NRZI

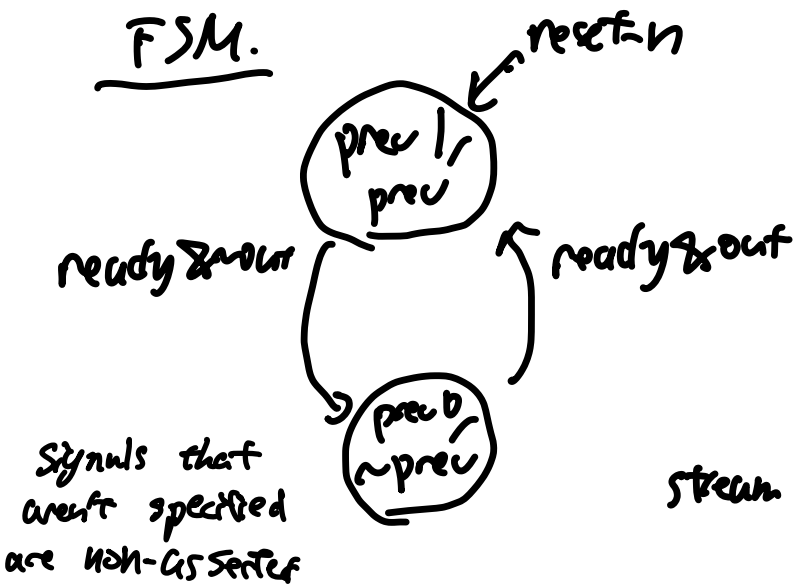
Takes in the bit stream serially, and outputs the processed stream serially. The ready signal is controlled by the overall FSM, and is asserted when sending the SYNC and the packet

ready : Activates the module

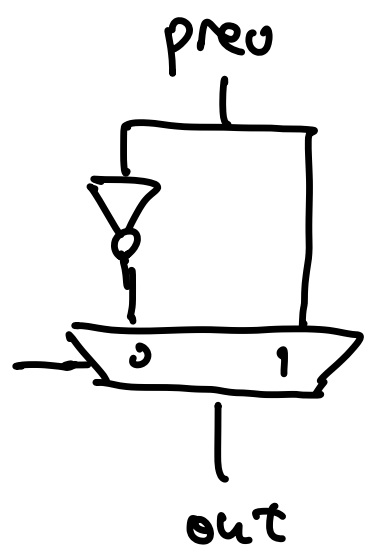
stream: The bit stream

out : The output

FSM.



