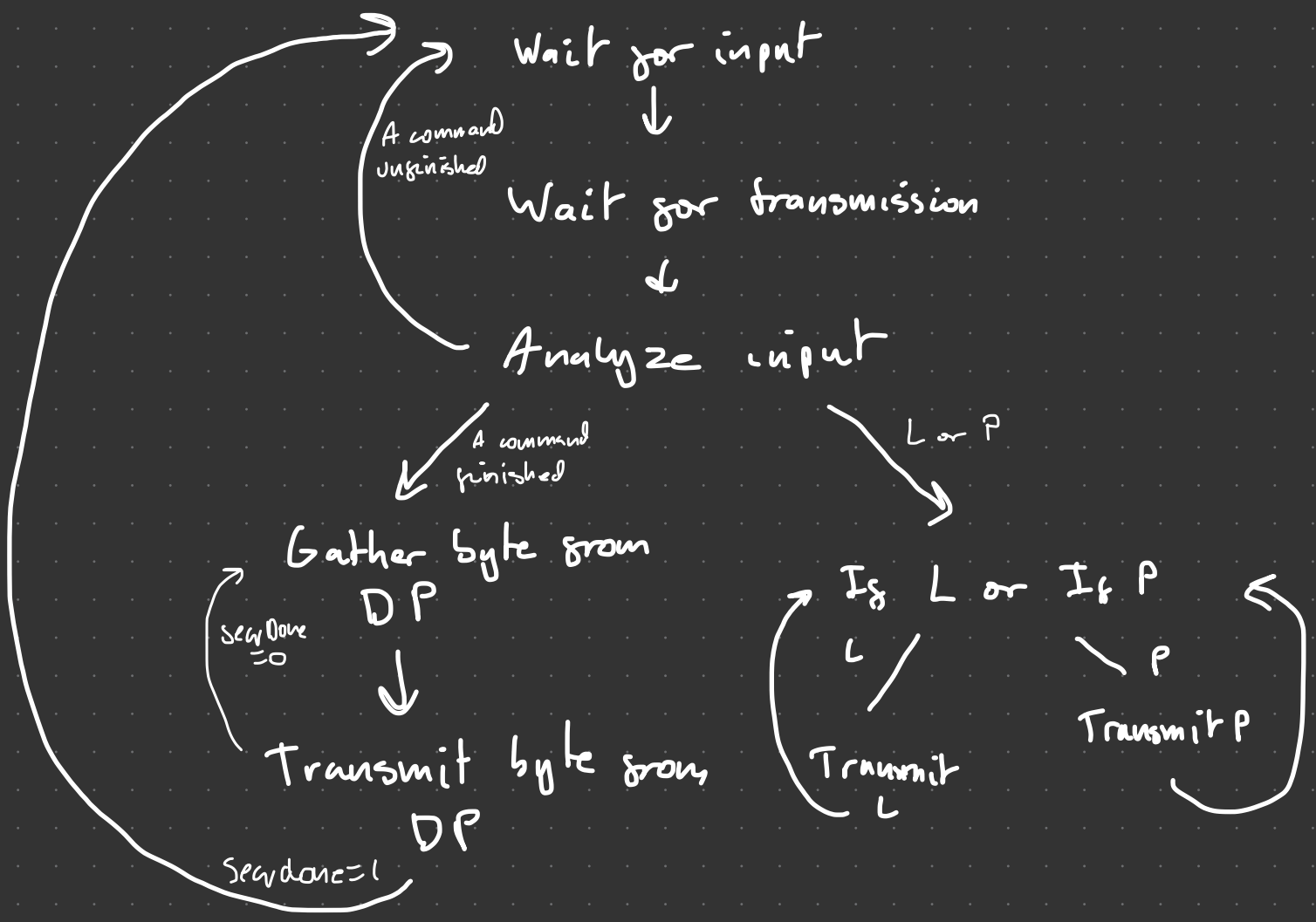
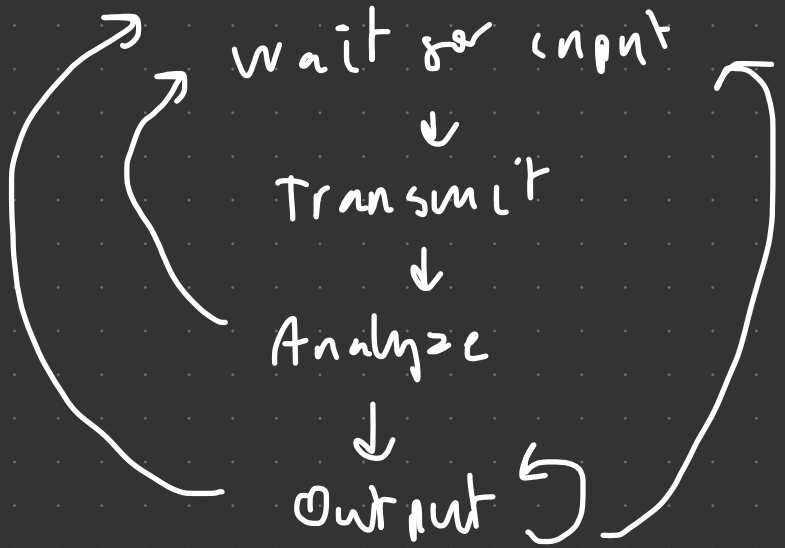
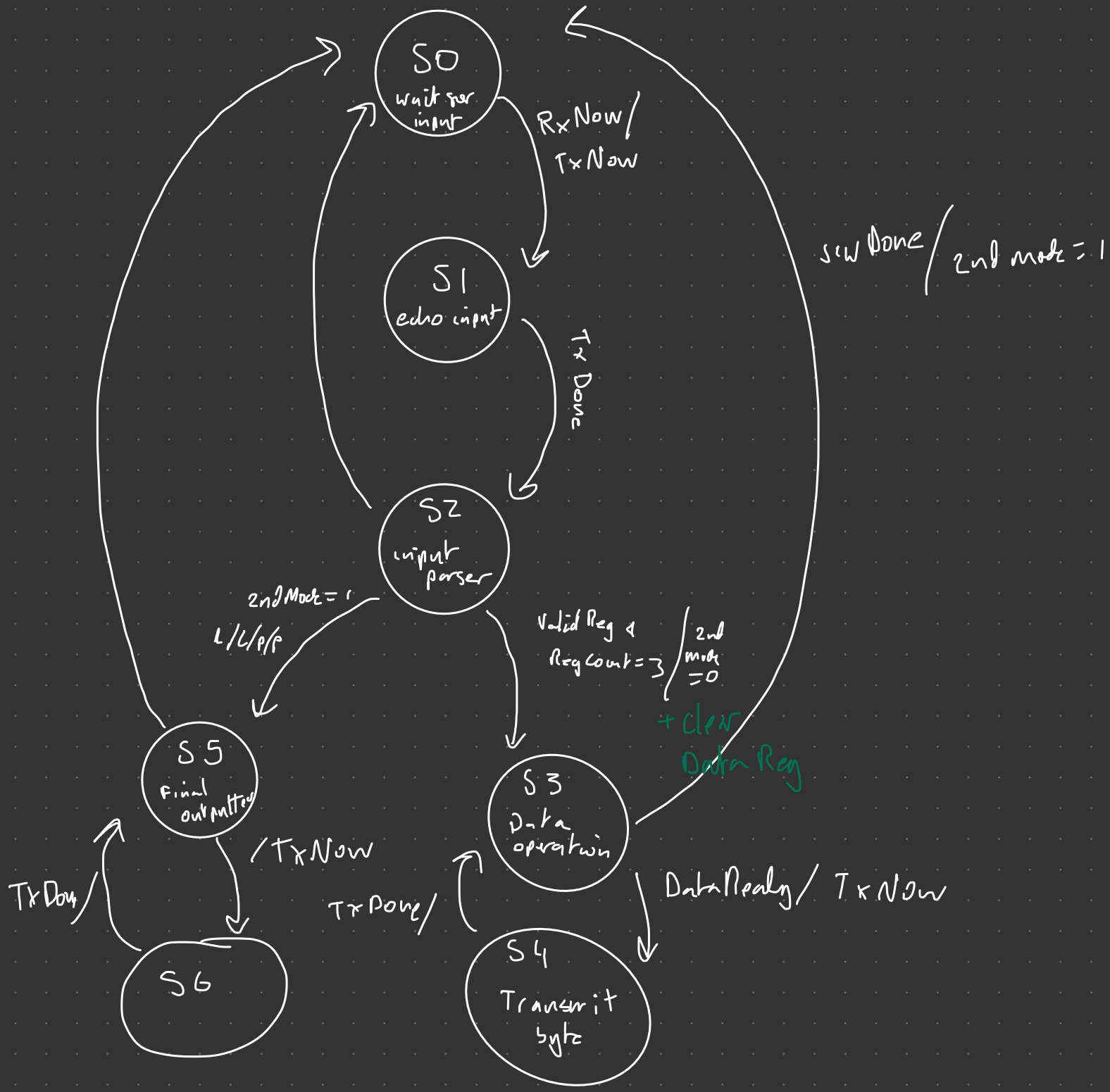
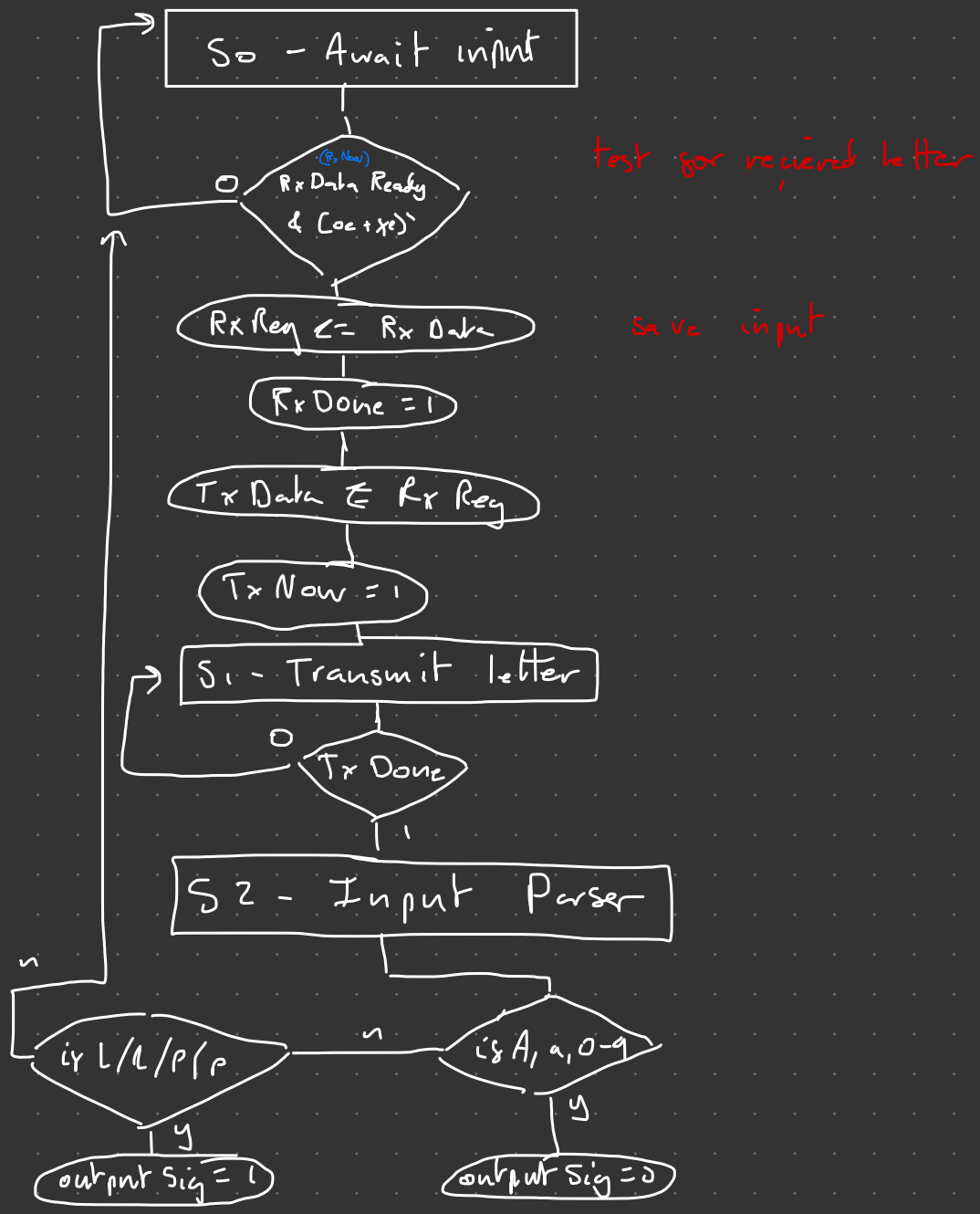




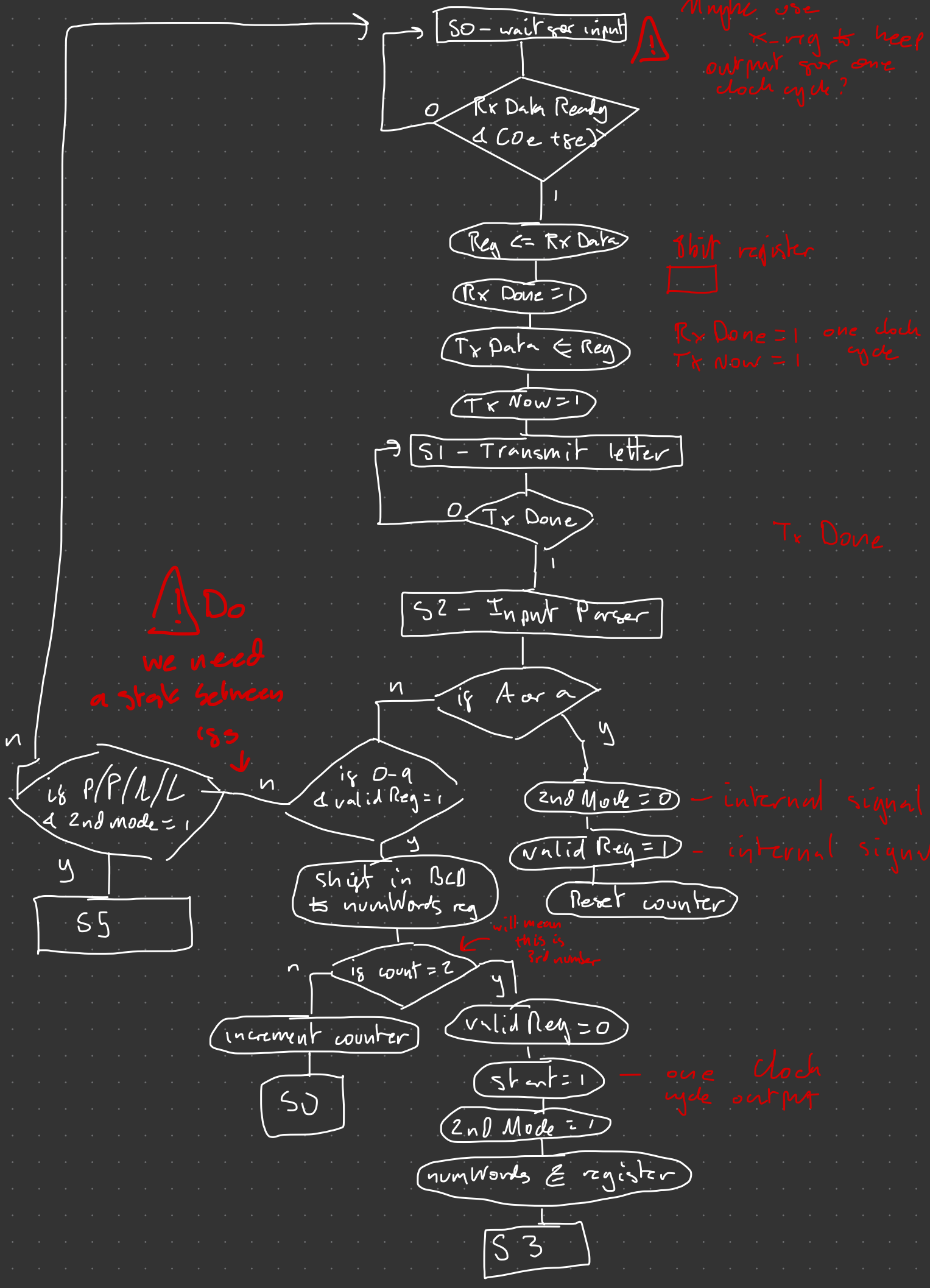
output sig binary signal

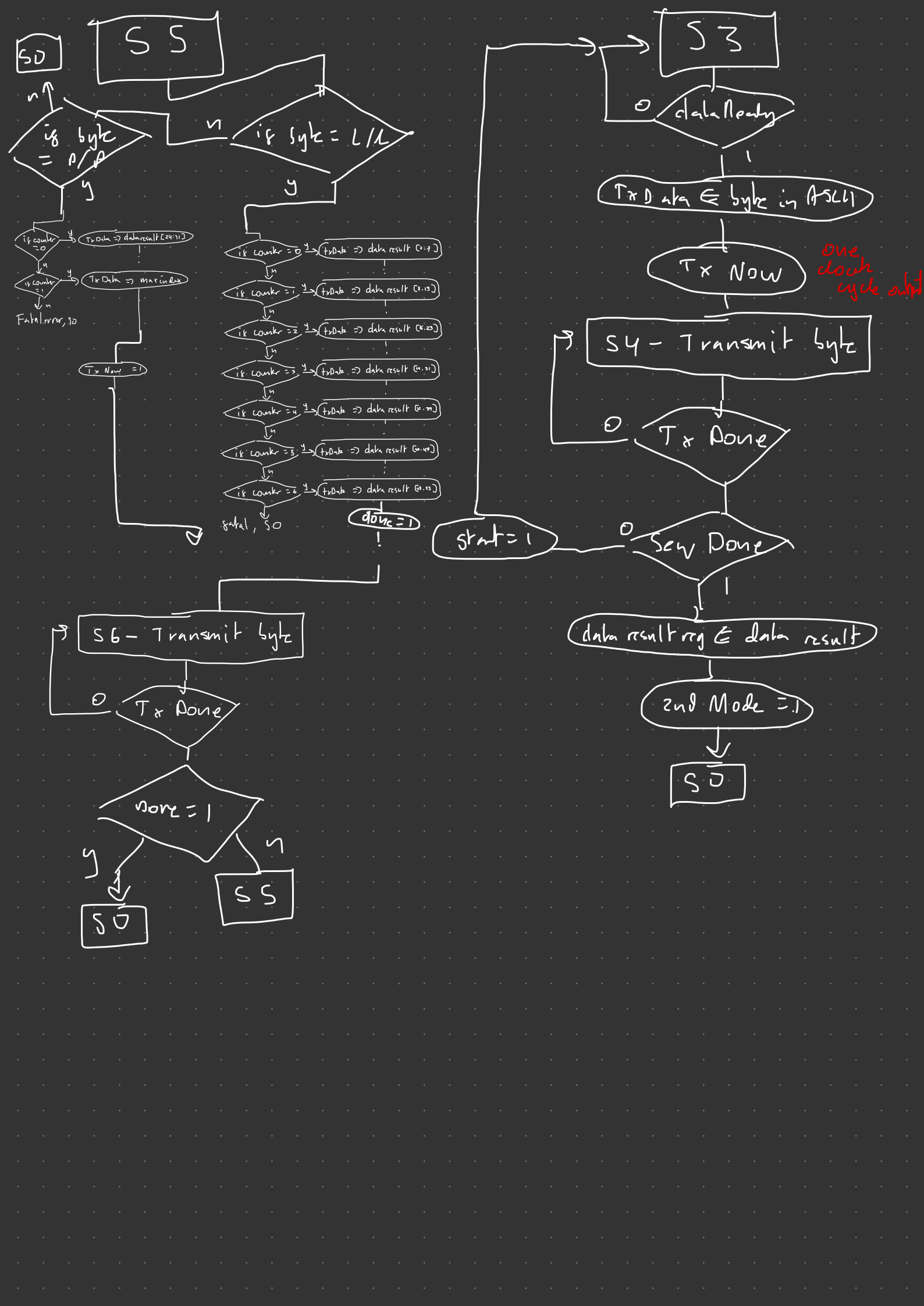
Rx Reg 8-bit binary


S0
wait for input
Rx Now / Tx Now
S1
echo input
Tx Done
S2
input parser
