Fair Switch Arbiter

Implement the FSA proposed by:

A Low-Latency Fair-Arbiter Architecture for Network-on-Chip Switches,
Luo, J.; Wu, W.; Xing, Q.; Xue, M.; Yu, F.; Ma, Z, 2022

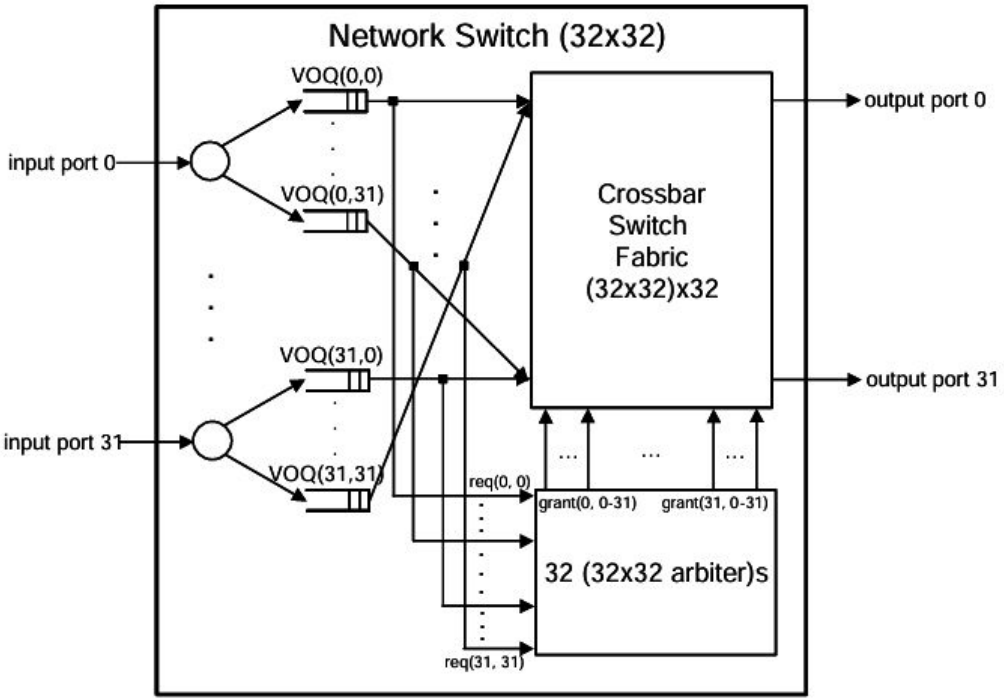
FSA is a step from **Switch Arbiter(SA)** proposed by:

Round-robin Arbiter Design and Generation
E, Shin, V, Mooney III and G, Riley, 2010

Background

VOQ: N*N
(N input x N output)

