CSE311
Computer Architecture
Project

Programmable Interrupt Controller

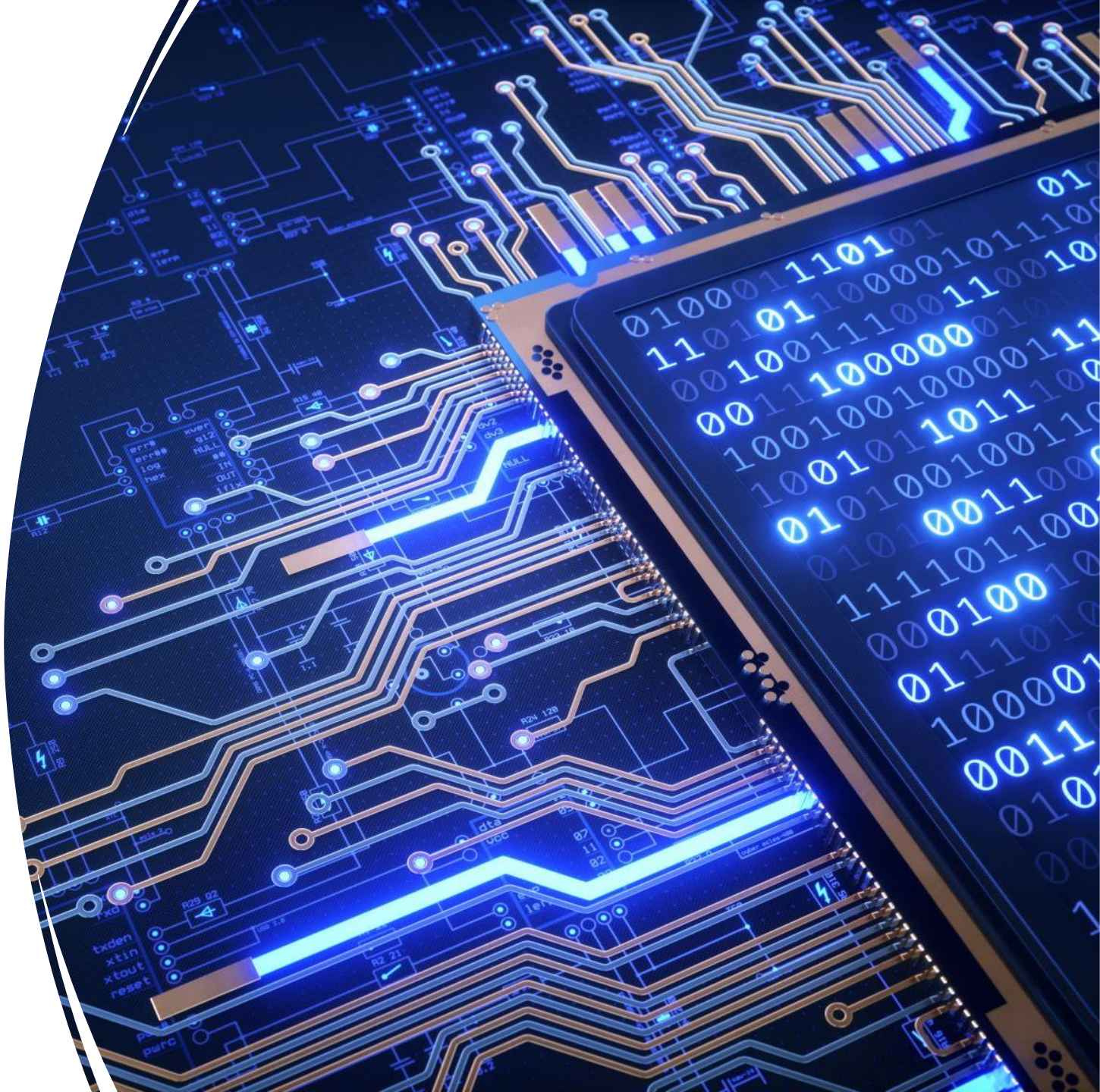
Supervised By:
Dr. Ashraf Salem
TA. Tasneem Adel

