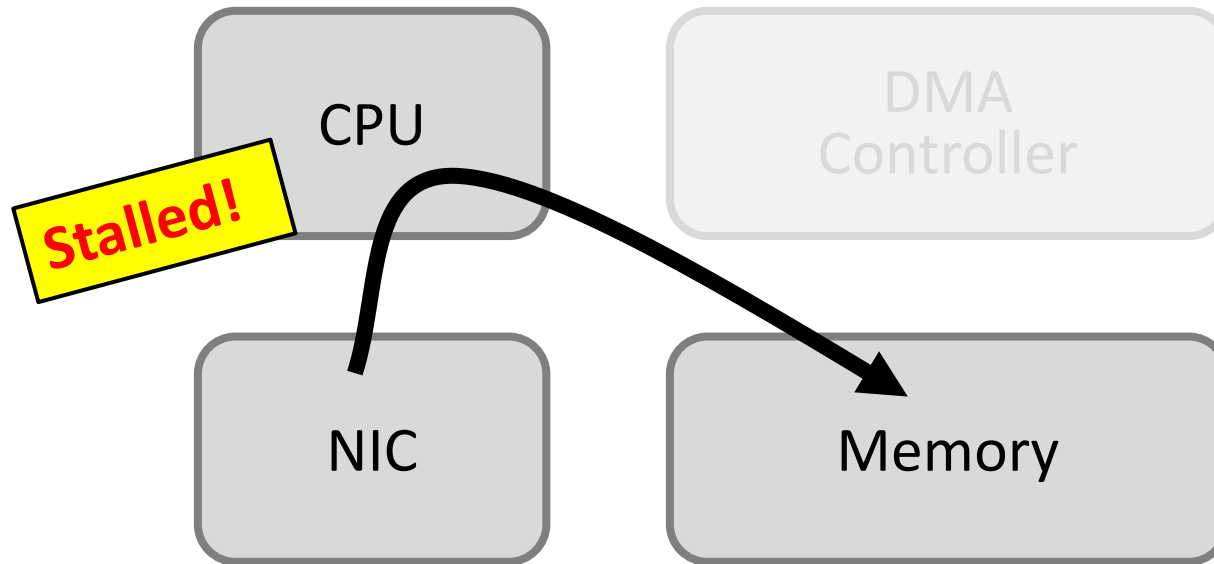


CSED311 Lab7:

