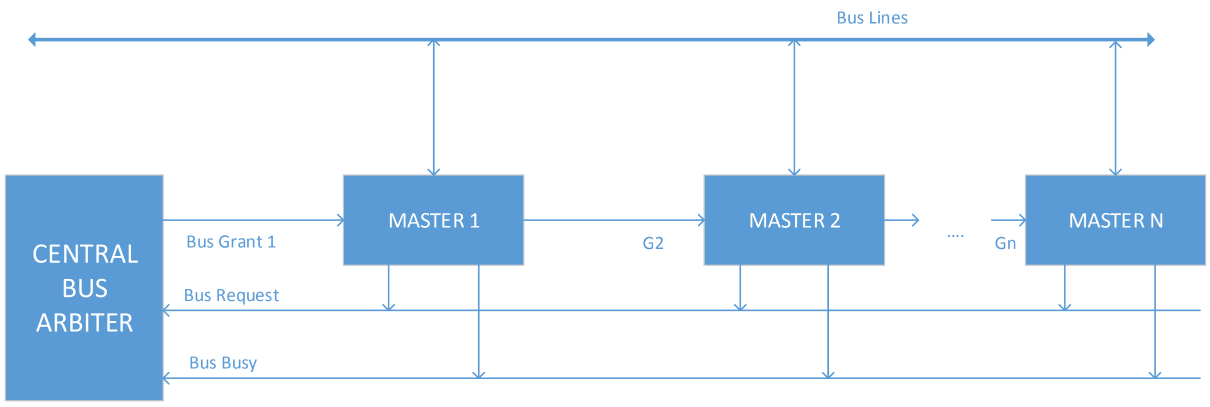
LECTURE 9

<u>Centralized Arbitration</u>

Protocol of allocating the bus:

i. Master$_i$ requests the bus by activating its dedicated bus request line
ii. If the bus busy line is passive, that is, no other master is using the bus, the arbiter immediately allocates the bus to the requester by activating the grant$_i$ line
The requestor deactivates its request line and activates the bus busy line prohibiting the allocation of the bus for other requests. After completing the bus transaction, the requestor deactivates bus busy line
iii. When the bus busy line is active, the arbiter does not accept any requests
iv. When several request lines are active by the time the bus busy line becomes passive, the arbiter can use any of the following bus allocation policies: fixed priority, rotating priority, round robin, least recently used, first come first served

