EC551

Lab2

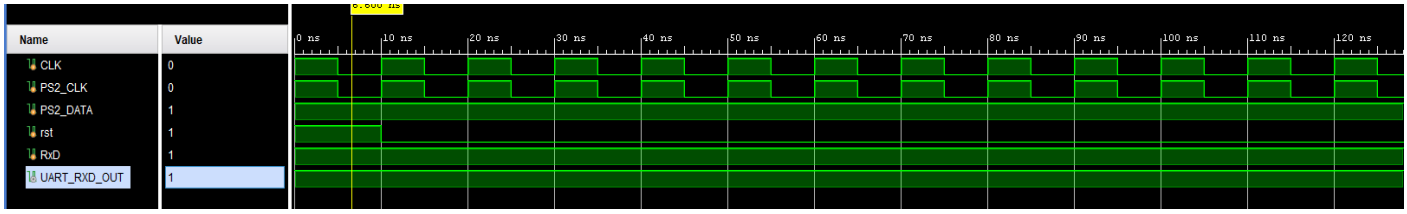
11/11/2022

Bin Xu

Yimin Xu

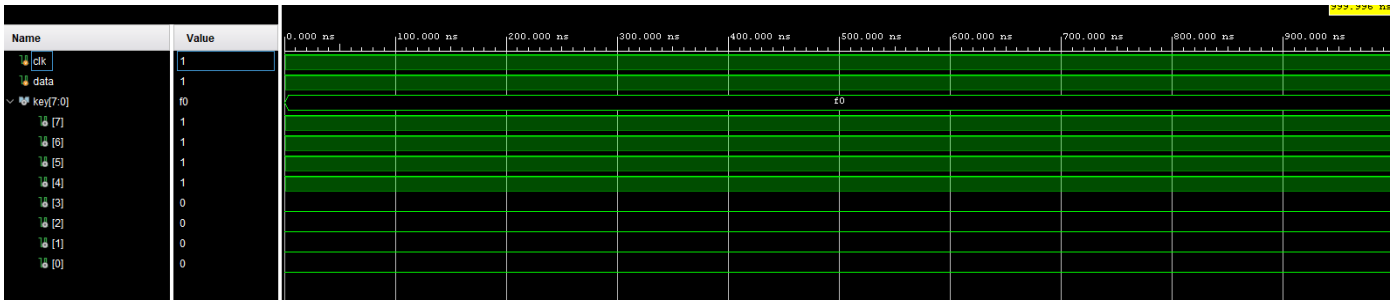
Top.v

In the top module, we connect to the keyboard and uart_tx controller. In the keyboard part, we need to transfer the keyboard input to the ascii for the pc to read the input.



Uart_tx_control.v. we need it This module implements the FIFO buffer. The FIFO buffer output is connected to the input buffer. The data is passed from logic, then the tx controller will output the data to the terminal.

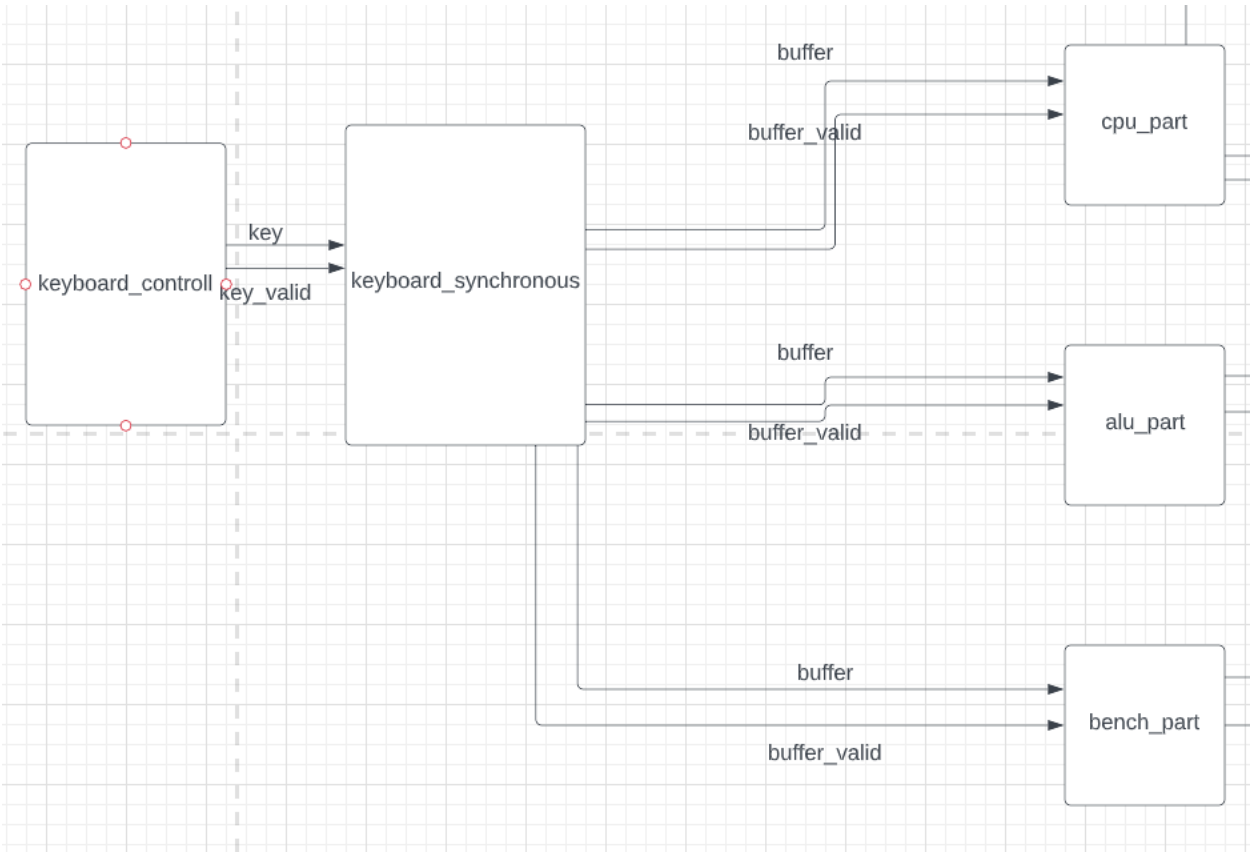
PS2Receiver.v kb_code_ascii_convert.v debouncer.v