Direct Memory Access

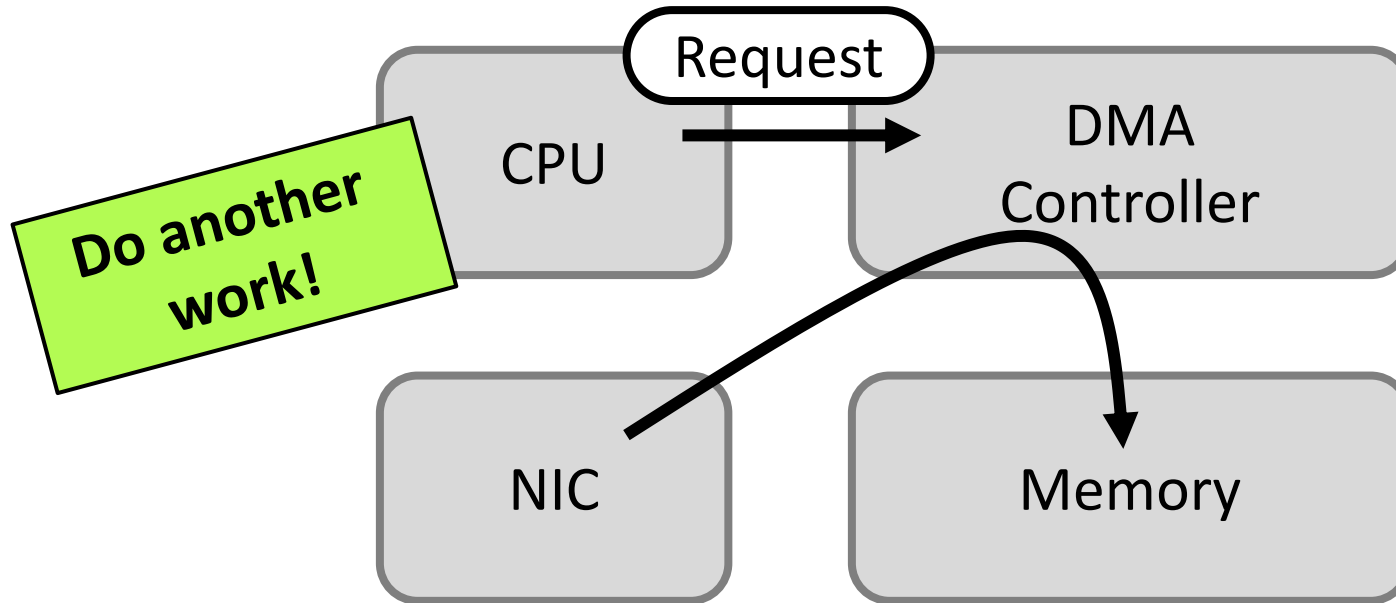
Hyunuk CHO
gusdnr9779@postech.ac.kr

Direct Memory Access



- Without DMA, the CPU must be stalled during the data transfer between I/O devices and the memory
 - To transfer the data, the CPU must read data from the I/O device and write them to the memory
 - During this process, the CPU cannot do useful work

Direct Memory Access



- By letting the DMA controller handle data transfer, the CPU can perform another work while transferring the data
 - CPU requests the DMA controller to load data
 - The DMA controller then loads the data on behalf of the CPU
 - CPU can do another work while the DMA controller transfers data

Experiment Scenario

- Let's simulate an external I/O device
 - An external device that has at least 12 words of data