Datapath



control point: prev

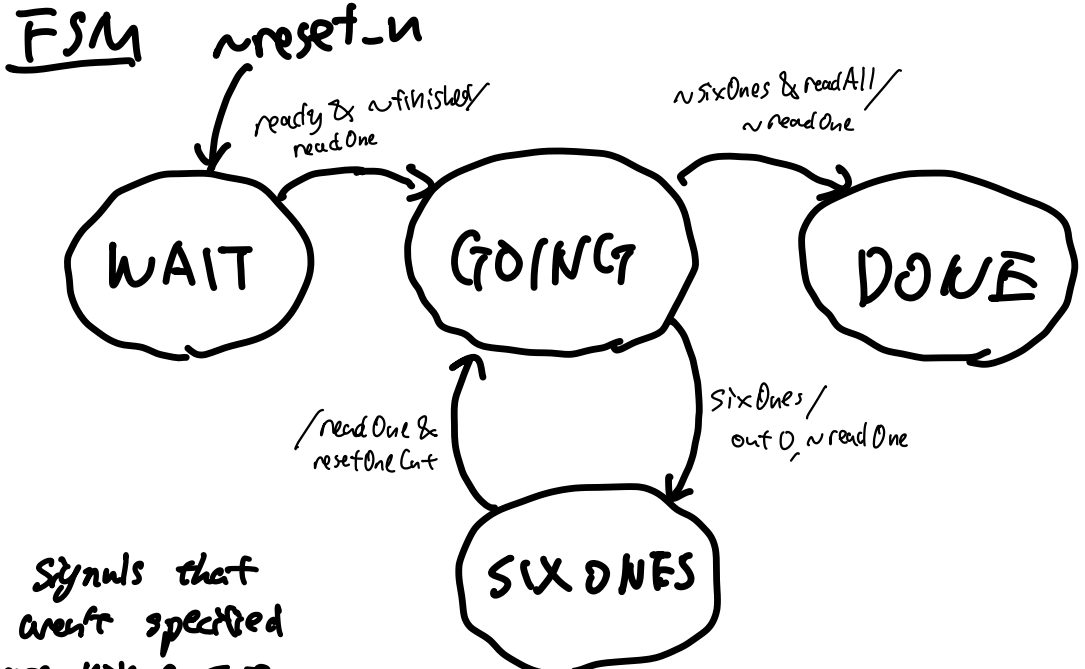
status point: out

Bit Stuffer

Takes in a parallel 24 bit input, and outputs a bitstream that has been bit stuffed. The ready signal is controlled by the overall FSM, and is asserted when the actual data is being sent, not when SYNC is being sent.

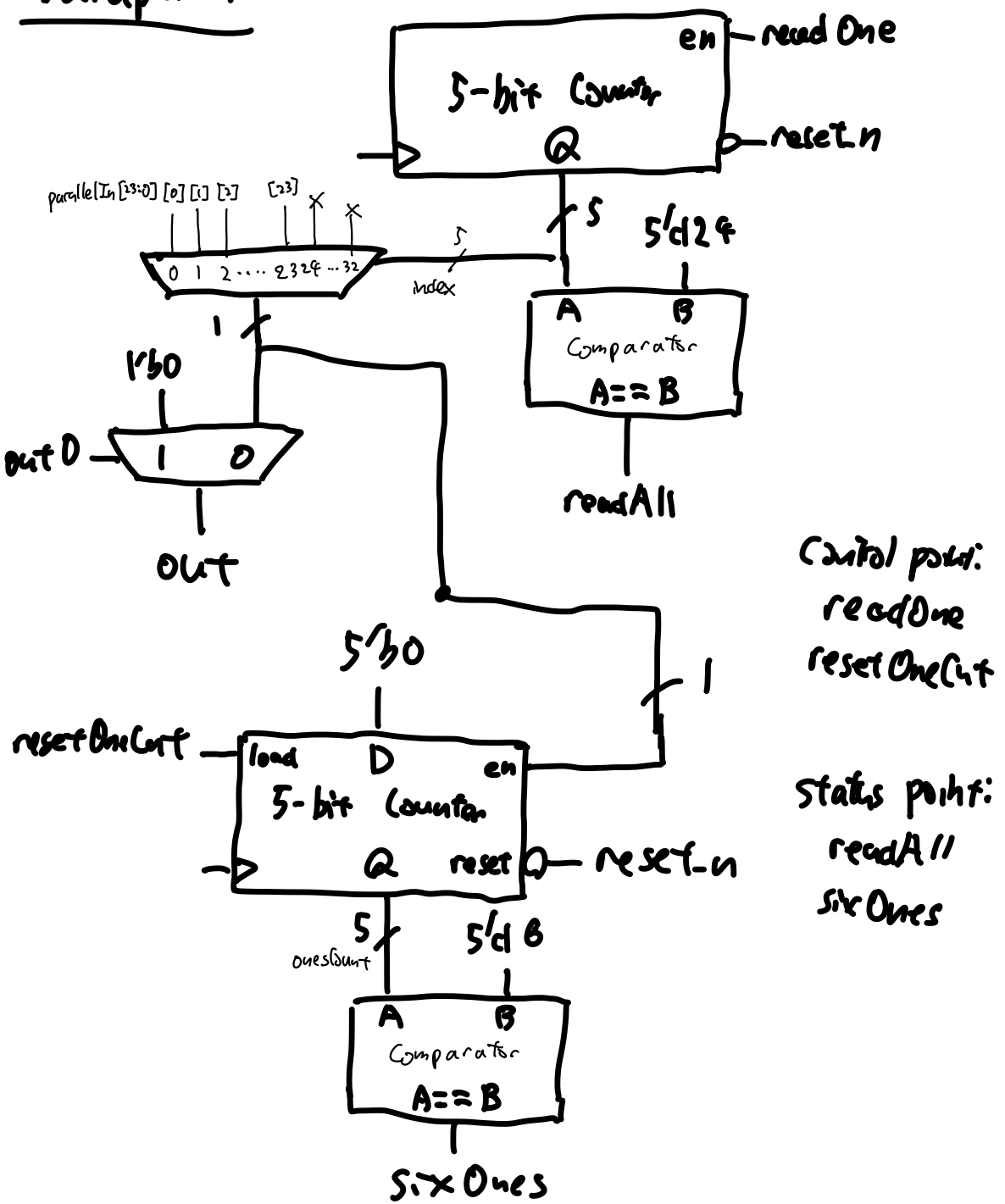
- ready: Activates module
- parallelIn: The 24-bit input
- out: Output
- finished: Asserted when done processing the input

