LAB 3 REPORT

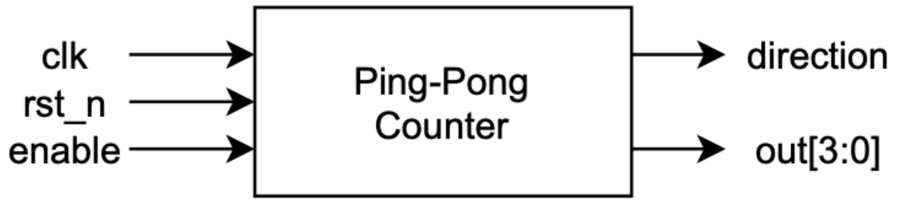
4-bit Ping-Pong Counter, First-In First Out (FIFO) Queue, Multi-Bank Memory, Round-Robin FIFO Arbiter, 4-bit Parameterized Ping-Pong Counter

FPGA Implementation on 4-bit Parameterized Ping-Pong Counter

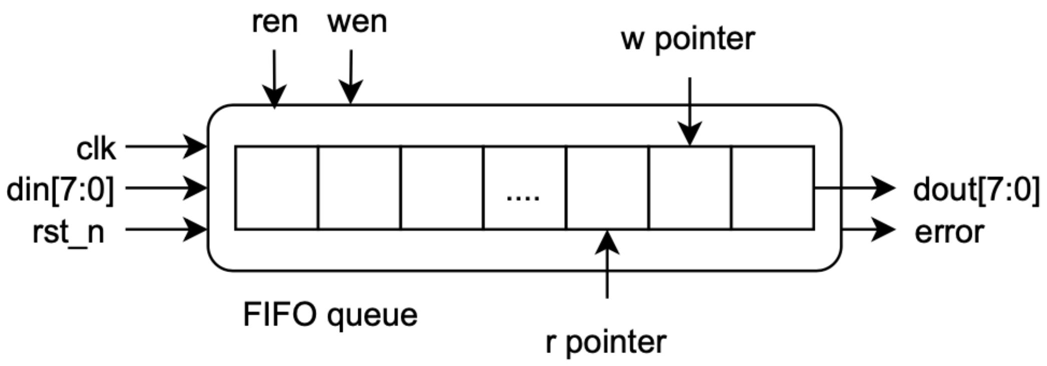
Team 37:

1. 徐美妮 Mary Madeline Nicole 109006205
2. 林之耀 Kevin Richardson Halim 109006277

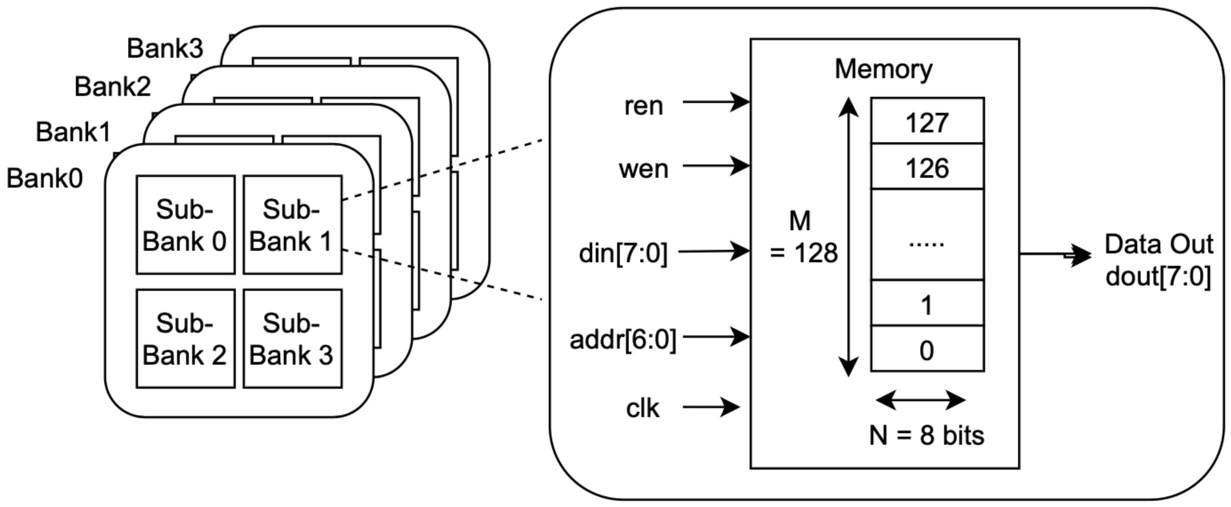
I. Gate-Level Circuits



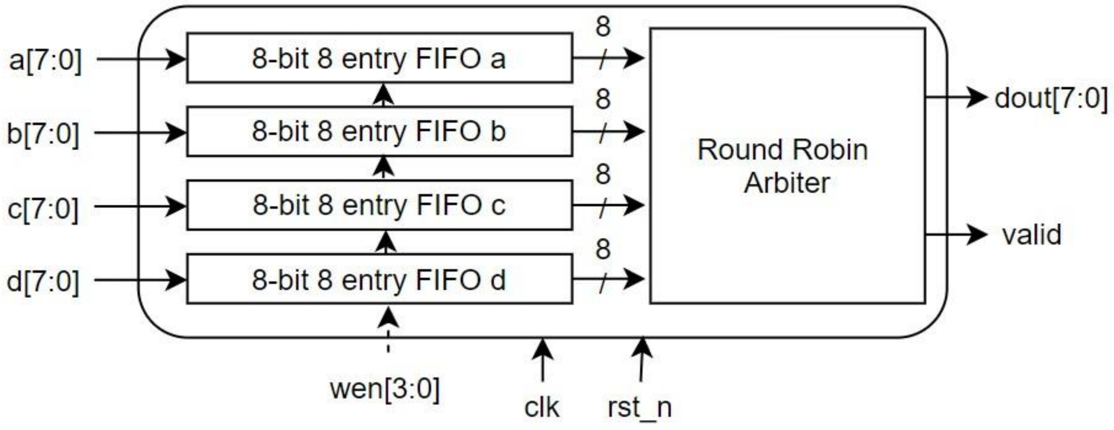
1.1 4-bit Ping-Pong Counter



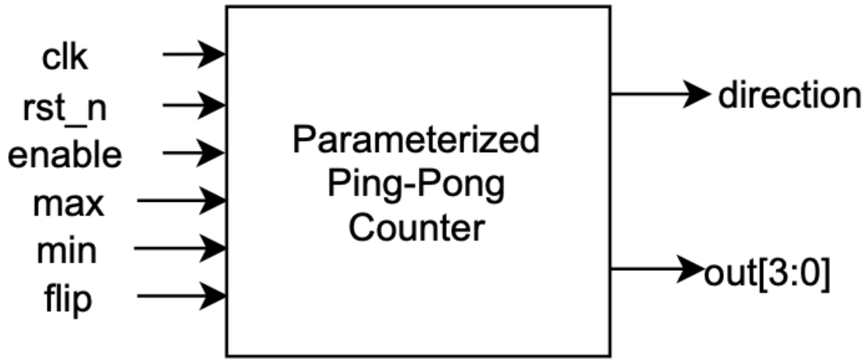
1.2 First-In-First Out (FIFO) Queue



1.3 Multi-Bank Memory with 128x8 Memory Array



1.4 Round-Robin FIFO Arbiter



1.5 4-bit Parameterized Ping-Pong Counter

II. Design Process

First, we start by drawing and sketching the design. We also slice the codes into smaller modules to create a bigger one hierarchically.

We implemented each module using sequential and combinational circuits. For sequential circuits, we define the reset condition, and assign the next state to the current state. Then, we state updates at the next clock edge just like a counter. For combinational circuits, instead of using gate-level modeling, we used continuous assignments and always block. In the always block, we prioritize reset signal over the rest of the input.

Then we check all spelling, syntax, and errors.Next, the testbench is built. The testbench is very useful to check whether or not the module works as it should. Moreover, by using the clock it is easier to manage and simulate timing. The use of display and task when making the testbench makes it easier to spot mistakes at our module, by using the CAD server. The testbench that we made covers the edge cases and some basic inputs. Specifying the timescale of the simulation which are the time unit and precision enables it to run fully on vivado. Critical conditions, or edge cases were also made, relying on when the module would break.

Finally, debugging and reiterating each line to fix some bugs that were faced. Such as wrong I/O, swapped variables, wrong naming, wrong behavior, typos, etc.

