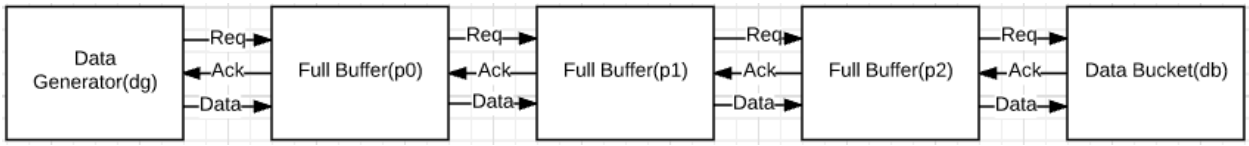
1. Linear Pipeline
    a. Block Diagram




    b.

#Execution cycle= 1000, Cycle Time= 10,

#Average CycleTime=9.996000, Average Throughput=0.100040

    c. Start time of each always block

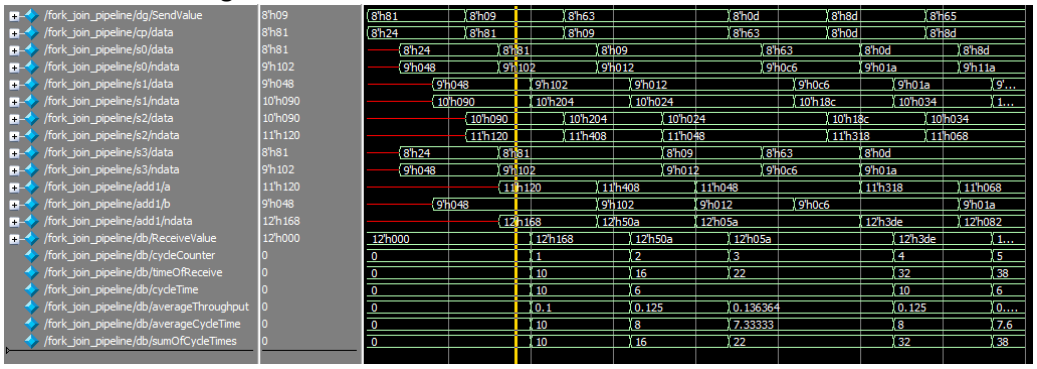
| Blocks | Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4 | Iteration 5 |
|---|---|---|---|---|---|
| dg | 0 ns | 8ns | 18ns | 28ns | 38ns |
| P0 | 0 ns | 8ns | 18ns | 28ns | 38ns |
| P1 | 0 ns | 12ns | 22ns | 32ns | 42ns |
| P2 | 0 ns | 12ns | 22ns | 32ns | 42ns |
| db | 6 ns | 16ns | 26ns | 36ns | 46ns |

    d.

| Simulation Time | Stalled Action | Waiting for | Stall Time |
|---|---|---|---|
| 2ns | P1.receive | P0.send | 2ns |
| 4ns | P2.receive | P1.send | 4ns |
| 6ns | db.receive | P2.send | 6ns |
| 8ns | Dg.send | P1.receive | 8ns |
| 12ns | P1.receive | P0.send | 2ns |
| 14ns | P2.receive | P1.send | 2ns |
| 16ns | db.receive | P2.send | 10ns |
| 18ns | Dg.send | P1.receive | 10ns |
| 22ns | P1.receive | P0.send | 2ns |
| 24ns | P2.receive | P1.send | 2ns |
| 26ns | db.receive | P2.send | 10ns |

    e. The cycle time of an asynchronous linear pipeline is equal to the <u>maximum</u> cycle time of all stages. (Pick from average, maximum, minimum).

2. Re-convergent Fan-out



a. True/False: The circuit can multiply a number by 10 at best every 6ns. Explain your choice.
If the multiplier is at the idle state, the time between first data send out and result received is 10ns. And follows by two 6ns delay of out but. The pattern is 10 6 6. The first number is always 10ns. And the best average cycle time is (10+6+6)/3=7.33. the first cycle time cannot be equal to 6ns. Therefore it cannot at best every 6ns.