 - The device notifies the CPU if it has some data to transfer
 - The CPU then transfers the 12 words of data from the device to the memory, using the DMA controller
- Our memory model is accessed in a 4-word unit.
 - Thus, 12 words of data requires 3 ($=12/4$) memory transactions
i.e., write 4 words x 3 times

Experiment Scenario

- Let's simulate an external I/O device

- An external device that has at least 12 words of data
- The device notifies the CPU if it has some data to transfer
- The CPU then transfers the 12 words of data from the device to the memory, using the DMA controller

Need to write a short Verilog code for this device!

- Our memory model is accessed in a 4-word unit

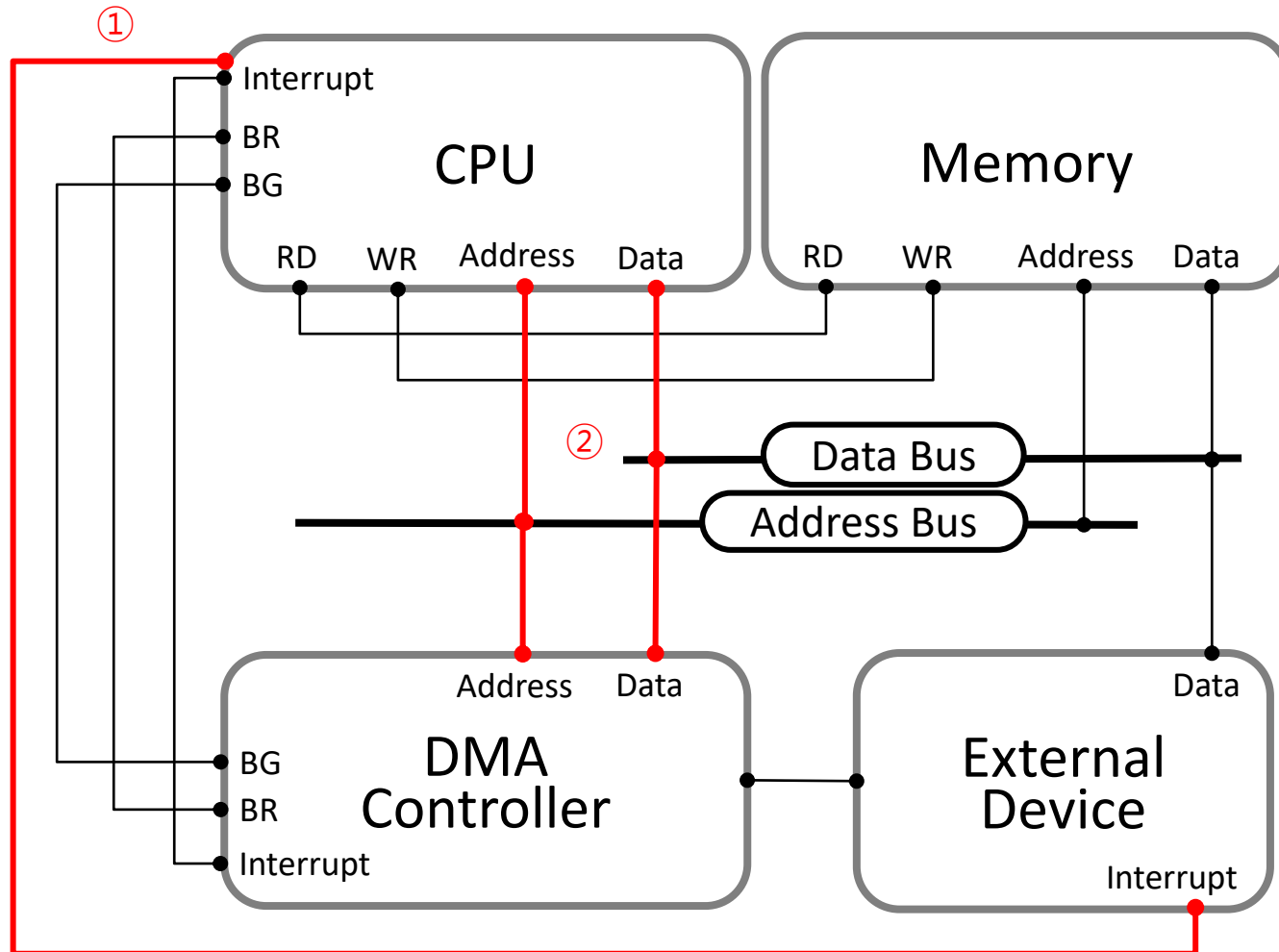
- Thus, 12 words of data requires 3 memory transactions
i.e., write 4 words x 3 times