FSM



Signals that aren't specified are non-asserted

Datapath


5-bit Counter
en
Q
reset_n
readOne
parallelIn[23:0]
0 1 2 ... 23 24 ... 32
5
index
5'd24
A
B
Comparator
A == B
readAll
1'b0
out0
1
0
out
1
5'b0
resetOneCnt
load
D
en
5-bit Counter
Q
reset
reset_n
5
oneCount
5'd6
Comparator
A == B
SixOnes
Control port:
readOne
resetOneCnt
Status port:
readAll
SixOnes

CRC5

Takes in 11 bits (7-bits for address, 4-bits for endpoint, both have to be reversed), and calculates the CRC5. It will get calculated at the same time as sending the SYNC. The ready signal is controlled by the overall FSM, and is asserted when sending the SYNC.

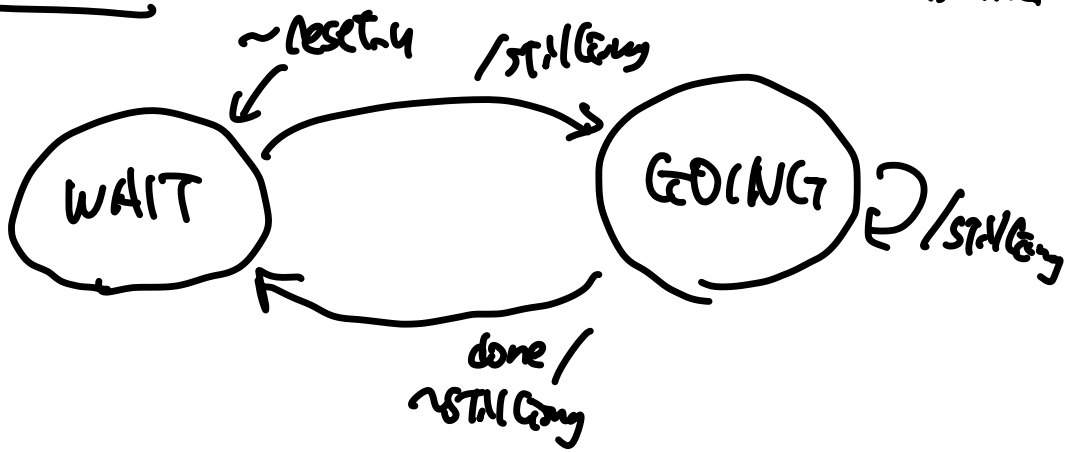
ready: Activates module

parallelIn: 11 bit input

out: 5 bits

FSM

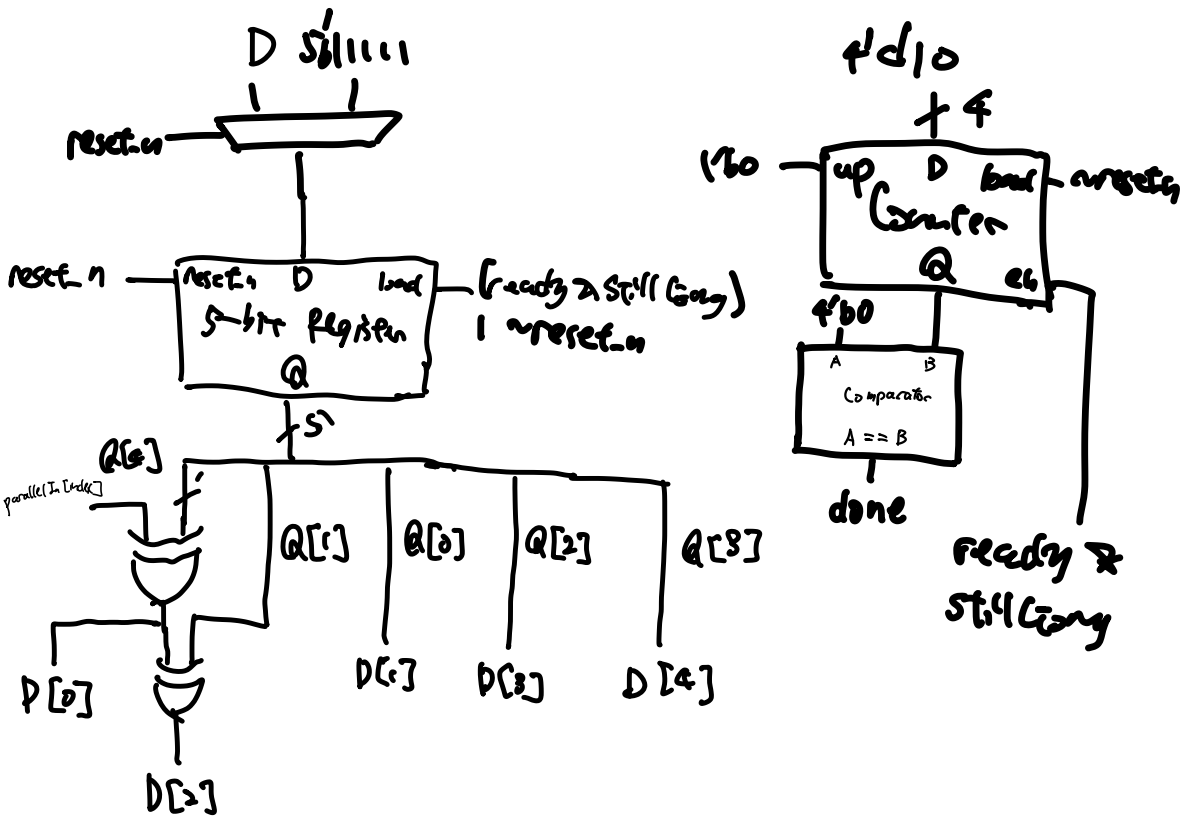
Signals that aren't specified are non-asserted



control port: stillGoing

Status port: done

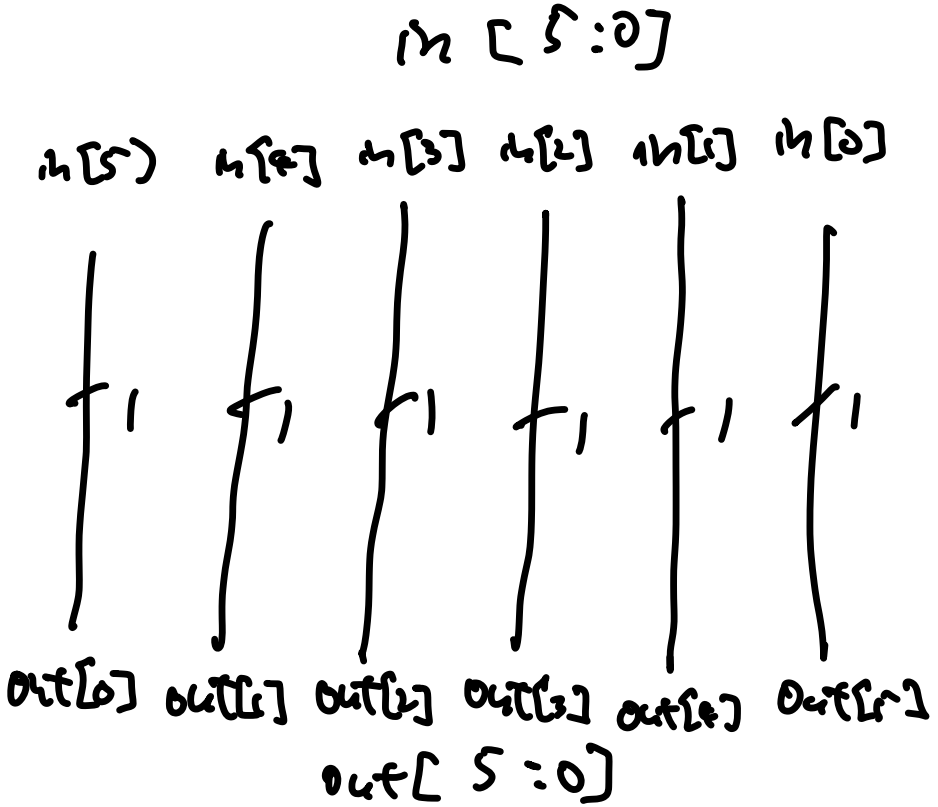
Datapath


D 5'b11111
reset_n
reset_n
reset_n D load
5-bit Register
Q
{ready & Still Going}
| reset_n
5'
Q[4]
parallel_In [4:0]
Q[1]
Q[0]
Q[2]
Q[3]
D[0]
D[1]
D[3]
D[4]
D[2]
4'd10
+4
4'b0
up
D
load
reset_n
Counter
Q
eq
A
B
Comparator
A == B
done
ready & Still Going

Reverse

Reverses an input.