Modules:

1. **4-bit Ping-Pong Counter**

   The Ping-Pong counter is a synchronous, up.down binary counter. Where all flip flops are clocked simultaneously so that the outputs change with each other during a certain clock cycle. The counter will reset its position and direction if rst_n = 1, and start its operation when it is enabled. The counter will add its number when direction is 1, and decrease when direction is 0. When the counter reaches the maximum count, it will then decrease and repeat up and down, until the enable is 0, it will stay and hold its value.

2. **First-In-First Out (FIFO) Queue**

   FIFO, first in, first out, a method of organizing the manipulation of a data structure where the first entry is processed first. We created a circular FIFO that stored eight entries of 8-bit datas. The order of the read follows the FIFO pattern, where the first data written would be read out first. If both write enable and read enable are performed, only read operation goes. When a read is issued to an empty FIFO, or when a write is issued to a full FIFO, it will result in error. We created the fifo by implementing a write pointer and a read pointer to know the address of write and read respectively. During read, the read pointer will add its address by one, but when it is at the edge of the queue, and the write pointer is at the beginning of the queue, we mark it as empty. During write, the write pointer will add its address by one and will mark it as not empty. These pointers will be the key to detect whether or not the queue is empty or full.

3. Multi-Bank Memory with 128x8 Memory Array

Memory array is a collection of registers to mimic memory arrays. The memory array used in this module is from the basic question. Memory is a two dimensional storage element which stores M numbers of words. And each element is N-bit wide. These memories enable signals for read and write. We access the memory by address register.

This multibank memory contains 4 banks of memory where each bank consists of 4 sub-bank memory modules. Each sub-bank memory module contains 4 memories. We used a combinational circuit to assign the dout and decode the write and read on respective banks.

4. Round-Robin FIFO Arbiter

Four FIFOs from the previous module were used to connect it with a round robin arbiter. There are 4 states, where the read-state can change from a to d (a,b,c,d). We add the state by 1 every clock cycle and decode from read state to read enable. Then, when one of the FIFO has errors, it will be invalid and the output will be 0. We also assign dout according to the valid and FIFO state, and control it so it would be either 0 or 8 bits data.

5. 4-bit Parameterized Ping-Pong Counter

The parameterized Ping-Pong counter was made by adding more ports to the Ping-Pong Counter. We have to parameterize the upper bound (max) and lower bound (min). The concept of the Counter is exactly like a normal Ping-Pong counter, just more complicated with Parameters. We made sure that the max and min values are valid, otherwise the counter holds its current value. The min will be the starting point of the counter and the max will be the edge of the counter before it flips direction and counts down.

The flip is only one cycle in length and the counter will flip its direction when the flip is true. We had to make sure that the max and min does not violate the rules. Such that the max and min might change during counting, which is why we created another condition for that, when it violates the rules, it will stay and hold its state. Once the value of the counter is out of range, then the value and direction is held.

Because the PPPC has two outputs, out and direction, that update the value when clk is at posedge trigger, so a sequential circuit is used for every posedge clk, changing the out and direction. We set the output out and direction by combinational *assign*.

III. List of Contributions

徐美妮 Mary Madeline Nicole 109006205's contributions, such as :

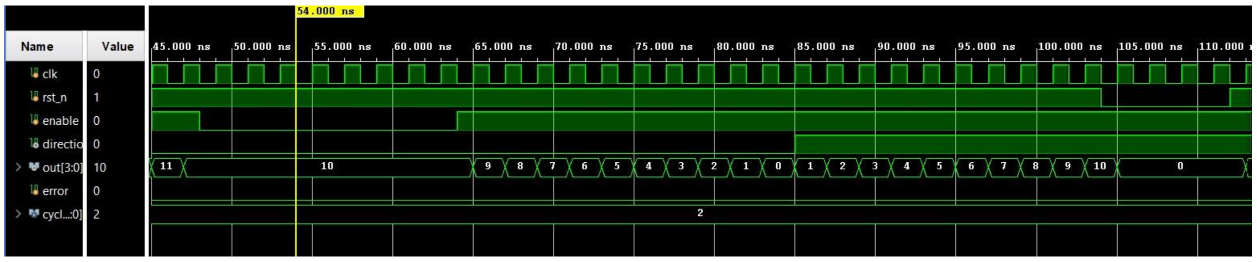
- Design and write module 4 bit Ping-Pong counter,
- Design and write 4-bit Ping-Pong Counter testbench,
- Design and write 4-bit Parameterized Ping-Pong Counter testbench,
- Design and write module 4-bit Parameterized Ping-Pong Counter,
- Design and write FIFO Queue testbench,
- Design and write module FIFO Queue,
- Draw gate level circuit,
- Program the FPGA board,
- Design and write XDC file,
- Write the report.