- GitHub Repository: link
- Presentation Video: link

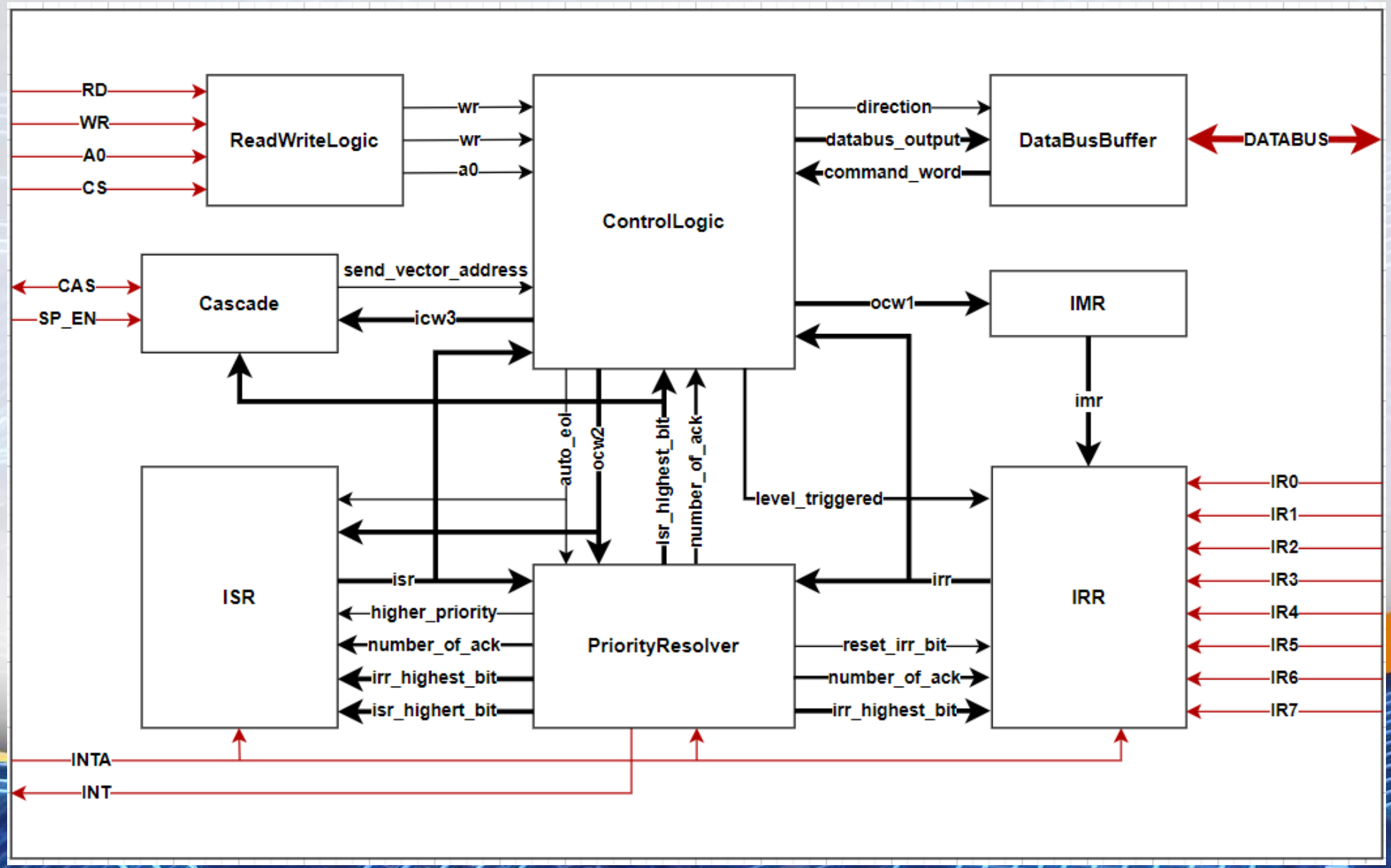
Contributors:

Name	ID	Tasks
Ali Mohsen Yehia Ateya	2000289	PriorityResolver
Amr Essam Mahmoud Anwar	2001089	ControlLogic
Ahmed Adel Abdelrahman	2001778	TestBench
Nassar Khaled Masoud	2001464	IRR & IMR
Ahmed Sherif Mohamed	2001547	Cascade, DataBusBuffer & ReadWriteLogic
Adham Mohamed Mohamed	2001378	ISR

Table of Contents



Block diagram



Block diagram

- **ControlLogic:** This module is responsible for receiving the ICWs and OCWs to store them, and sending the databus_output to DataBusBuffer and enabling it to write on the data bus.
- **IRR:** This module is responsible for storing the raised interrupts from the devices through level detection or edge detection, and resetting them when they are handled.
- **IMR:** This module is responsible for storing the masked interrupts.
- **PriorityResolver:** This module is responsible for deciding the highest priority interrupt to send from IRR to ISR, handling rotation, and raising the INT signal to the CPU in case of interrupt.
- **PriorityLevel:** This module is responsible for outputting the highest priority bit and its level from IRR or from ISR based on the values in priority register.

Block diagram

- **ISR:** This module is responsible for storing the currently serviced interrupts and resetting them after serving whether immediately in case of AEOI or after the command in case of normal EOI.
- **DataBusBuffer:** This module is responsible for driving the data bus when sending the vector address or the CPU want to read a register, and reading the command words from the CPU.
- **Cascade:** This module is responsible for deciding if the chip is master or slave, and driving the CAS lines in case of master, or reading the CAS lines in case of slave to decide if the chip will send vector address or not.
- **ReadWriteLogic:** This module is responsible for reading the WR and RD signals from the CPU and enabling them with CS.