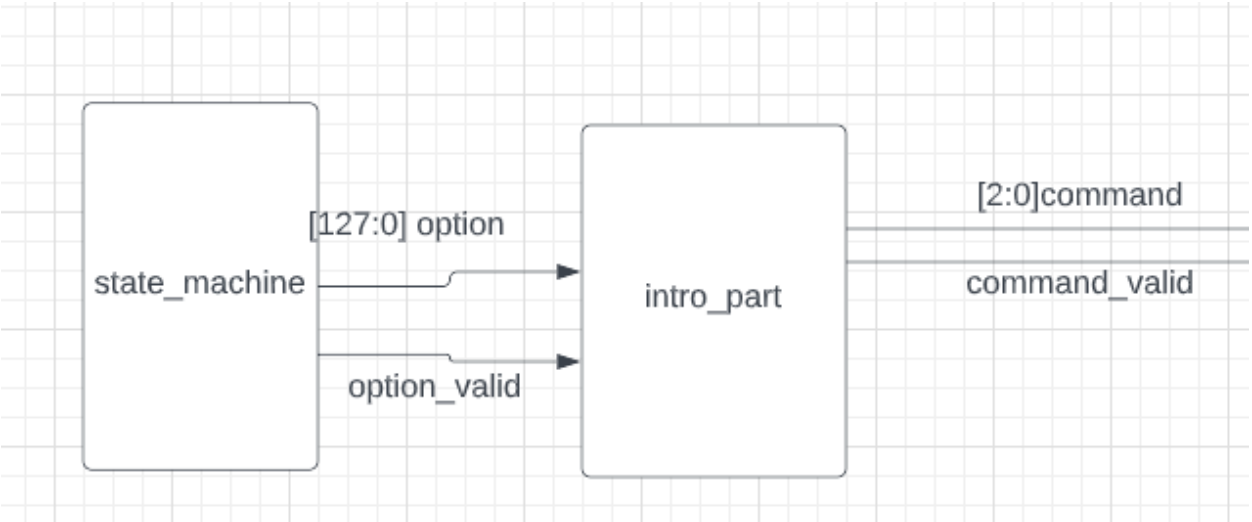
Our ps2keyboard uses the example code in the lab manual. Also we wrote the keyboard ascii convert to ascii. When the previous keyboard code change from not 8'hf0 to 8'hf0, the number will be transferred to top.v.

scheduler.v, in this module we need to figure out the state transfer when we enter different inputs. Firstly, we need to state_input, which is passed to the terminal. We also need state_string, state_execution. For the state_execution, we need to include intro part, alu part, cpu part and