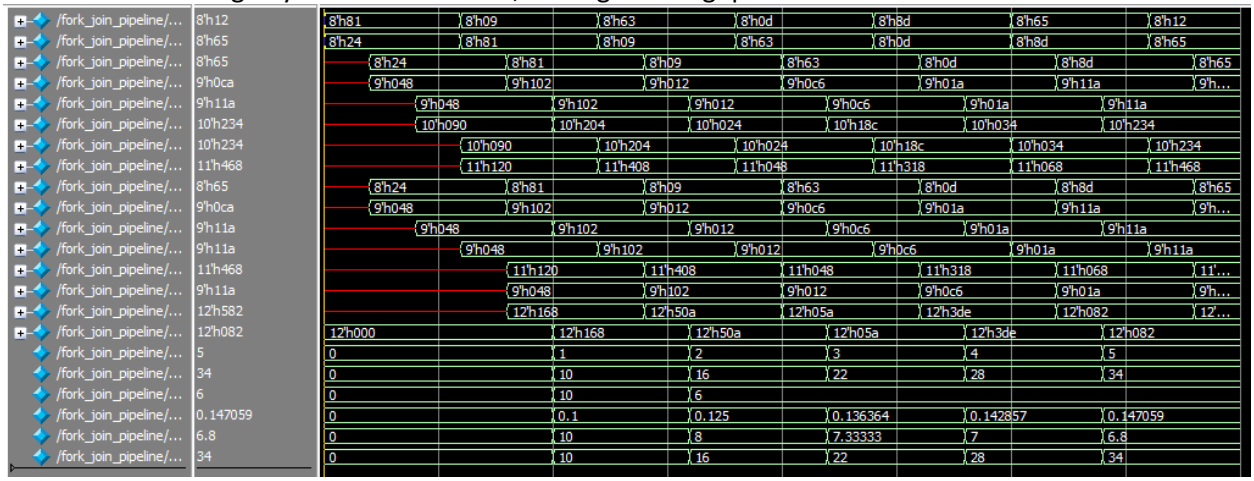
b. Average cycle time and throughput
# Execution cycle= 1000, Cycle Time= 10,
# Average CycleTime=7.336000, Average Throughput=0.136314

c. stalls

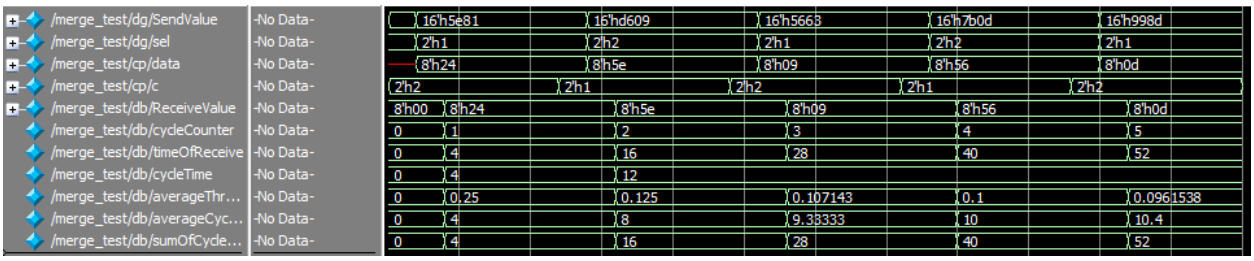
| Simulation Time | Stalled Action | Waiting for | Stall Time |
|---|---|---|---|
| 2ns | S0.receive | Cp.send | 2ns |
| 2ns | S3.receive | Cp.send | 2ns |
| 4ns | S1.receive | S0.send | 4ns |
| 4ns | Add1.receive | S3.send | 4ns |
| 6ns | Dg.send | Cp.receive | 6ns |
| 6ns | S2.receive | S1.send | 6ns |
| 8ns | Add1.receive | S2.send | 8ns |
| 10ns | Db.receive | Add.send | 10ns |
| 12ns | Dg.send | Cp.receive | 6ns |
| 14ns | S3.send | Add1.receive | 4ns |
| 16ns | Db.receive | Add1.send | 6ns |
| 18ns | Cp.send | S3.receive | 4ns |
| 22ns | Dg.send | Cp.receive | 10ns |
| 22ns | Db.receive | Add1.send | 6ns |

d. Look at the iteration start time of the shifter at the lower branch. Does the always block start iteration every 6ns? If not explain why.
The lower branch always block does not iteration every 6ns. Since it will need to wait for the upper branch stall. The copy module will wait until upper and lower both can receive data.

