

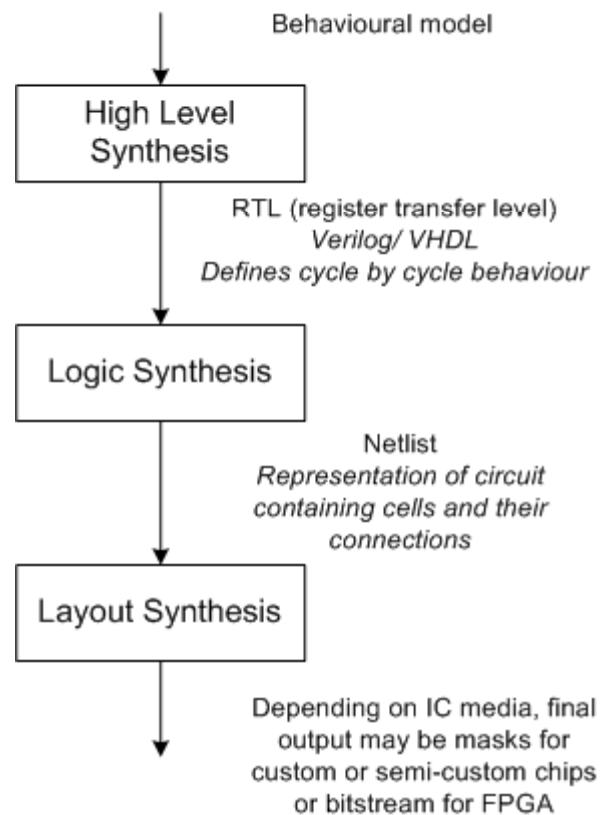
**UNIVERSITY OF  
TORONTO  
ECE-1387 CAD FOR  
DIGITAL CIRCUIT  
SYNTHESIS AND LAYOUT**

Instructor: J. Anderson  
Scribe: K. Ng

Lecture 2 Routing

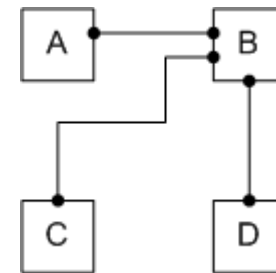
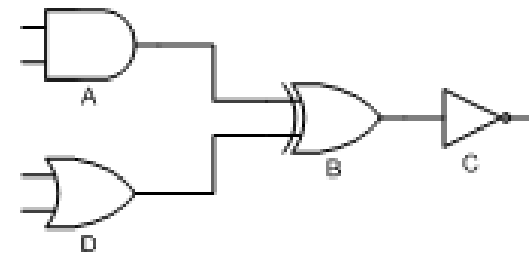
# Synthesis

- Synthesis consists of 3 major steps
- Routing is the last step in layout synthesis



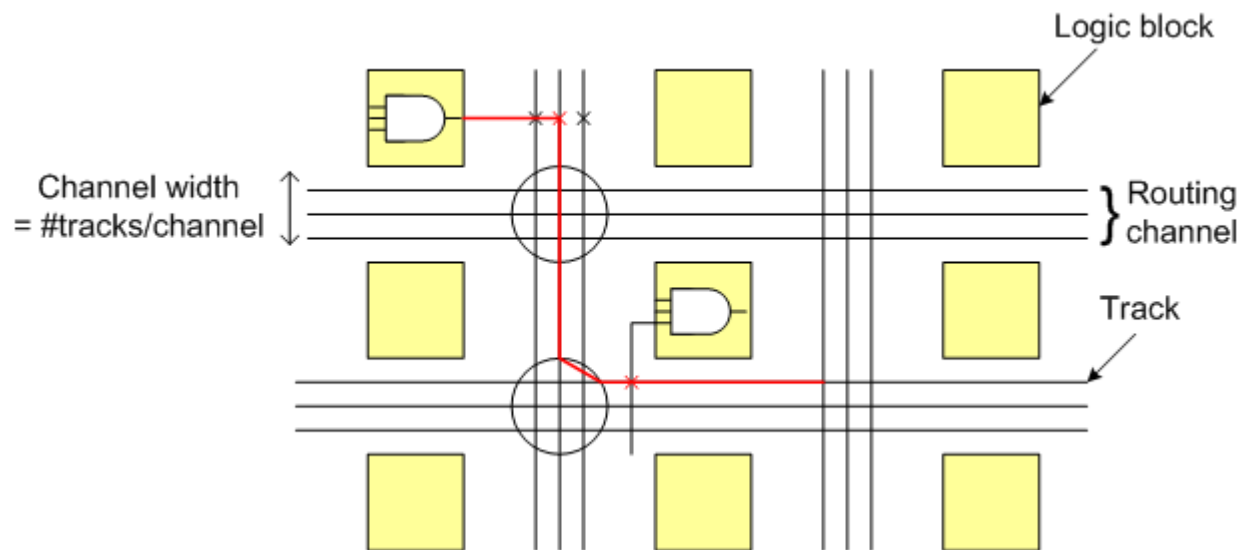
# Generic Problem Statement

- Given
  - ▣ positions of cells, gates/flip-flops and pins
  - ▣ “Netlist” of connections between cells
- Find
  - ▣ paths of the wires so that required connections are made and no unrelated wires are shorted
- Exact problem statement depends on IC media (eg: FPGA) and constraints (ex: 100W power constraint, 50MHz speed constraint)