And this DMA controller too!

In testbench ...

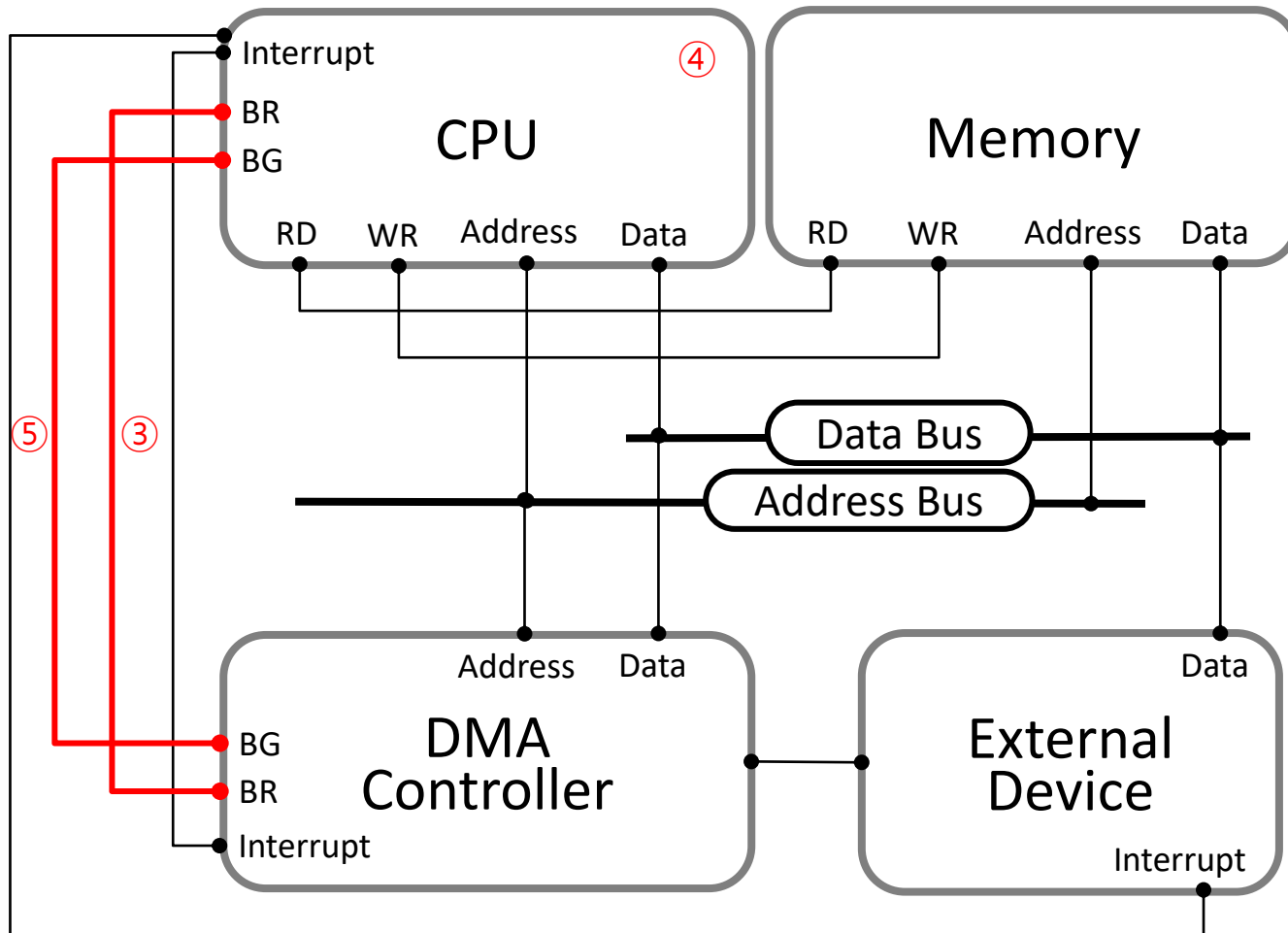
```
cpu UUT ( ... );  
Memory NUUT ( ... );  
DMA_controller DMAC ( ... );  
external_device ED ( ... );
```