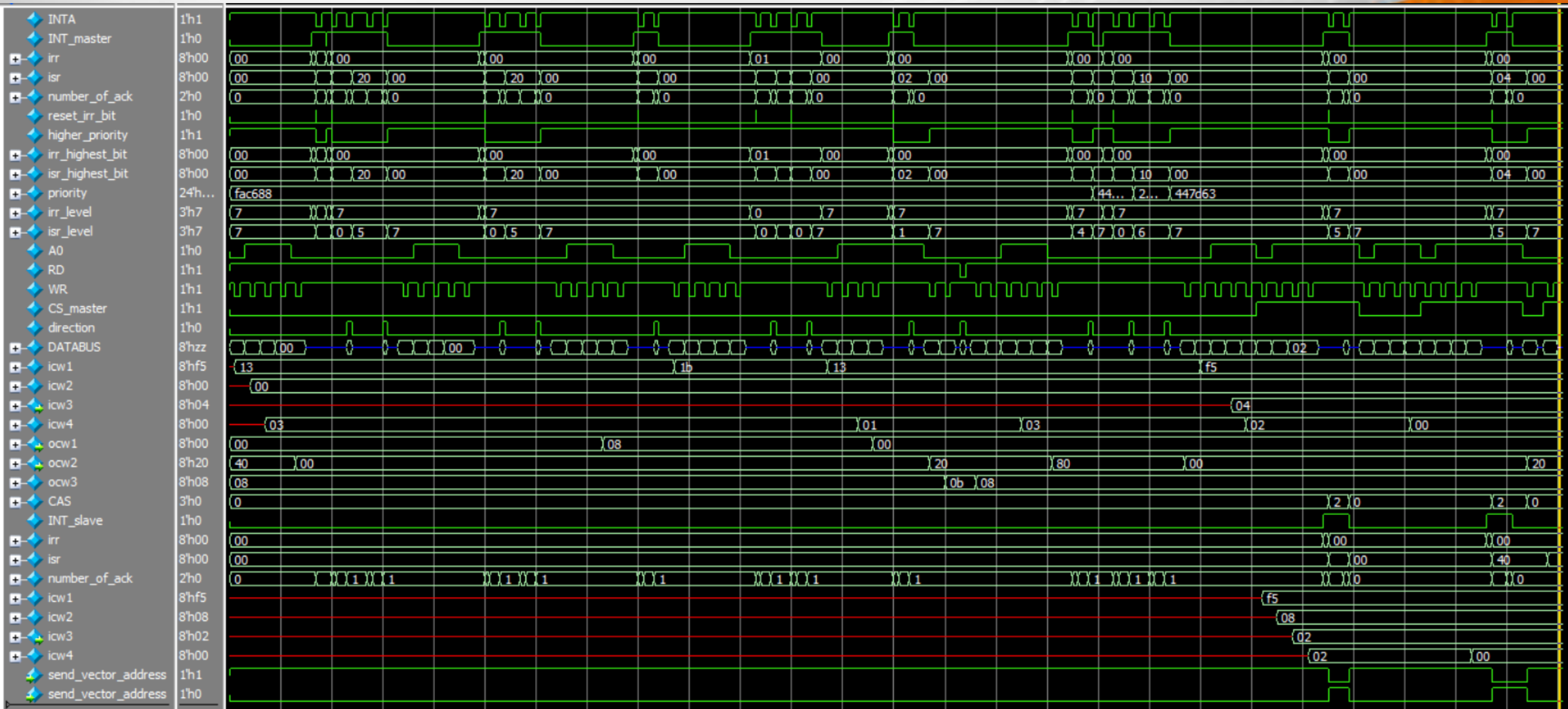
Description of internal signals

Signal	Its Function
wr	It’s the write signal after checking on chip select signal
rd	It’s the read signal after checking on chip select signal
direction	It’s a control signal for tri-state buffer that allows driving the bus
databus_out	The data that is to be written on the data bus
command_word	The received data from the data bus
send_vector_address	In master mode: it enables sending vector address in case that the interrupt is from a device and not a slave PIC In slave mode: it enables sending vector address in case that the CAS lines are the save as the slave ID.
number_of_ack	It indicates the current number of acknowledgement pulses received
reset_irr_bit	It reset the highest level irr bit that will be served
higher_priority	It checks if there is a higher priority interrupt than the highest one in isr

Description of internal signals

Signal	Its Function
irr_highest_bit	It is a 8’bit register that holds the highest level in irr bit only
isr_highest_bit	It is a 8’bit register that holds the highest level in isr bit only
auto_eoi	It is bit 1 in icw4, it programs the PIC to work in Auto End of Interrupt
level_triggered	It is bit in icw1, it programs irr to work in level triggered mode
irr	Interrupt request register
isr	Interrupt service register
imr	Interrupt mask register
isr_highest_bit	It is a 8’bit register that holds the highest level in isr bit only
auto_eoi	It is bit 1 in icw4, it programs the PIC to work in Auto End of Interrupt
level_triggered	It is bit in icw1, it programs irr to work in level triggered mode

Wave Forms:


INTA 1'h1
INT_master 1'h0
irr 8'h00
isr 8'h00
number_of_ack 2'h0
reset_irr_bit 1'h0
higher_priority 1'h1
irr_highest_bit 8'h00
isr_highest_bit 8'h00
priority 24'h...
irr_level 3'h7
isr_level 3'h7
A0 1'h0
RD 1'h1
WR 1'h1
CS_master 1'h1
direction 1'h0
DATABUS 8'hzz
icw1 8'hf5
icw2 8'h00
icw3 8'h04
icw4 8'h00
ocw1 8'h00
ocw2 8'h20
ocw3 8'h08
CAS 3'h0
INT_slave 1'h0
irr 8'h00
isr 8'h00
number_of_ack 2'h0
icw1 8'hf5
icw2 8'h08
icw3 8'h02
icw4 8'h00
send_vector_address 1'h1
send_vector_address 1'h0

Test strategy:

Inputs	Expected output	Feature
Test Case (1) Initialize ICWs (edge/AEOI) IR5=1 INTA pulse IR0=1 4 INTA pulses	Vector address of IR0 will be sent then vector address of IR5	Handling interrupt with higher priority occurred during an interrupt between two INTA pulses (AEO)
Test Case (2) Initialize ICWs (edge/AEOI) IR0 = 1 & IR5 = 1 4 INTA pulses	Vector address of IR0 will be sent first then IR5	Normal priority
Test Case (3) Initialize ICWs (edge/AEOI) OCW3 to mask IR3 IR3=1 & IR7=1 2 INTA pulses	Vector address of IR7 only is sent	Masking feature

Test strategy:

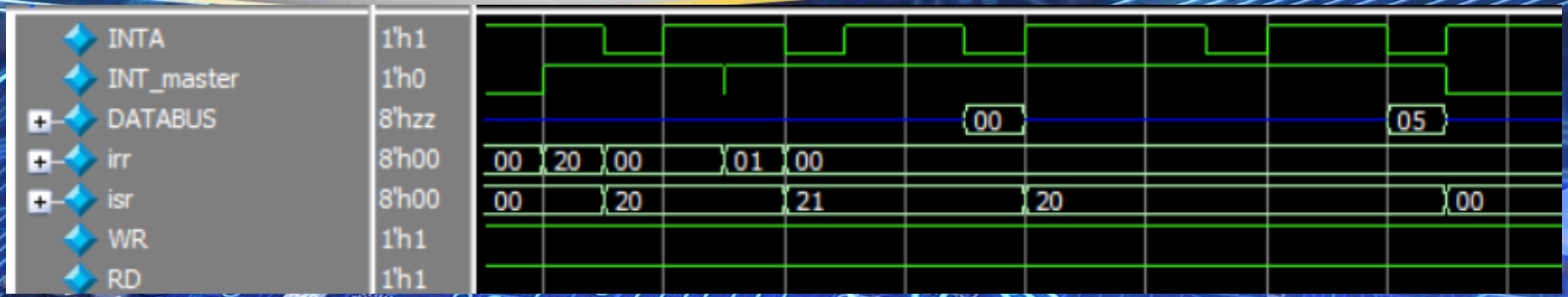
inputs	Expected output	feature
Test Case (4) Initialize ICWs (level/AEOI) IR0 = 1 4 INTA pulses	Vector address of IR0 will be sent twice	Level Triggered
Test Case (5) Initialize ICWs (edge/normal EOI) IR1 = 1 2 INTA pulses OCW2 EOI command OCW3 read ISR operation	Vector address of IR1 will be sent then the value in isr will be sent.	Non specific EOI Read operations
Test Case (6) Initialize ICWs (edge/AEOI) OCW2 for automatic rotation IR4 = 1 2 INTA pulses IR4 = 1 & IR5 = 1 4 INTA pulses	Vector address of IR4 will be sent then IR5 then IR4	Automatic rotation

Test strategy:

inputs	Expected output	feature
Test Case (7) Initialize ICWs for Master and slave, (edge/AEOI) IR6_slave = 1 2 INTA pulse	The slave will send the vector address of IR6_slave	Cascade AEOI
Test Case (8) Initialize ICWs for Master and slave, (edge/ Normal EOI) IR6_slave = 1 2 INTA pulse OCW2 EOI command twice	The slave will send the vector address of IR6_slave	Cascade normal EOI

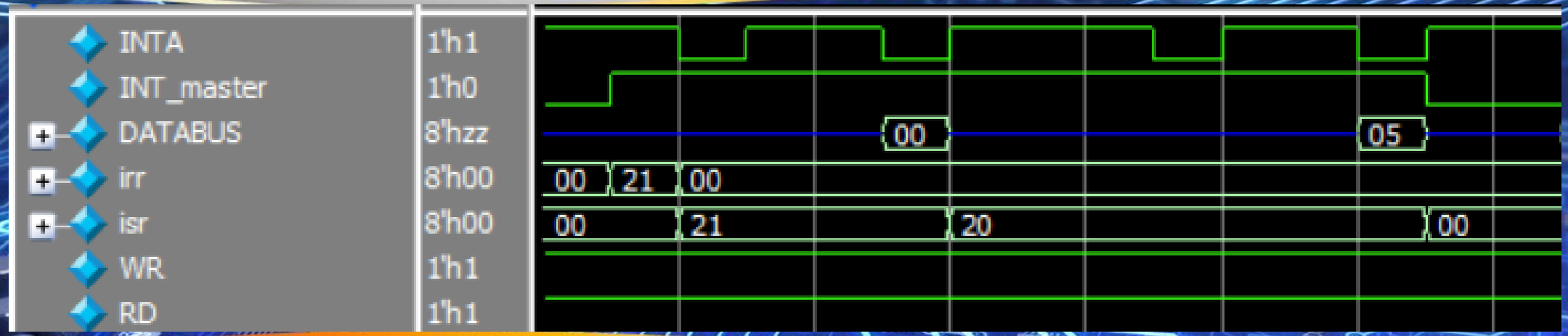
(1) First test in input strategy:

