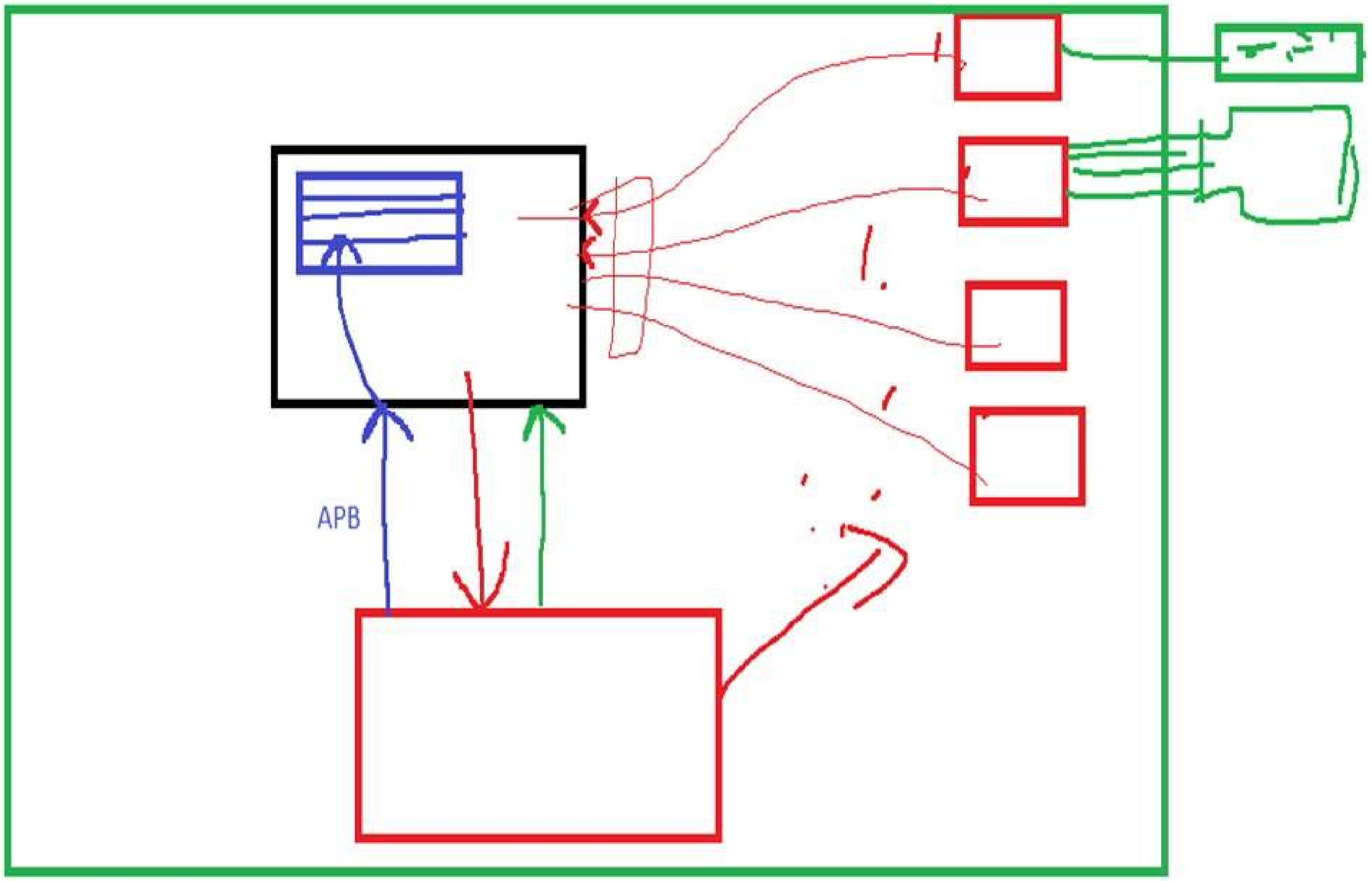
- why we need Interrupt Controller?
  - Without INTC, there is a chance of losing(dropping of) interrupts.
  - Processor functionality will get complex, because without INTC, processor will have to do the work of INTC.
  - INTC makes sure that, it collects all the interrupts, gives them to processor one by one, so that interrupts are handled properly.
    - It also takes care of interrupt priority.