e. Verified can be improved.
# Execution cycle= 1000, Cycle Time= 6,
#Average CycleTime=6.004000, Average Throughput=0.166556

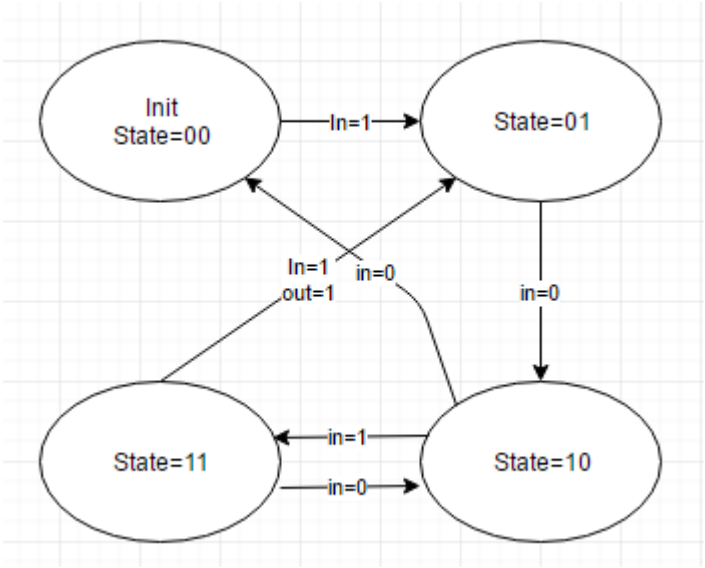


3. Merge



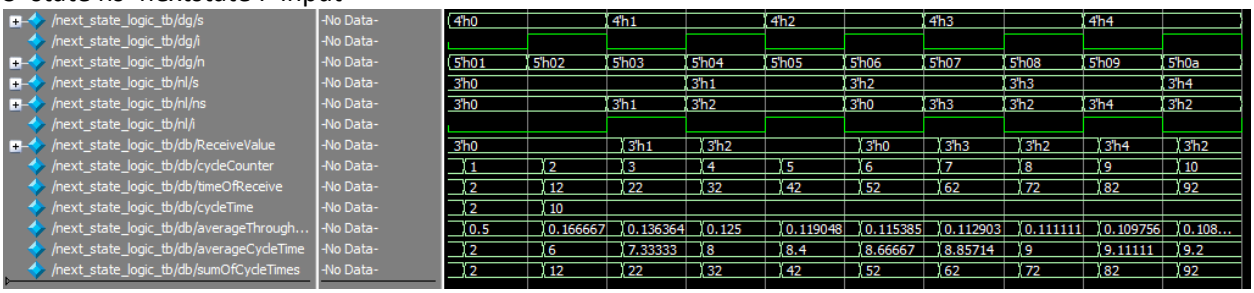

4. Finite State Machine

a. State machine



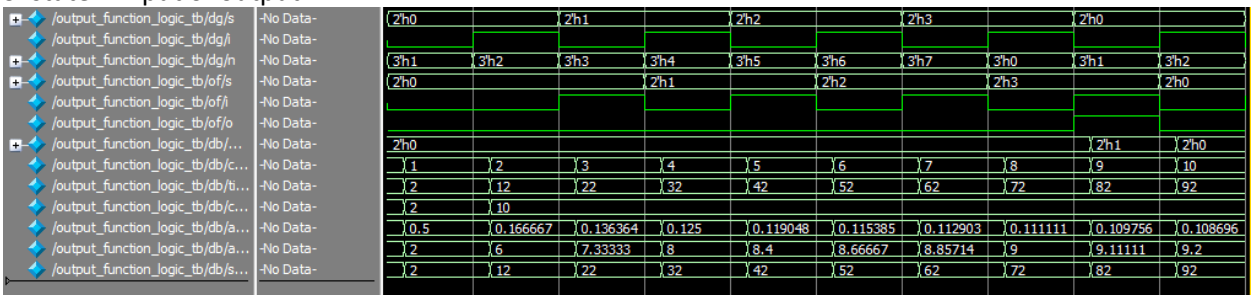

b. Next state logic

S=state ns=nextstate I=input

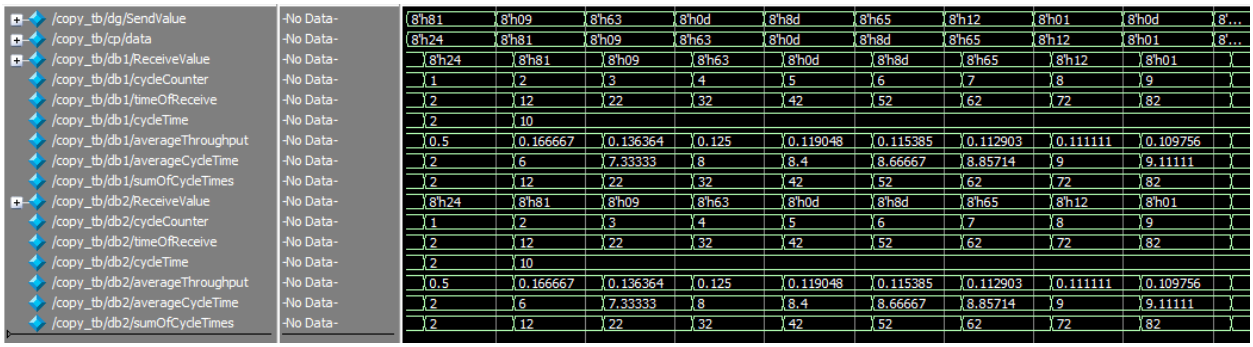



c. Output Function Logic

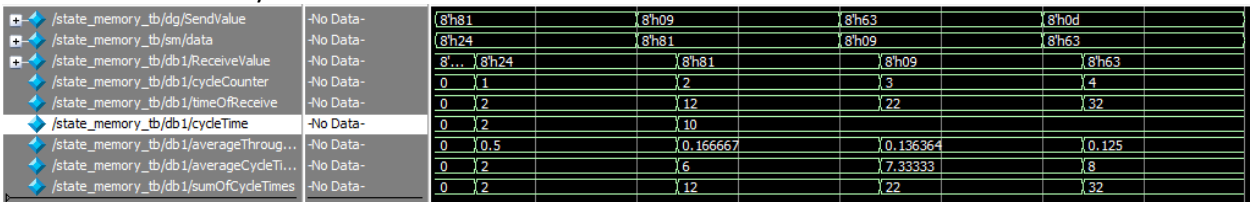
S=state I=input o=output




d. Copy