# FPGA

- FPGA routing
  - ▣ Same basic routing algorithm as for custom chips
- FPGA is
  - ▣ an array of programmable logic blocks surrounded by programmable routing



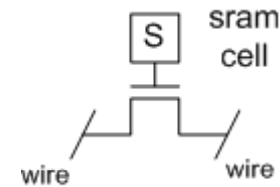
W: # of tracks per channel

# Programmable Routing

- Contains wires and programmable switches
- Programmable switch in early FPGAs

- Pass transistor switches
- SRAM cell set to 0  $\Rightarrow$  off, no connection
- SRAM cell set to 1  $\Rightarrow$  on, wires connected
- Bi-directional switch
- Disadvantages

- nmos is good for passing 0 and bad for passing 1, i.e.: bad rise time
- Additional loads adds capacitance (delays through the routing fabric are highly fanout dependent).



# Programmable Routing

- Modern FPGAs have buffered switches

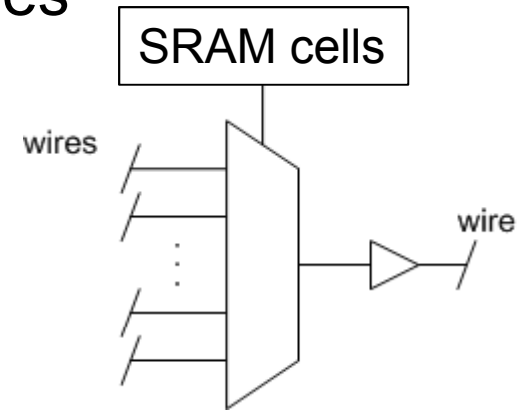
- Advantages

- Better speed through the buffer (repeated)
    - Fanout independence

- Unidirectional connection

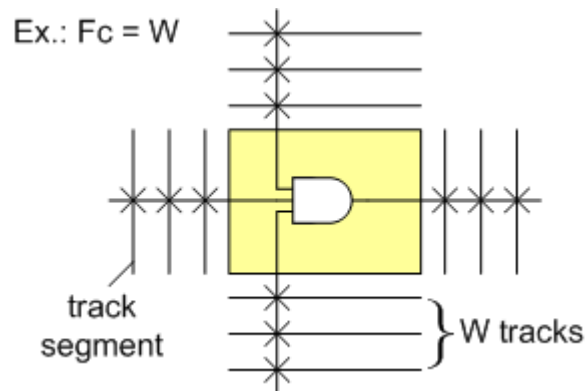
- **Main idea:**

- A route is made by turning on the right programmable switches to connect gates as desired
  - **Router chooses the switches to turn on**



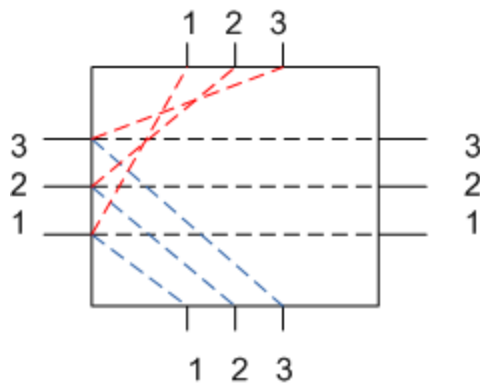
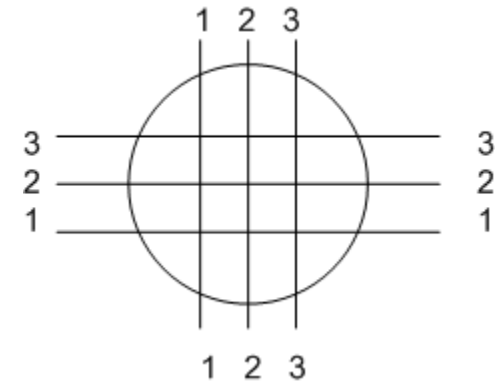
# 2 Basic Routing Structures

- 1.) Connection block/box: connection between gates' pins and tracks in neighboring channel
  - ▣ Can choose to turn on anywhere with an X
  - ▣ #X's per pin:  $F_c$  (flexibility of connection box)
  - ▣ Track segment: segment of a track spans one logic block



