Description:

Coordinate between input data (num\_of\_signals\_g) to output data (data\_width\_g)

If input < output use in\_small\_out\_cordinator,

if input = output use in\_equal\_out\_cordinator,

else (input > output) use in\_big\_out\_cordinator.

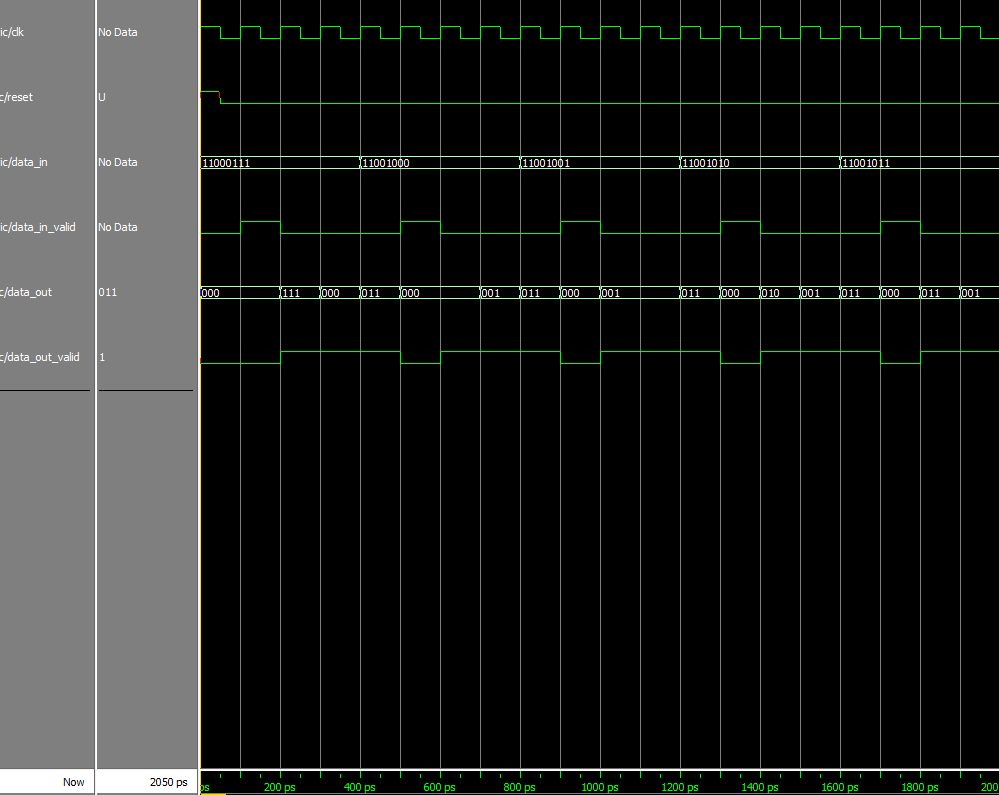
Generic table

|  |  |  |
| --- | --- | --- |
| Name | Width | Description |
| reset\_polarity\_g | 1 | '1' reset active high, '0' active low |
| out\_width\_g | 3 | Width of outputting data |
| in\_width\_g | 8 | Width of incoming data |

Signals table

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Direction | Width | Description |
| clk | In | 1 | System clock |
| Reset | In | 1 | System reset |
| data\_in | In | in\_width\_g | Incoming data |
| data\_in\_valid | In | 1 | Incoming data valid |
| data\_out | In | out\_width\_g | Outputting data |
| data\_out\_valid | In | 1 | Outputting data valid |

Simulation



In that example, the input width is 8 and the output width is 3.

Each time that data\_in\_valid signal rise, the entity sample the input and start to "break" it to output signals in the width of 3. Of course that one "output cycle" take us 3 clock cycle and for that reason we can see that data\_out\_valid signal is high 3 clock cycles each time. The sampled input data is divided in order to keep the value of the original input, meaning that we start outputting the number from the less significant bit (LSB) to the most significant bit (MSB), and if needed (like in that case) we add 0 to the MSB of the last output data.

In our example we divide the input 11000111 to the outputs of: first- 111, second- 000 third- 011. We can see that one 0 was added to the last output in order to fit him to the output width.