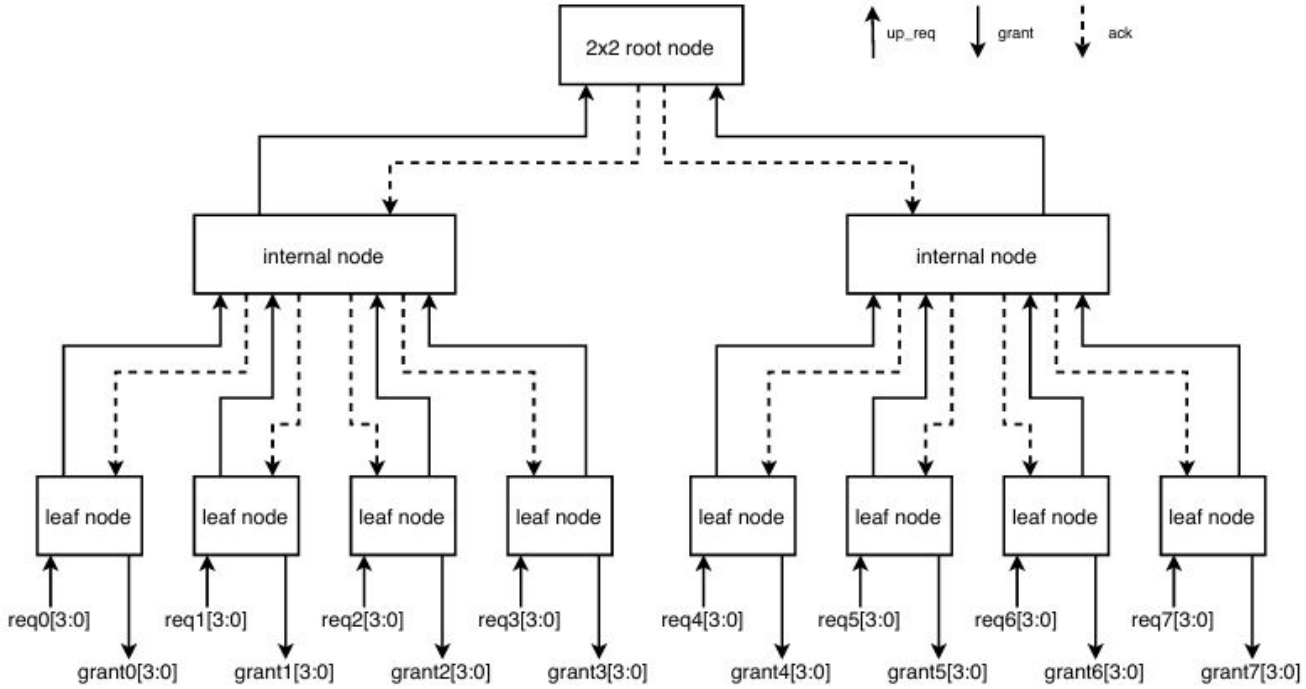
N arbiter (N output ports)



Switch Arbiter

Tree based structure

Reduces to O(log4N)



SA - Unfair

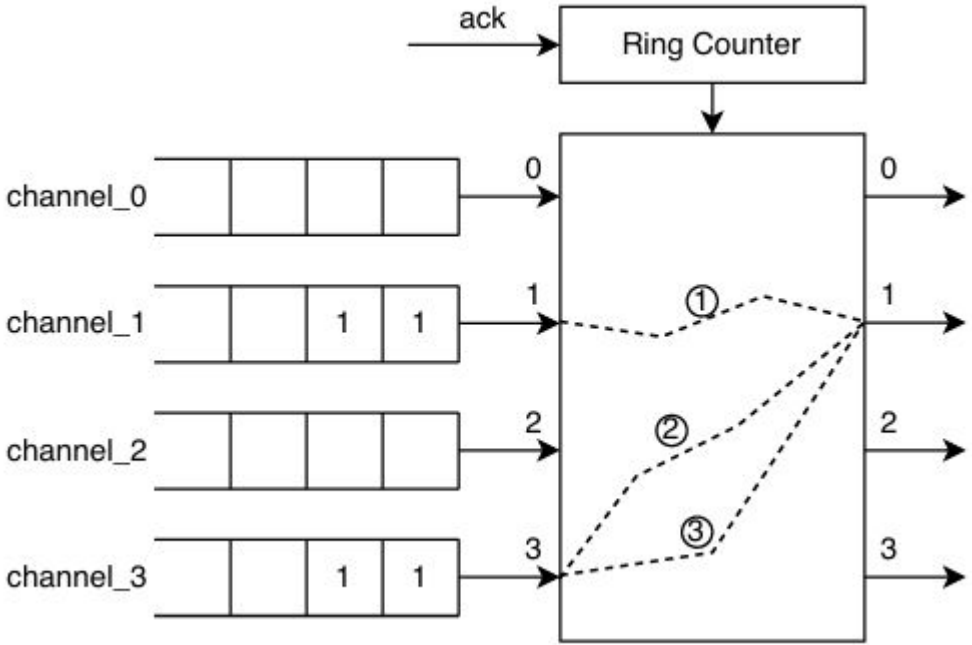
1. Assume all request access to output port 1. 1>2>3>0