Detailed Scenario



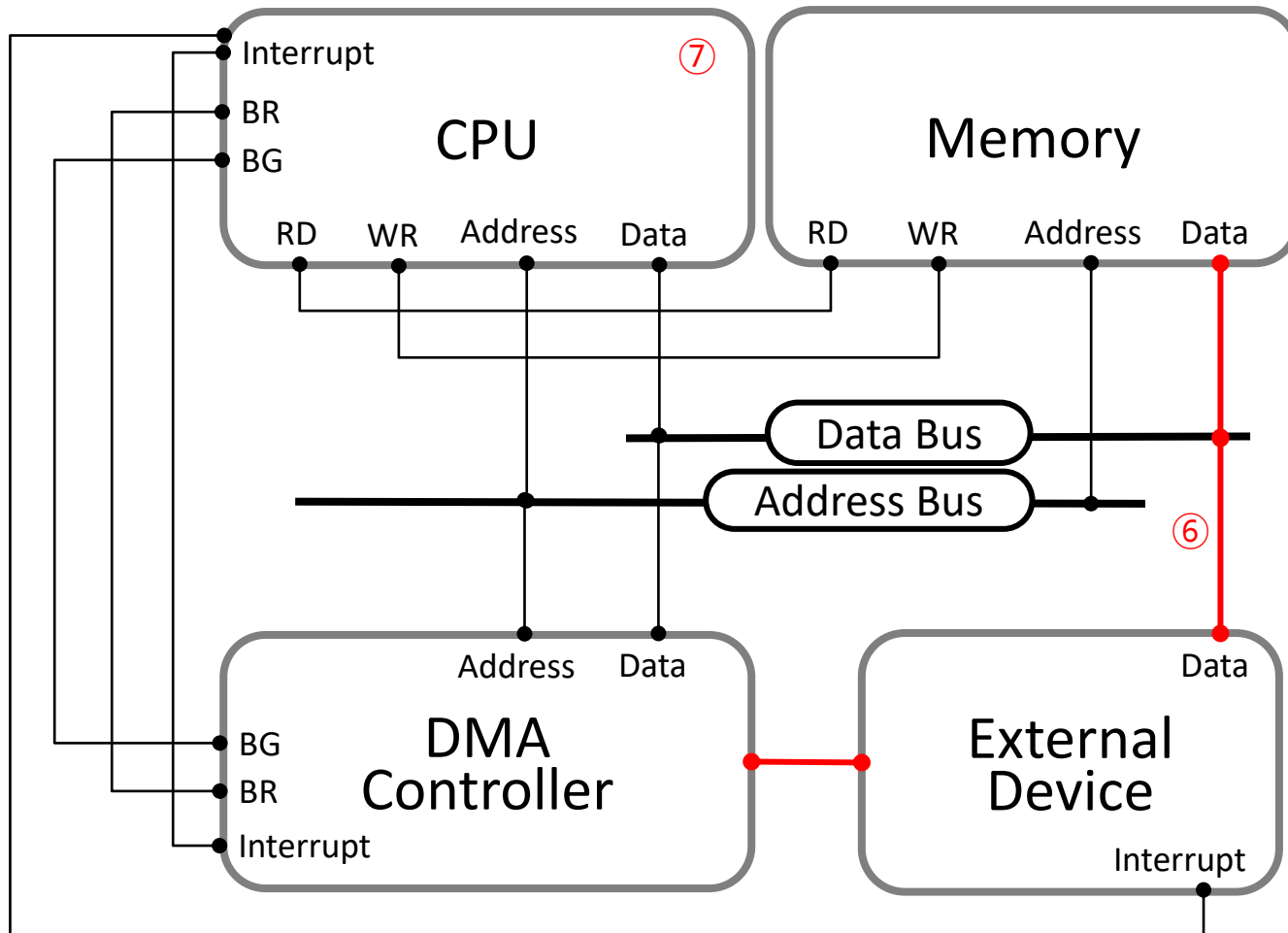
1. An external device sends an interrupt to a CPU
 - You should design an interface for the interrupt
 - CPU should always be ready for the interrupt
2. The CPU sends a command (length, address) to a DMA controller
 - Length: the length of data **12 words (fixed)**
 - Address : the target memory address