2nd Mode = 1
L/L/P/P
Valid Reg & Reg Count = 3 / 2nd mode = 0
+ clear Data Reg
S3
Data operation
Seq Done / 2nd mode = 1
S4
Transmit byte
Data Ready / Tx Now
Tx Done /
S5
Final output
/ Tx Now
S6
Tx Done /



S0 - Await input
(Rx Now)
Rx Data Ready & (oe + xe)'
0
test for recieved letter
Rx Reg <= Rx Data
save input
Rx Done = 1
Tx Data <= Rx Reg
Tx Now = 1
S1 - Transmit letter
0
Tx Done
1
S2 - Input Parser
n
if L/l/p/p
n
if A, a, 0-9
y
y
output Sig = 1
output Sig = 0


S0 - wait for input
Maybe use x_reg to keep output for one clock cycle?
Rx Data Ready & (Oe + Se)
0
1
Reg <= Rx Data
8bit register
Rx Done = 1
Tx Data <= Reg
Rx Done = 1
Tx Now = 1
one clock cycle
Tx Now = 1
S1 - Transmit letter
Tx Done
0
1
Tx Done
S2 - Input Parser
Do we need a state between 1&3
if A or a
n
y
if P/P/L/L & 2nd mode = 1
n
y
S5
if 0-9 & valid Reg = 1
n
y
2nd Mode = 0
— internal signal
valid Reg = 1
— internal signal
Reset counter
shift in BCD to numWords reg
will mean this is 3rd number
if count = 2
n
y
increment counter
S0
valid Reg = 0
start = 1
— one clock cycle output
2nd Mode = 1
numWords <= register
S3


S0
S5
if byte = A/a
y
n
if counter = 0
y
Tx Data => data result [24:31]
n
if counter = 1
y
Tx Data => max index
n
Fatal error, S0
Tx Now = 1
if byte = L/l
y
n
if counter = 0
y
txData => data result [0:7]
n
if counter = 1
y
txData => data result [8:15]
n
if counter = 2
y
txData => data result [16:23]
n
if counter = 3
y
txData => data result [24:31]
n
if counter = 4
y
txData => data result [32:39]
n
if counter = 5
y
txData => data result [40:47]
n
if counter = 6
y
txData => data result [48:55]
fatal, S0
done = 1
S3
0
dataReady
1
TxData ∈ byte in ASCII
Tx Now
one clock cycle output
S4 - Transmit byte
0
Tx Done
1
Serv Done
0
start = 1
data result reg ∈ data result
2nd Mode = 1
S0
S6 - Transmit byte
0
Tx Done
1
more = 1
y
S0
n
S5