**08/01/2021**

**2.**

**1.**

m1, m3, m2, m5

m1, m3, m2, m5

m4, m2

**2.**

Fifo: m1 -> m3 (optional for p3), m4 -> m5

Local: m3 -> m2

**Causal:**

M1, m3, m2, m4, m5

M1, m3, m2, m4, m5 (p3 m3->m1)

M1, m3, m4, m5, m2

M1,m4, m5, m3, m2

m4, m5, m1, m3, m2

M1, m3, m4, m2, m5

Ecc.

**Total:**

Same, but also p3 has to respect both FIFO constraints

**3.**

**3.**

**1.**

Read2() -> 0,5,7

Read4() -> 5,7

Read1() -> 5,7

Read3() -> 5,7

Read5() -> 5,7

**2.**

Read2() -> 0,5,7

Read4() -> 5,7

Read1() -> 5,7

Read3() -> 5,7 if 1|2|4->5, 7 if 1|5|7->7

Read5() -> 5,7 if 3->5, 7 if 3->7

**3.**

Read2() -> 0

Read4() -> 5

Read1() -> 5

Read3() -> 7

Read5() -> 7

**5. Implement RB with FIFO p2p, perfect oracle with new\_neighbour(p,side)**

**Upon event <rb,Init> do**

Correct = PI

From[p] = [NULL]N

**Upon event <rb,Broadcast | m> do**

Trigger <FIFOp2p, Send, m, self> to self.south;

Trigger <FIFOp2p, Send, m, self> to self.east;

Trigger <FIFOp2p, Send, m, self> to self.west;

Trigger <FIFOp2p, Send, m, self> to self.north;

**Upon event <FIFOp2p, Deliver, p | m,s> do**

If m not in from[s] do

From[s] = from[s] u m

Trigger <rb, Deliver | m>;

Trigger <rb, Broadcast | m>;

**Upon event <Oracle, Crash | p> do**

Correct = correct \ {p}

New\_neighbour(p, east) to p.west

New\_neighbour(p, west) to p.east

New\_neighbour(p, north) to p.south

New\_neighbour(p, south) to p.north

Forall m in from[p] do

Trigger <rb, Broadcast | m>;