Detailed Scenario



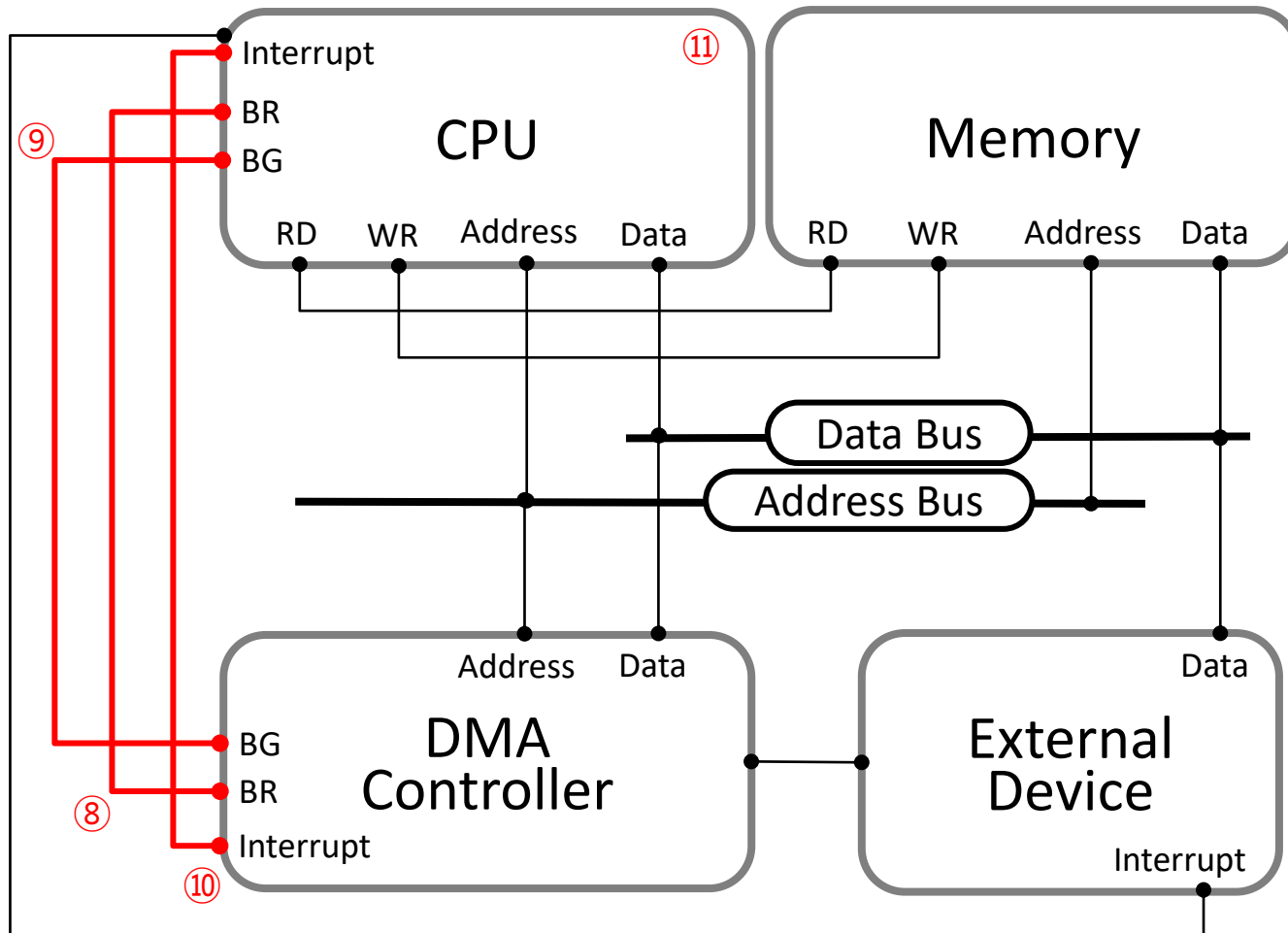
3. The DMA controller raises a BR (Bus Request) signal
4. The CPU receives the signal, then it blocks its usage of the address bus, data bus, and RD/WR line
5. The CPU raises a BG (Bus Granted) signal