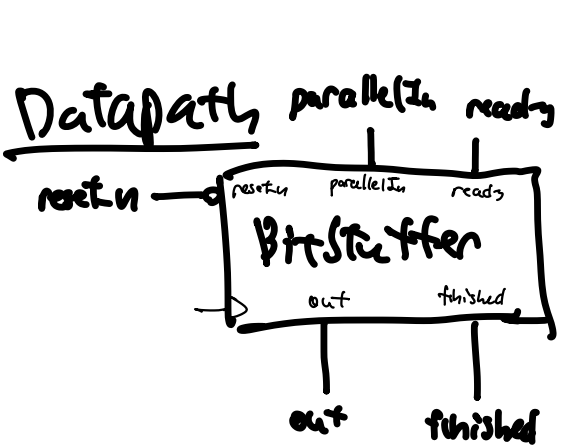
Usage is changing MSB first to LSB first and vice-versa. It is parameterized, so the below diagram is for 6-bit signals

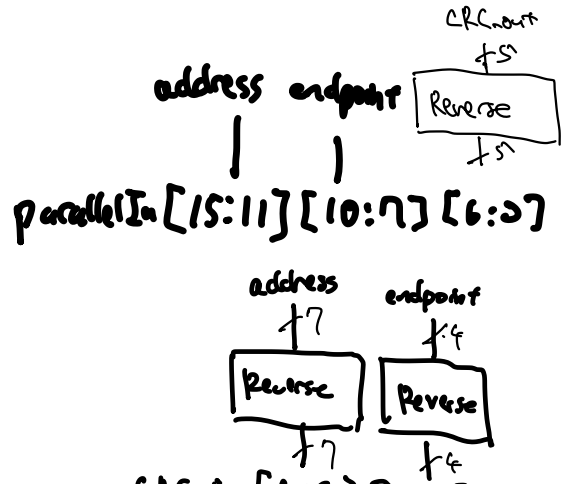


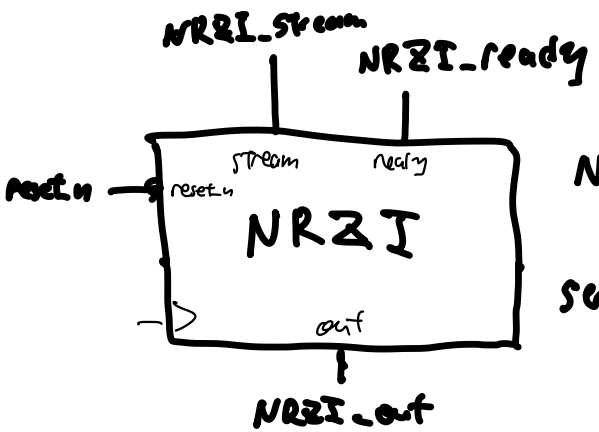
OutPacket

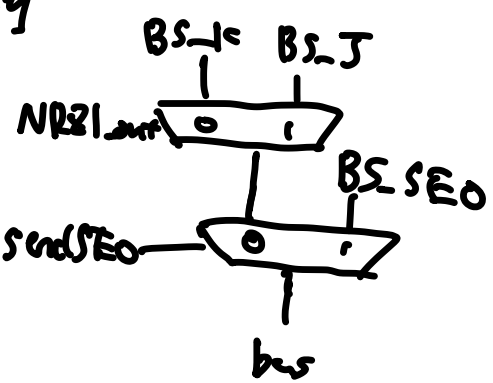
OutPacket structures a OUT packet and sends input to the modules based on its FSM (referenced as overall FSM in the previous descriptions).