林之耀 Kevin Richardson Halim 109006277's contributions, such as :

- Design and write testbench 4- bit Ping-Pong Counter,
- Design and write module Multi-Bank Memory,
- Design and write Multi-Bank Memory testbench,
- Design and write module FIFO Queue,
- Design and write FIFO Queue testbench,
- Design and write module Round-Robin FIFO Arbiter,
- Design and write Round-Robin FIFO Arbiter testbench,
- Design and write 4-bit Parameterized Ping-Pong Counter testbench,
- Simulate and synthesis modules in Vivado,
- Program the FPGA board,
- Design and write XDC file,
- Write the report.

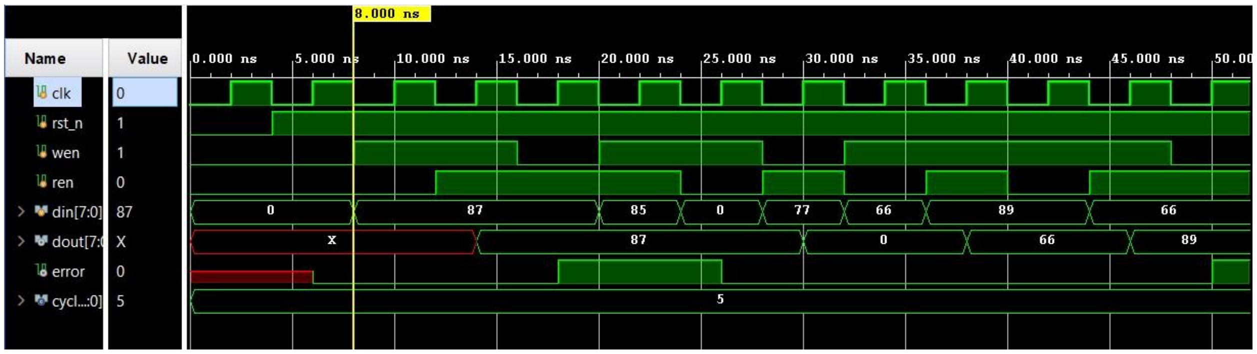
IV. Design Test

For every testbench, we set inputs into reg and outputs into wire. We also create a cycle for a neater and constant time cycle. Where we set the clock into 1 and 0 consecutively according to the time delay using the cycle.



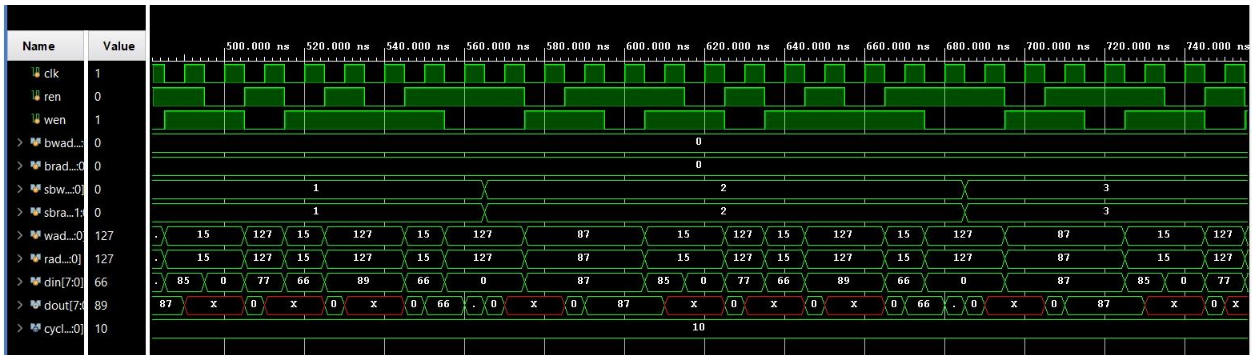
4.1 4-bit Ping-Pong Counter

We tested out Ping-Pong Counter by first initializing all of the inputs. First, we test when the reset is 1, enable is 1. Then set the enable to 0, and did it repeatedly. Then we observe the waveform of the counter. And the result of the waveform was right.