Detailed Scenario



6. The DMA controller BG signal and, the external device writes 12 words of data at the designated memory address
7. While the transfer is going on, the CPU can only access the cache (should be blocked when a cache miss occurs)
 - Because it cannot access the memory now

Detailed Scenario



8. When the DMA controller finishes its work, it clears the BR signal
9. The CPU clears the BG signal and enables the usage of memory buses
10. The DMA controller raises an interrupt
11. The CPU handles the interrupt

What to Implement: CPU

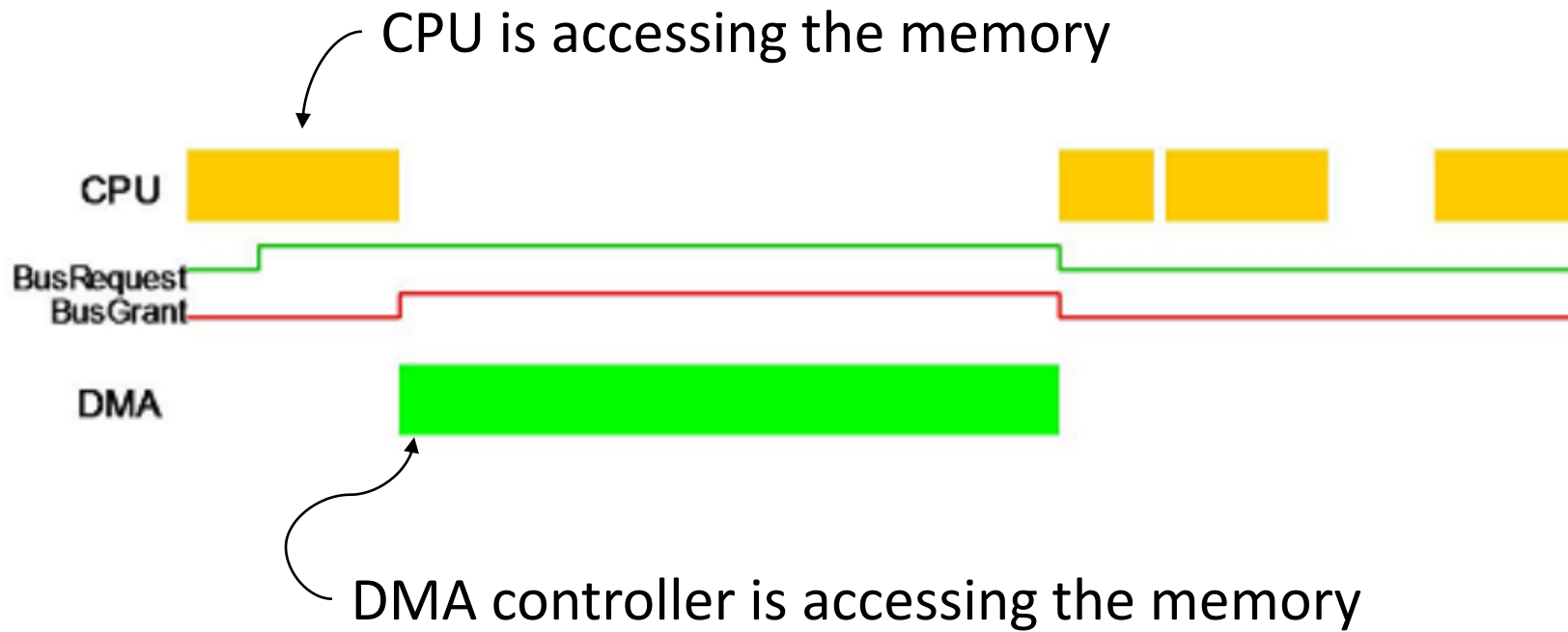
■ CPU Part

- A simple interrupt handling logic
 - Two kinds of interrupt: DMA begin, DMA end
 - CPU should be halted and process the interrupt when it occurs
- A bus arbitrating logic
 - When the DMA controller requests the memory bus, it should be granted as soon as possible
 - When the DMA controller uses the memory bus exclusively, the CPU should not use them
 - When the DMA controller releases the memory bus, the CPU can access the memory

What to Implement: Etc.

- External Device Part
 - It should wait few hundreds of clock cycles, then raise an interrupt
This interrupt means that the device has at least 12 words of data to send
 - When the DMA controller tells the offset of data (0, 4, or 8), writes the proper data on the data bus
- DMA Controller Part
 - When a DMA command from the CPU is received, it tries to grab the memory bus exclusively
 - Then, it sends a request to the external device with the memory address to make the device send the data on the bus
 - After the DMA is finished, send an interrupt to the CPU

Working Model (Waveform)



Grading Criteria

- Your CPU must be still functional!
 - Your CPU should pass every test on the previous testbench
 - Its functional correctness must not be affected by the DMA
- Your CPU must handle external signals and interrupts properly
- Your CPU must be working while the DMA is working, except when it has a cache miss
- After the “End-of-DMA” interrupt, the device’s data should be read from the designated memory address

You should prove these by a waveform or additional WWD tests.

Include a README.txt file in your submission that tells us a detailed explanation about how can we check your DMA result.

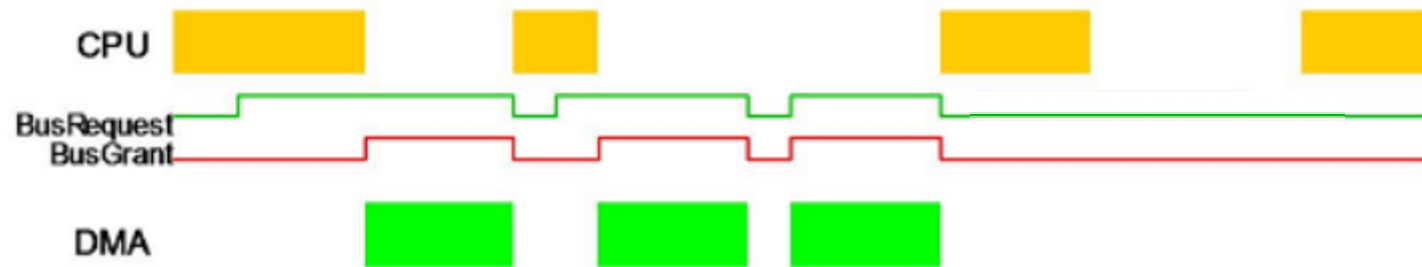
Grading Criteria

- Please follow the coding rules for Verilog
 - Blocking and non-blocking assignments should be used for asynchronous and synchronous circuits, respectively
 - Always statement except for * and pos, neg edge clock is forbidden
 - Delay statement is forbidden
 - Etc...
- Coding rules for the previous labs also apply here

**Any violation of coding rules in this lab
will result in a penalty on your grade.**

Extra Credit (1)

- After the DMA controller sends 4 words, it releases the bus and grabs them again later
- When the bus is release, the CPU can access the memory if needed
- Otherwise, the DMA controller will retake the bus as soon as possible



Extra Credit (2)

- The cache lines can have stale data after the memory is updated with data from the external device
 - ex) cache line for the address $0x32 = 0xAA$
received data written in the address $0x32 = 0xBB$
- The cache line corresponding to the updated memory locations should be invalidated

Tips

- No additional bus
 - CPU, DMA, and DMA controller should share the same bus
- Choose when/where to start DMA carefully
 - Your CPU should still be functional with DMA
 - To get extra credit (2), your DMA scenario should invoke cache invalidation

Q&A