2. Round 1: input 1 wins: 2,3,0,1

3. Round 2: 2 absent -> input 3 wins

3,0,2,1

4. Round 3: input 3 wins again



FSA - lock

$$lock = g_0 + g_1\overline{r_0} + g_2\overline{r_1}\cdot\overline{r_0} + g_3\overline{r_2}\cdot\overline{r_1}\cdot\overline{r_0}$$

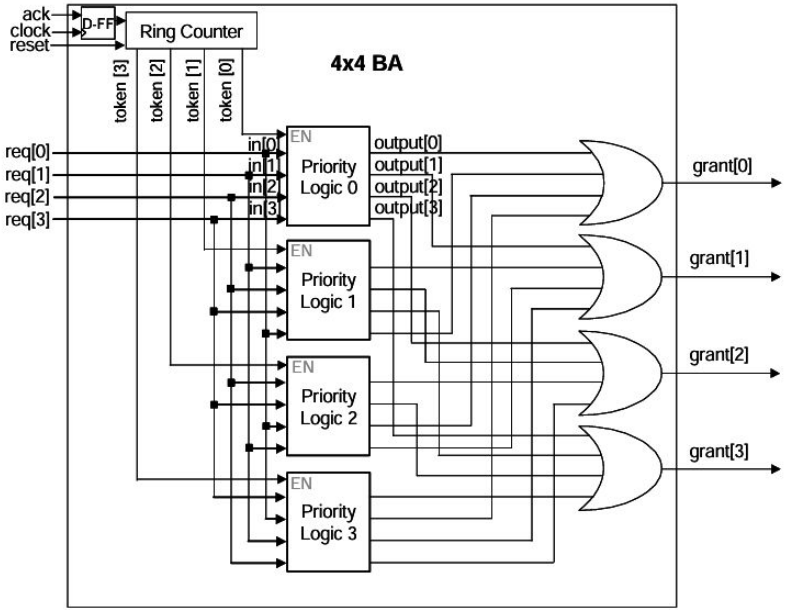
$$up\_req = \overline{\overline{lock}}' \cdot any\_req$$

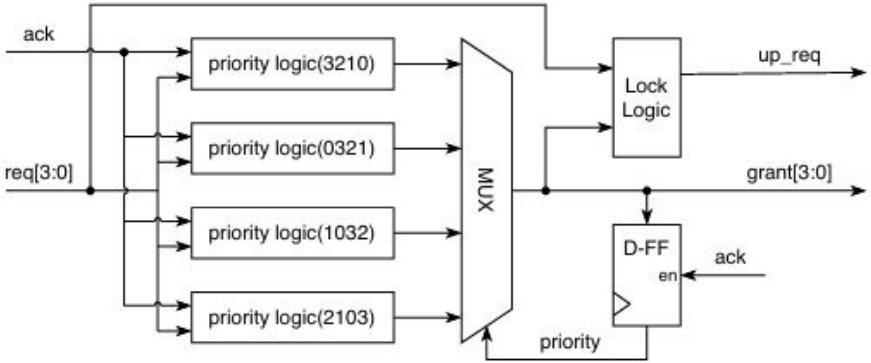
all lower priority have been served or absent

Every input of leaf nodes should be serviced once before the priority vector of the higher-level node is updated.

Fairness Under non-uniform inputs

Leaf node


ack
clock
reset
D-FF
Ring Counter
4x4 BA
token [3]
token [2]
token [1]
token [0]
req[0]
req[1]
req[2]
req[3]
in[0]
in[1]
in[2]
in[3]
EN
Priority Logic 0
Priority Logic 1
Priority Logic 2
Priority Logic 3
output[0]
output[1]
output[2]
output[3]
grant[0]
grant[1]
grant[2]
grant[3]


ack
req[3:0]
priority logic(3210)
priority logic(0321)
priority logic(1032)
priority logic(2103)
MUX
Lock Logic
up_req
grant[3:0]
D-FF
en
ack
priority

SA to FSA

1. Add Lock logic at leaf node - achieve absolute fairness

2. achieve lower latency and less area in the upper nodes

Putting all together forms an optimized Round-robin like arbiter (iSLIP)

Design and Analysis of High-Speed Parallel Prefix Adder for Digital Circuit Design Applications, G. Thakur, H. Sohal and S. Jain, 2020