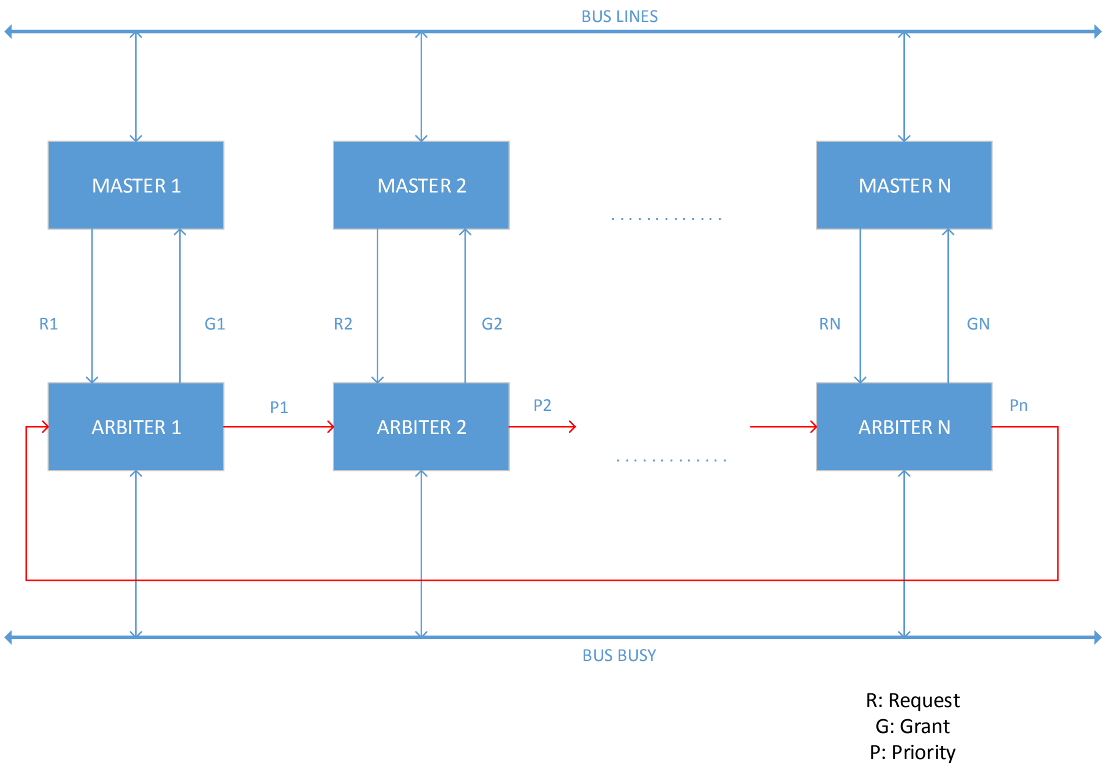
<u>DAISY CHAINED Bus Arbitration</u>



(Figure 1)

- One of the most popular organizations of arbiter logic is daisy-chaining
- There is only one shared bus request line. All the masters use this line to indicate their need to access the shared bus.
- The arbiter passes the bus grant line to the first master and then it is passed from master to master, creating a chain of masters
- The priority of a master is determined by its position in the grant chain
- The closer it is to the arbiter, the higher it's priority
- A master can access the shared bus if the bus busy line is passive and its input grant line is active
- When the master does not register the bus and receives an active grant line, it activates its output grant line, enabling the next master to use the bus.

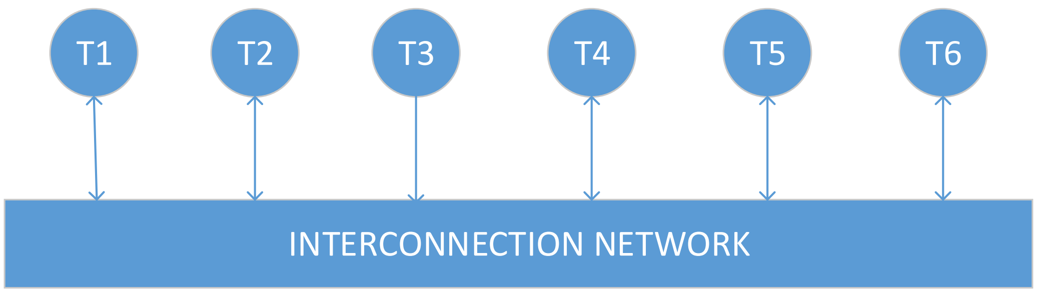
DECENTRALIZED ROTATING Arbiter



(Figure 2)

- Daisy chained scheme lacks fairness
- Rotating arbiter eliminates this drawback
- Arbiter$_i$ is allowed to grant its coupled master unit if master$_i$ has activates its bus request line, the bus busy line is passive, and the priority (i-1) input line is active.
- If master$_i$ has not activates its bus request line, arbiter$_i$ activates its output priority$_i$ line
- Selecting the lowest priority unit:
    - Daisy chained arbiter -> the master farthest away
    - Rotating Arbiter -> the master that releases the bus

INTERCONNECTION NETWORKS



(Figure 3: Functional View of an Interconnection Network)

- An interconnection network is a programmable system that transports data between terminals
- When terminal T3 wishes to communicate some data with terminal T5, it sends a message containing the data into the network and the network delivers the message to T5
- The network is programmable in the sense that it makes different connections at different points in time. The network in the figure may deliver a message from T3 to T5 in one cycle and then use the same resources to deliver a message from T3 to T1 in the next cycle
- The network is a system because it is composed of many components: buffers, channels, switches and controls that work together to deliver data

Topology, Routing and Flow Control:

TOPOLOGY: The interconnection network is implemented with a collection of shared router nodes connected by shared channels. The connection pattern of these nodes defines the network topology. A message is then delivered between terminals by making several hops across the shared channels and nodes from its source terminal to the destination terminal.

ROUTING: Once a topology has been chosen, there can be many possible paths (sequences of nodes and channels) that a message could take through the network to reach its destination.
Routing determines which of these possible paths a message actually takes.

FLOW CONTROL: Flow control dictates which messages get access to particular network resources over time.