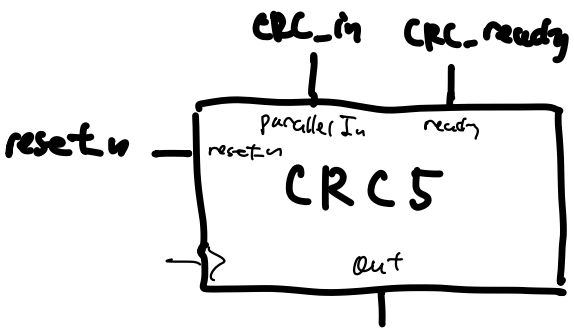
Datapath

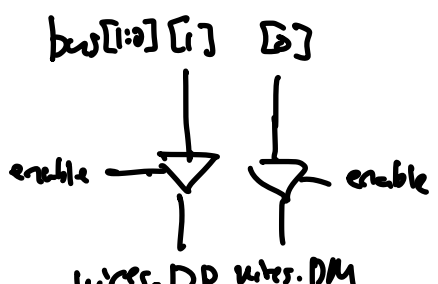




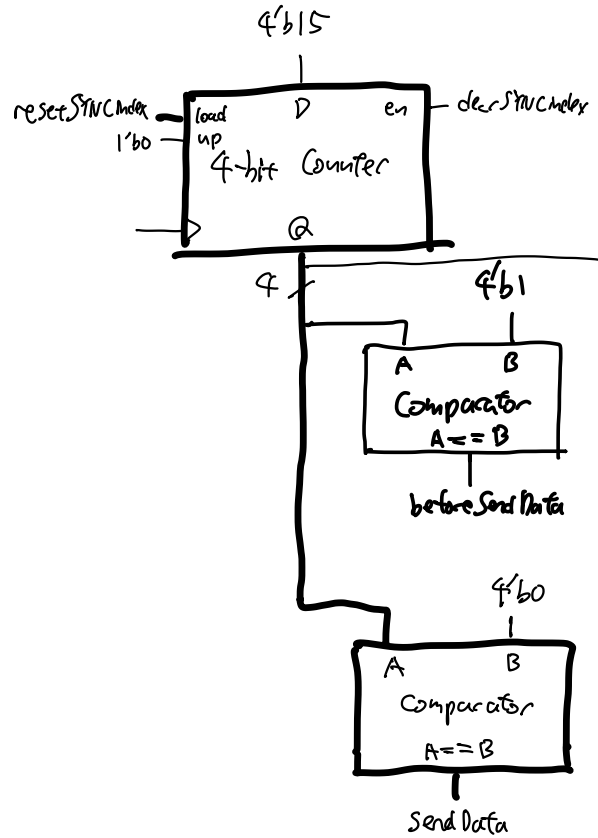


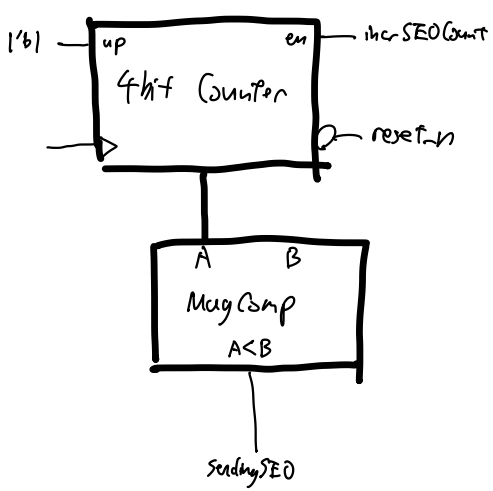






Datapath



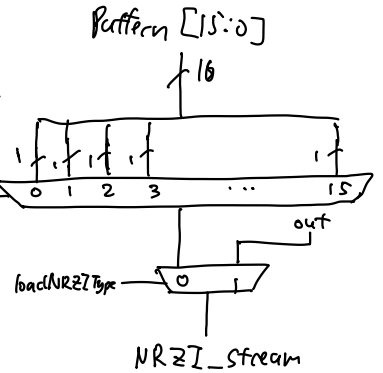


8'b0000_0001 8'b0101_1010

8 8

[15:8] [7:0]

Pattern [15:0]



Control points:

resetSYNCIndex,
loadNRZIType
incrSEOCount
sendSEO
CRC_ready
NRZI_ready
ready

Status port:

sendData
sendingSEO
beforeSendData


FSM
~reset_n /
enable=0
reset SYNCindex,
finishedOut=0
sendSEO=0
idle=0
WAIT
startOut
START
/CRC_ready,
NRZI_ready
/enable=1,
localNRZIType=0
clear SYNCindex
SendSYNC
Signals that aren't specified are non-asserted except for CRC_ready, NRZI_ready, and ready
before SendData,
ready=1
sendData & ~finished
SendData
~finished/
localNRZIType=1
/idle=1
finishedOut=1
enable=0
finished &
sendingSEO
sendingSEO /
incrSEO_count,
sendSEO=1
SendSEO
IDLE
~sendingSEO