Handling interrupt with higher priority occurred during an interrupt between two INTA pulses (AEO)



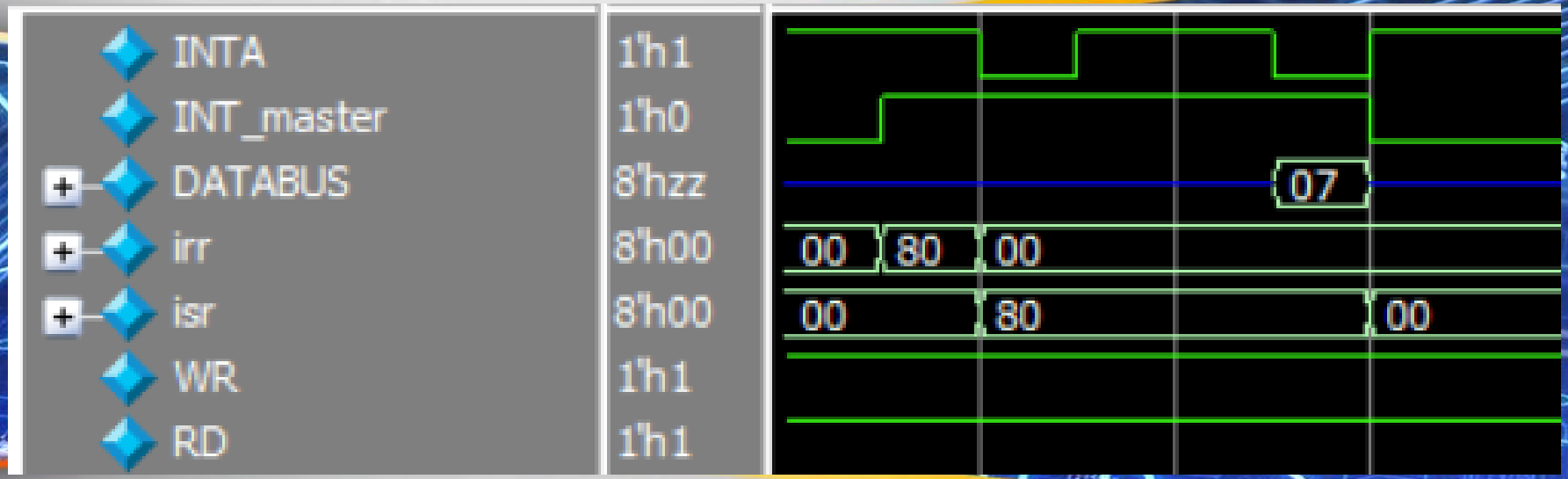
(2) Second test in input strategy:

Normal priority



(3) Third test in input strategy:

Masking feature

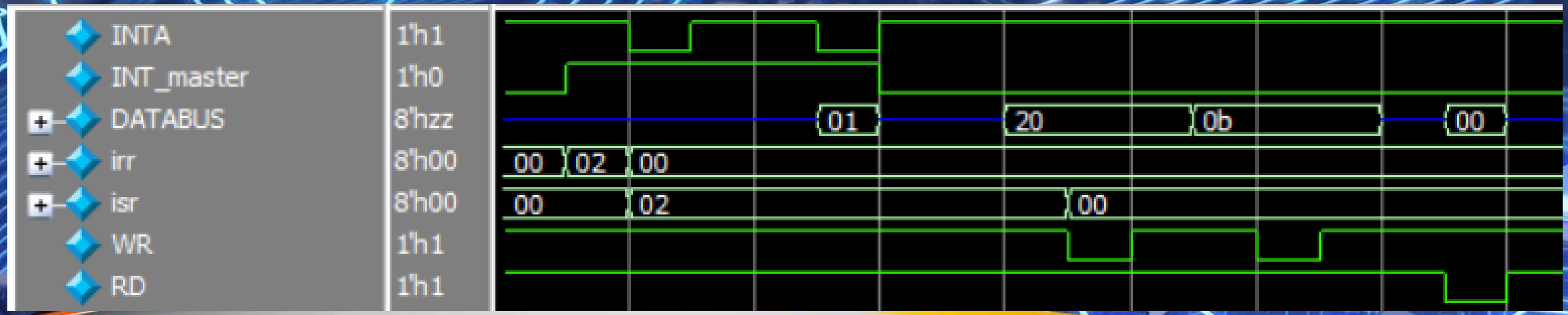


(4) Fourth test in input strategy:

Level Triggered

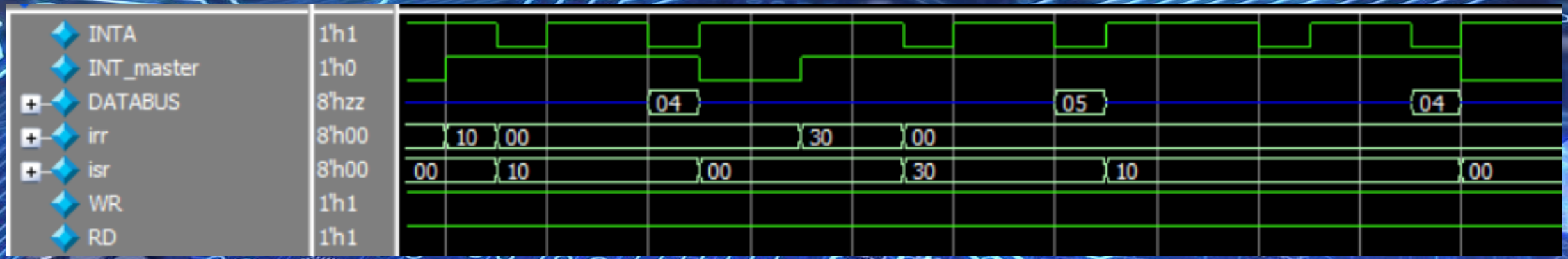
(5) Fifth test in input strategy:

Non specific EOI
Read operations



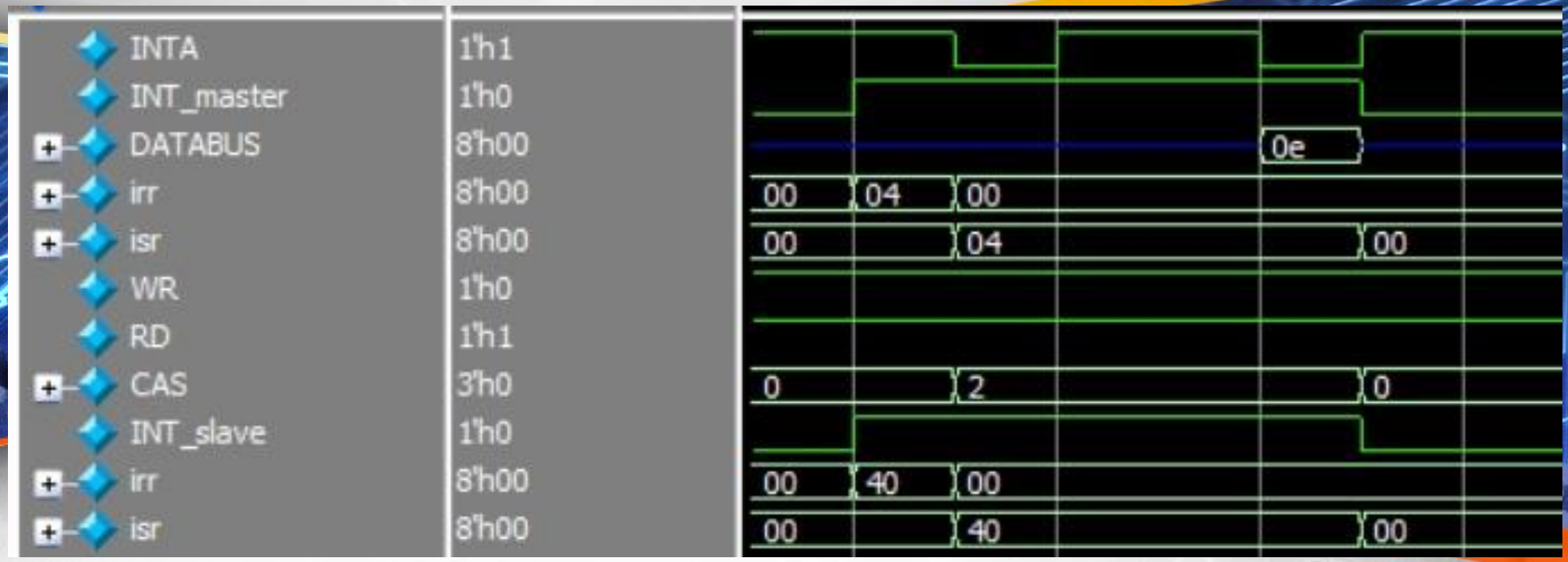
(6) Sixth test in input strategy:

Automatic rotation



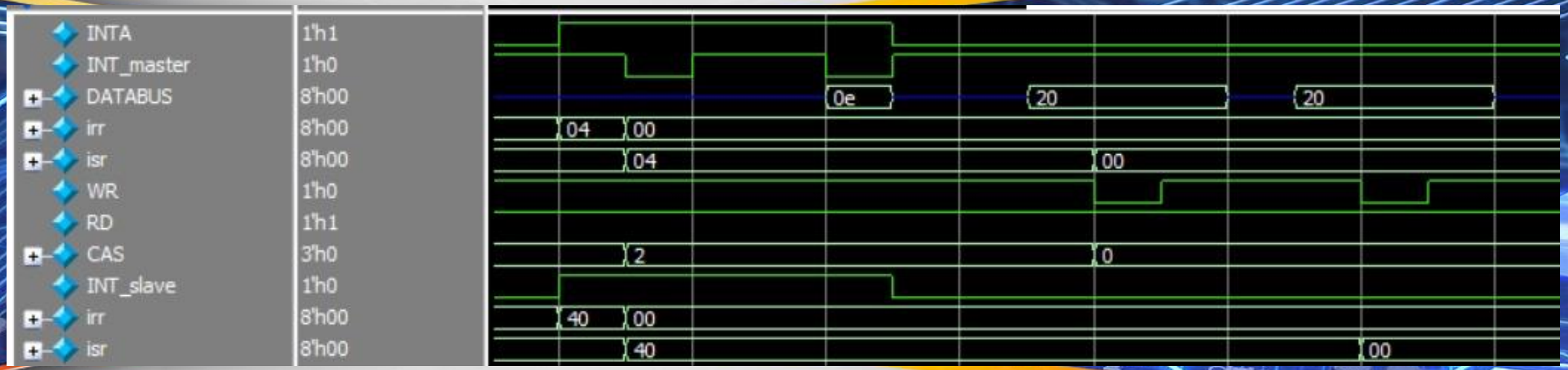
(7) Seventh test in input strategy:

Cascade AEOI



(8) Eighth test in input strategy:

Cascade normal EOI



Enhancement:

A Moore FSM to receive initialization command words in internal registers: icw1, icw2, icw3, icw4
Synchronized with WR pulse.
We used one-hot encoding because the state can jump between non-sequential states