CIRCUIT SWITCHING: Circuit switching is a form of bufferless flow control that operates by first allocating channels to form a circuit from source to destination, and then sending one or more packets along this circuit. When no further packets need to be sent, the circuit is deallocated.

NON BLOCKING Networks:

A network is said to be non-blocking if it can handle all circuit requests that are a permutation of the inputs and outputs. That is, a dedicated path can be formed from each input to its selected output without any conflicts (shared channels)

Conversely, a network is blocking if it cannot handle all such circuit requests without conflicts

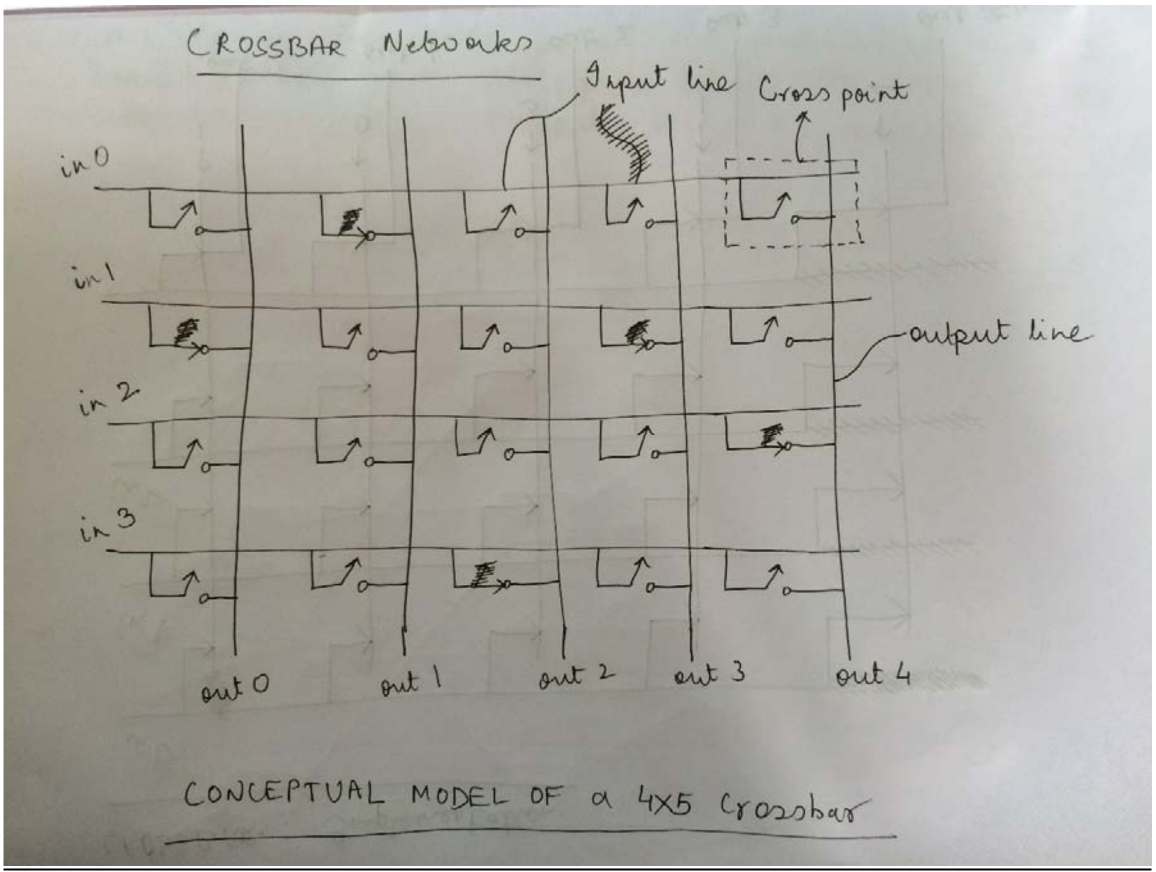
STRICTLY NON-BLOCKING Networks

A network is strictly non-blocking if any permutation can be set up incrementally, one circuit at a time, without the need to reroute (or rearrange) any of the circuits that are already set up

REARRANGEABLY NON-BLOCKING (or REARRANGEABLE) Networks

Such a network can route circuits for arbitrary permutations, but incremental construction of a permutation may require rearranging some of the early circuits to permit later circuits to be set up.