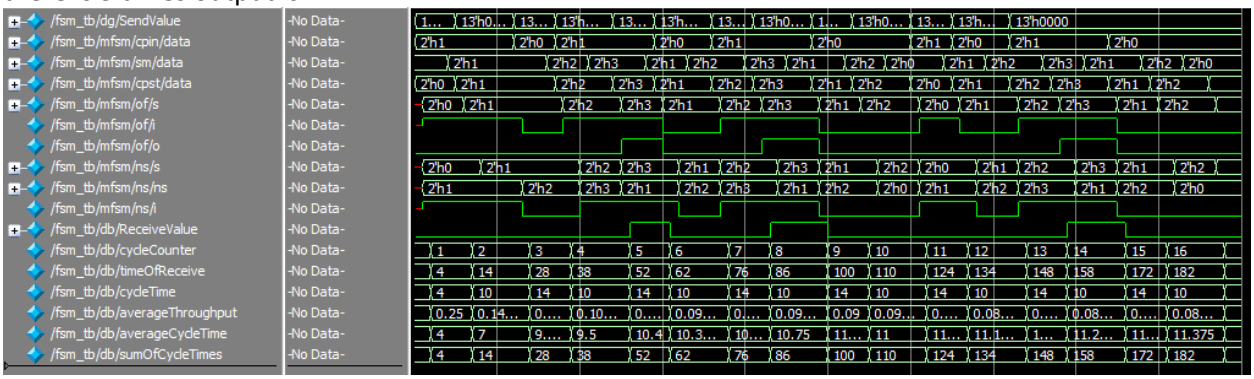



e. State memory




f. FSM

The input sequence is 1011011001011. And there are three 1011 sequences. And the test bench shows there is 3 times output is 1.




5. Palindrome Detector

The P1 module is the first pcell. P initially to be 1. And when S=1, first data feed in the P is still 1. Then when input is 4aa4 P is 1. Then b inserted, the P is0. In the second sequence when the first item inserted the P is 1 and when the last item is inserted the P is 1. For the third sequence, the P is 1 at all times.