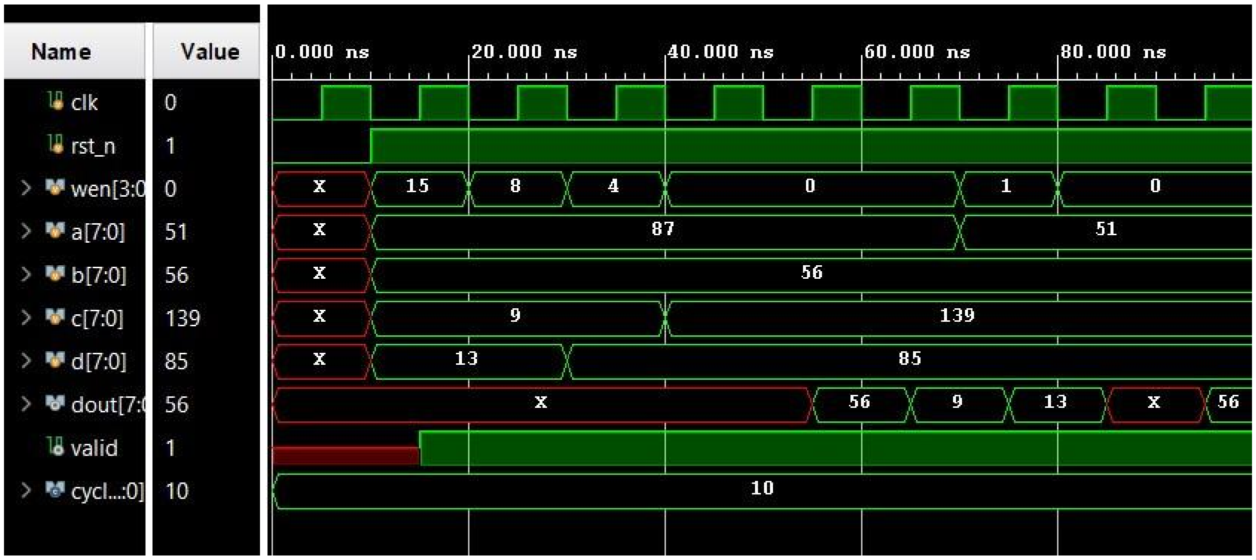
4.2 First-In-First Out (FIFO) Queue

For the FIFO Queue, we made some cases to test every negative edge of the clock, with the combination of a random input number, and combinations of 2 bits for wen and ren. To specify, the testbench contains read before writing, set wen and ren at 1 at the same time before writing, then set wen 1 at the same time as wen after writing, then random reading and writing test code.



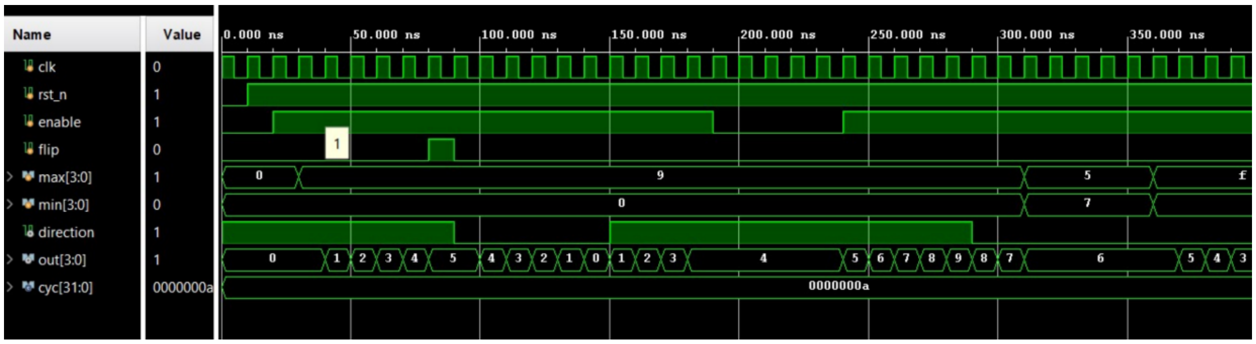
4.3 Multi-Bank Memory with 128x8 Memory Array

The multibank memory was tested by creating an adder for the bank, and adder for the sub bank, also for the address. For this module, we set the cycle = 10. First we initialize all of the adders. For the bank write adder, bank read adder, sub bank write adder and sub bank read adder are set from 2’d0 to 2’d3 respectively. Then, we test for the combinations of 2 bit ren and wen, when read enable, but write disable, then read enable write enable, and all the other combinations.