CROSSBAR Networks

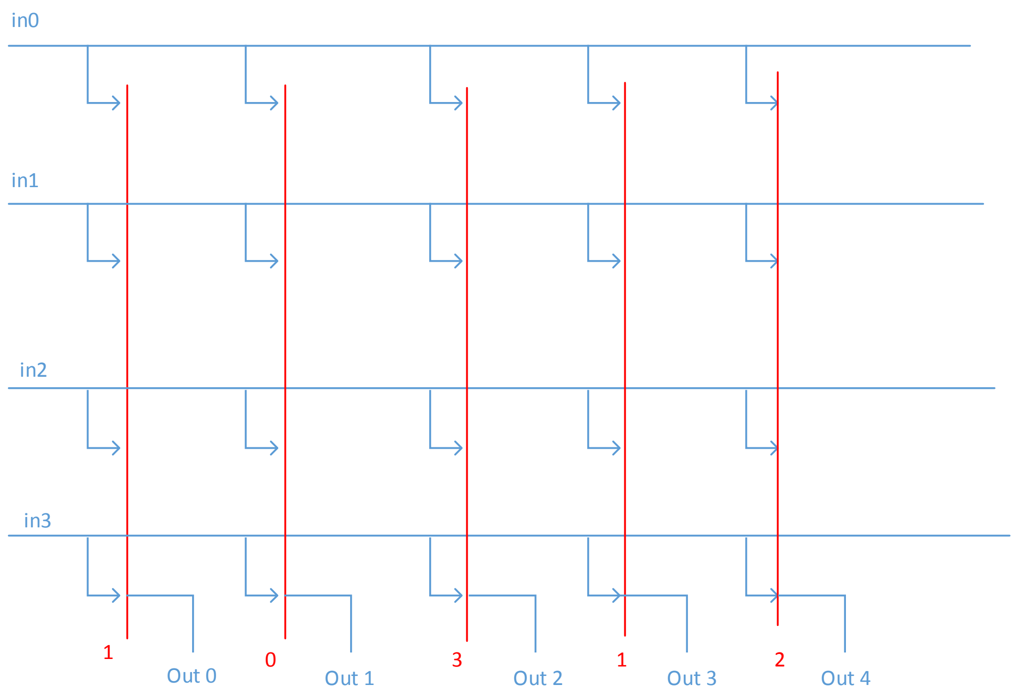


(Fig 4: CONCEPTUAL MODEL OF A 4x5 CROSSBAR)

The crossbar in the fig consists of 4 input lines, 5 output lines, and 20 crosspoints. Each output may be connected to at most one input, while each input may be connected to any number of outputs. This switch has inputs 1, 0, 3, 1, 2 connected to outputs 0,1,2,3,4 respectively.

An n x m crossbar or crosspoint switch directly connects n inputs to m outputs with no intermediate stages. In effect, such a switch consists of m n:1 multiplexers, one for each output.

CROSSBAR: Implementation

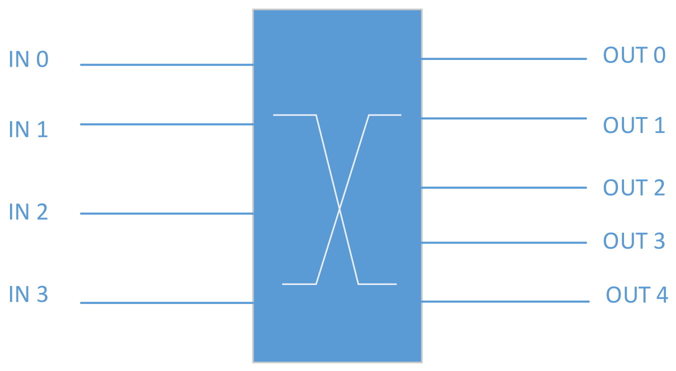


(Figure 5)

A 4x5 crossbar switch as implemented with 5 4:1 multiplexers. Each multiplexers selects the input to be connected to the corresponding output. Here the connection is