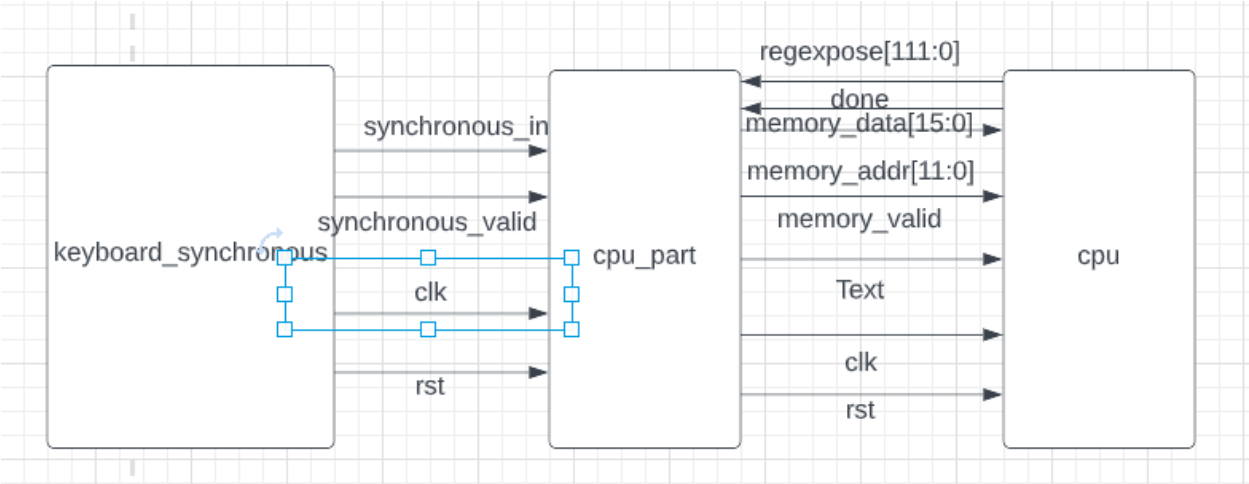
bench part. In every state_execution, we need to update state_input, state_string. For state_string, we need to cover input_blocked, input_allowed. For state_string, we match the state from the printer, which outputs the corresponding content to the terminal.



Keyboard_synchronous.v, this module is connected to internal_processing. This module is for queuing up single line inputs. After getting the character from keyboard_controller, the triggered character needs to be queued up. The queue_valid needs also be passed by controller. The space for the queue is 16. Then different parts, like alu_part, cpu_part, bench_part, will take the input from **Keyboard_synchronous.v.** There is a rst button passed by. It will be passed to state_machine.



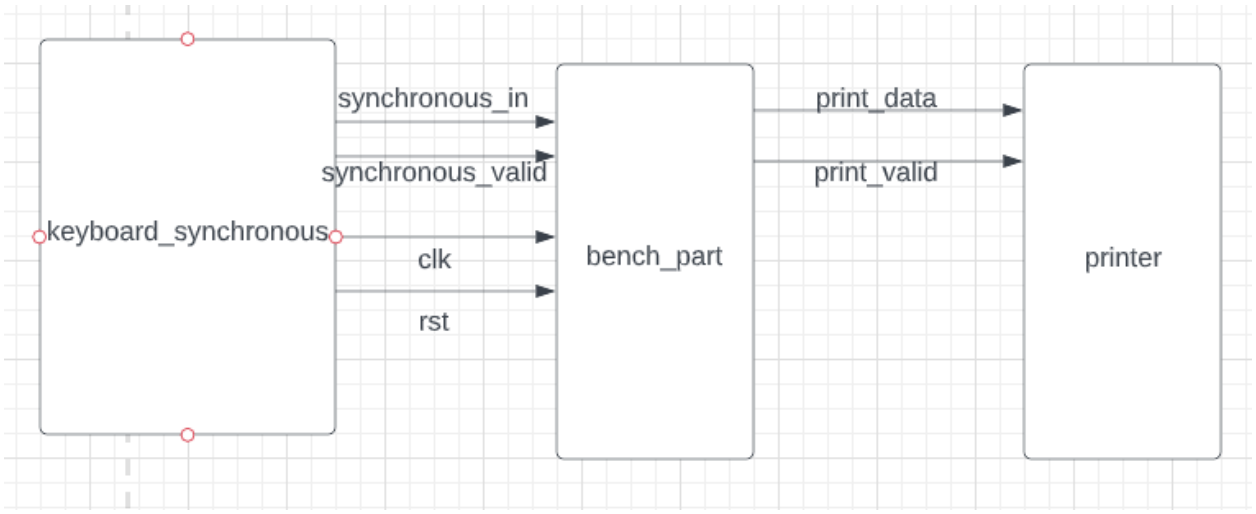
Introduction_part.v, at the beginning The state_machine will set **input blocked state**. This module is for controlling the different modules after the “hello ec551” There are three modules that can be chosen.



Cpu_part.v. Cpu_part needs to be connected to CPU logic from Lab1. We need to add the memory address, memory data and memory enable to the memory part, because we do not need to read the data from .txt in Lab2. The cpu part module decodes the input from the keyboard_synchronous, then writes them into the memory.

Instruction is entered as hex form. Every line is an address. We capitalized the character in hex input. The execution starts from address 31. Once the CPU finishes the execution, the pc counter will be outputted.

The memory cannot be overwritten during the execution.



Bench_part.v. In this module, we need to address the multiplication of matrices.

Reference:

https://github.com/Digilent/Nexys-A7-100T-Keyboard/blob/master/src/hdl/PS2Receiver.v

https://github.com/FPGAwars/FPGA-peripherals/blob/master/uart-rx/uart_rx.v