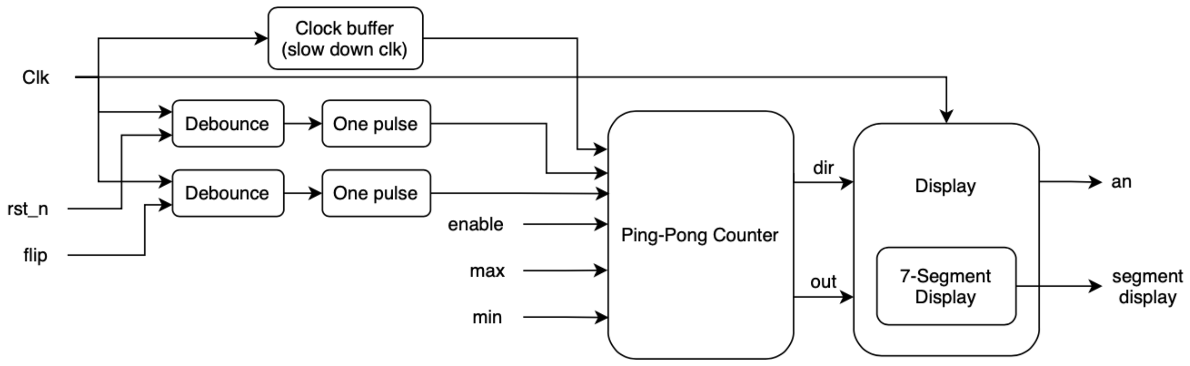
4.4 Round-Robin FIFO Arbiter

Round robin FIFO arbiter was tested by initializing all inputs. Then, we test it by creating our own test cases. We reset the module every negedge clock. Then assign random numbers to the inputs, a,b,c, or d. Then we also change the values of each input randomly, while assigning the write enable randomly too. After simulation, the waveforms are according to the right outputs, therefore our program runs fine.



4.5 4-bit Parameterized Ping-Pong Counter

The parameterized ping pong counter was also tested at every negative edge trigger of the clock. At first we enable and initialize everything. Setting the minimum to 0 and maximum to the maximum value of 4 bits. Then, we play around the values of the minimum and maximum. After setting the maximum and minimum, also the flip and resets, we tested it and looked at the waveform in order to make sure that it is correct. Our module worked and creates output as it should, therefore the code works fine.



4.6 Parameterized Ping-Pong Counter on FPGA

To handle Meta-Stability (glitches) after pressing the buttons, We debounced the circuit, where it shifts registers to allow time for signals to stabilize. Also, a one pulse circuit that generates a one-clock-cycle push when the button is pressed. Because a push button is usually pressed for many clock cycles. With this, the circuit is used when we want to trigger it once

To set the 7-segment display control, we set all anodes to Low, because we want all four-digits of the display to light up. Whereas two of them are used to display output, and the other two are for direction. The anodes are maintained to indicate the number of the seven-segment display to be updated, so that the an generated is accordingly. We convert the original output of the ping pong counter into bcd according to the display number, and pass it to the seven segment display. We also don't forget to set the buttons for reset and flip.

To test the FPGA, we play with the switches to determine the max and min, also the reset and flip button.

V. What we have learned from Lab 2

- ➔ The difference between coding sequential and combinational circuits.
- ➔ Understand that mem[addr] will map addr to mem by itself without writing a decoder and other modules to deal with.
- ➔ When the FPGA exhibits weird behaviour, and determines that the logic is correct, we have to double check the inputs, ports, and other internal/external input and outputs are correct.
- ➔ Learn to set constraints for the seven segment display.
- ➔ Using the buttons at the FPGA and making constraints for it
- ➔ Learn about clock divider and delays
- ➔ Implementing Decode Encode in FPGA
- ➔ We have trouble making the Multibank module