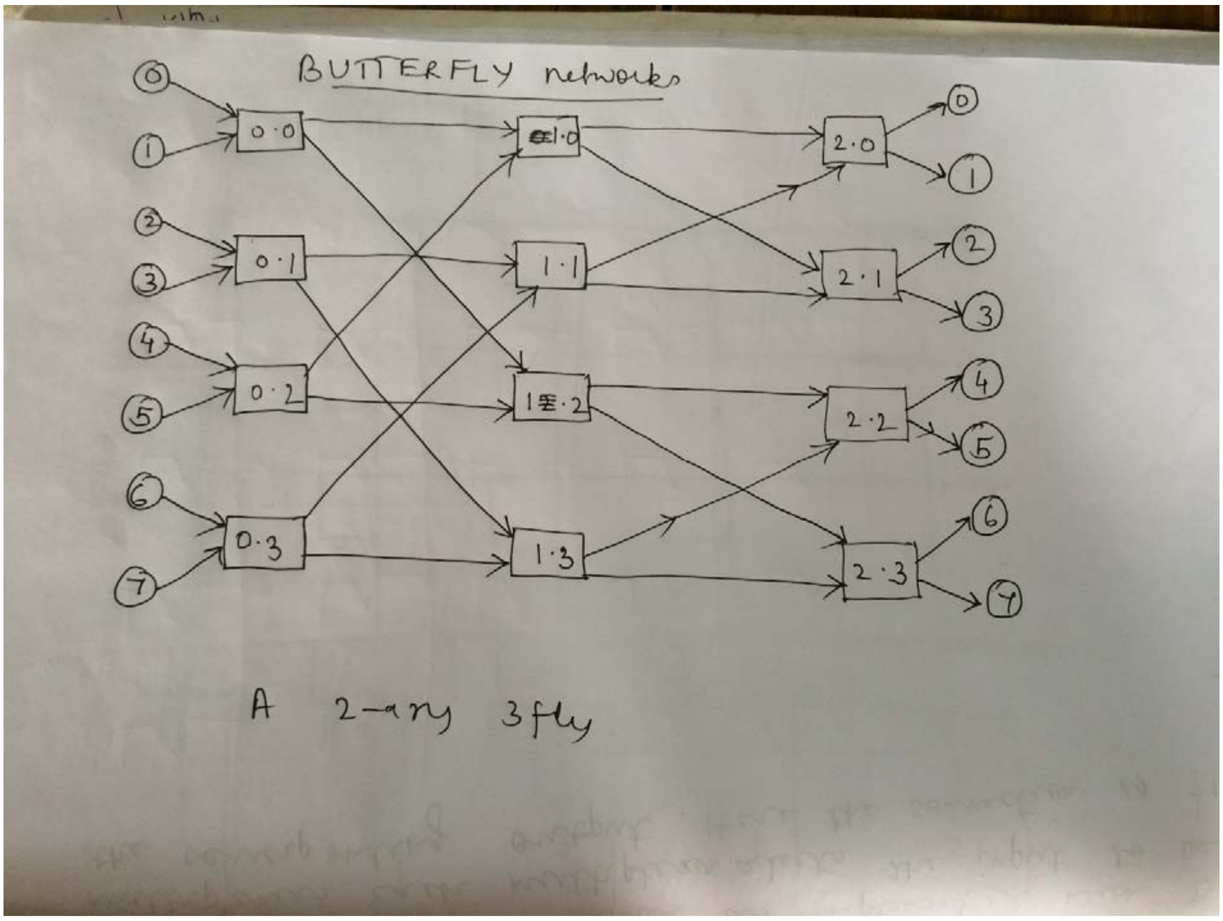
{1,0,3,1,2} -> {0,1,2,3,4}

Each of the n input lines connects to one input of the n n:1 multiplexers. The outputs of the multiplexers drive the m output ports. The multiplexers may be implemented with tri-state gates or wired OR gates driving an output line or with a tree of logic gates to realize a more conventional multiplexer



(Fig 6) (from Rounak)
(Symbol of a 4x5 CROSSPOINT switch)

BUTTERFLY Network



(Figure 7: A 2-ary 3-fly)

A k-ary n-fly network consists of k^n source terminal nodes, n stages of k^{n-1} kxk crossbar switch nodes and finally k^n destination terminal nodes.

Routing:

6→1

0 0 1   0- lower, 1 – upper

2→5

1 0 1