- What external blocks does interrupt controller interface with?
  - Processor
  - Peripheral Controllers(ex: KBD ctrl, Memory controller, USB ctlr, etc)
  - Testbench has to mimic all above components behavior.

- Interrupt controller detailed functionality
  - discussed in last slide.

- Interrupt controller Architecture

Interrupt Controller Architecture …



Interrupt Controller …

- **how many interface and Ports** required with directions: Input, Output, Inout
  - Processor interface
    - APB Protocol
    - intr_to_service_o[3:0], intr_valid_o, intr_serviced_i
      - Only if intr_valid_o is '1', intr_to_service_o is valid. Processor will consider this signal only if valid is '1'
    - **intr_serviced_i :** Interrupt has been serviced
  - Peripheral interface
    - intr_active_i[15:0] : Interrupts coming from peripherals to INTController
- Registers required?
  - Yes. Required for programming interrupt priority values.
  - How many registers? If there are 16 peripherals, 16 registers are required.
    - reg [3:0] priority_regA[15:0];
      - 4 bits to hold the priority values
- Memories required?
  - No
- Does design require Parameters?
  - parameter NUM_INTRS = 16;
- Sequential design
  - **states:** S_NO_INTR, S_INTR_ACTIVE, S_INTR_GIVEN_TO_PROC, S_PROC_SERVICED_INTR, S_ERROR
    - can we think of any other states for interrupt controller?

Interrupt Controller …

- Building the FSM
    - Write down all the possible states as circles
    - Then do all the possible state transitions(for all possible inputs)
    - What we get is a FSM.
    - We develop the Verilog code for FSM.

Interrupt Controller …

- How many processes: 2 processes
  - to implement one process, we require one always block.
    - programming the registers
      - work on pclk => always @(posedge pclk)
    - handling the interrupts
      - work on pclk
- How to figure out highest priority interrupt among active interrupts?
  - Analogy: Among 16 people, getting the highest number.
    - Ask all people to tell their numbers at once => problem
    - Right way: start from student#0, get number, to start with that is the highest number, then go to index#1, compare this number with earlier number, so on. Finally we have the highest number.
  - INTC will also work in same manner, only difference, we have to pick only indices which have active interrupts. If a peripheral is highest priority but it does not have interrupt, we should not consider it in the priority calculation.
    - FIND HIGHEST PRIORITY CONTROLLER AMONG ALL ACTIVE INTERRUPTS.

Interrupt Controller: Register programming

- priority_regA[paddr-REG_START_ADDR] = pwdata;
    - priority_regA indices are from 0 to 15
    - priority_regA register addresses are from REG_START_ADDR to REG_START_ADDR + 15
        - how do we map them:
            - substract REG_START_ADDR from register address => then we will get the register array index number

- ex:
    - priority_regA register addersses: 8'h30(REG_START_ADDR) to 8'h3F (16 in number)
    - if write tx is done to addr = 8'h35 => which register number is targeted(in priority_regA)? 5th (index starts from 0)
        - how did we get 5 number => 8'h35 - REG_START_ADDR => 8'h35-8'h30 = 5

    - priority_regA register addersses: 8'h40(REG_START_ADDR) to 8'h4F (16 in number)
    - if write tx is done to addr = 8'h35 => invalid address. We are targeting a non-existant location.
    - if write tx is done to addr = 8'h41 => which register number is targeted(in priority_regA)? 1st (index starts from 0)
            - 8'h41 - REG_START_ADDR =>

    - Analogy:
        - 10 houses in a street, address starts from 30 to 39
            - if post comes for house#33 => what is house number is the street = 3 (if count starts from 0)
            - house_array[3] = post;
            - house_array[house_addr-start_addr] = post;
            - house_array[33-30] = post;

Interrupt Controller …

- first_match_f
  - For finding highest priority interrupt, we need a reference number, with which we will compare current interrupt priority value.
  - first_match_f will give us the reference value when we start the for loop iteration, that reference value will be used for comparison for the 1st time.
    - during comparison, if we find another interrupt with higher priority, update the highest priority values.
- what is first_match_f is not there?
  - We don't have a reference number to compare the priroity values.