# 2 Basic Routing Structures

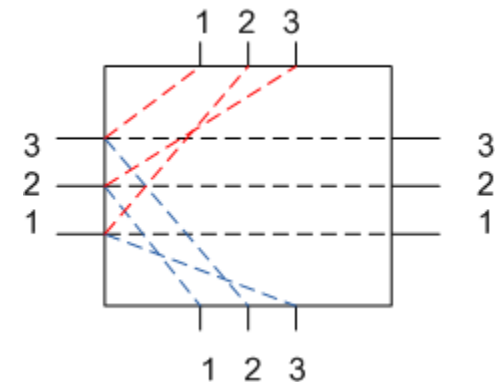
- 2.) Switch block/box: intersection between horizontal and vertical channels
  - ▣ No need to put switches between all wires (waste of silicon)
  - ▣ Planar switch box: track  $i$  on one side can connect through programmable switch to track  $i$  on the other 3 sides



# 2 Basic Routing Structures

## □ Wilton switch box

- Give better routing results: fewer routing resources are needed to route circuits.
- Switch model:
  - For connections that are “straight through”: stay on the same track.
  - Turning connections are rotated by 1 track, for example:
    - West  $\leftrightarrow$  North:  $i \leftrightarrow i+1$ 
      - 1  $\leftrightarrow$  2
      - 2  $\leftrightarrow$  3
      - 3  $\leftrightarrow$  1
    - West  $\leftrightarrow$  South:  $i \leftrightarrow i-1$ 
      - 1  $\leftrightarrow$  3
      - 2  $\leftrightarrow$  1
      - 3  $\leftrightarrow$  2



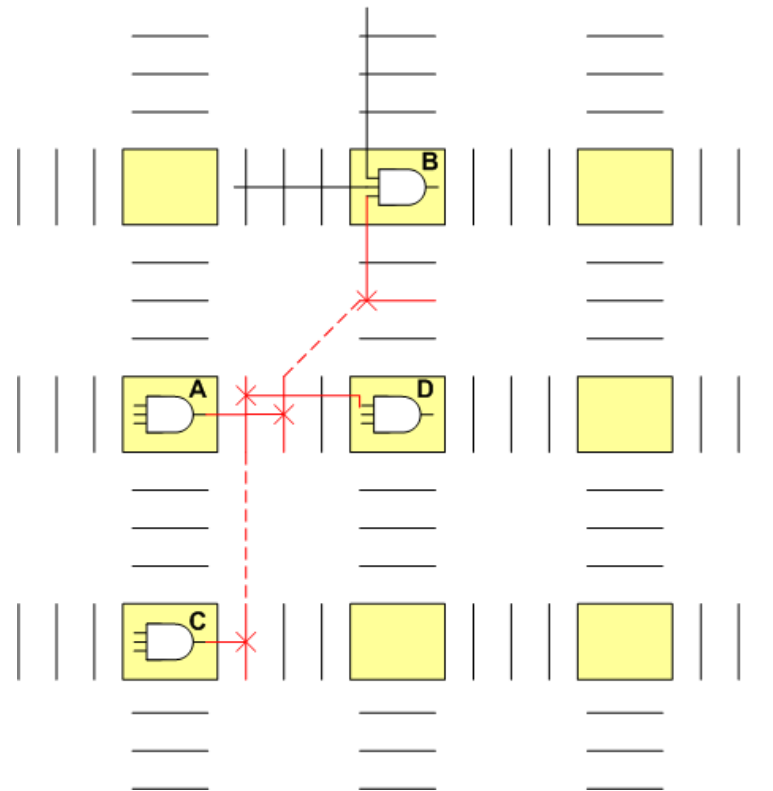
# 2 Basic Routing Structures

- So for each connection between pins, the router must select which programmable switches to turn on
  - Can't use wires previously used by other connections!

Router has to be aware of used wires (blockages)

Say, routing  $A \Rightarrow B$  first,  $C \Rightarrow D$  next, watch out for blockage when routing C to D, i.e. wires that were already used for  $A \Rightarrow B$

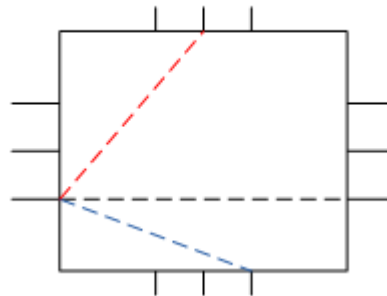
Incremental placement may be required if routing is not possible for current placement



# Wilton Switch Box

- West – North:  $i$  on West  $\leftrightarrow$   $i+1$  on North
  - West – South:  $i$  on West  $\leftrightarrow$   $i-1$  on South
  - East – North:  $i$  on East  $\leftrightarrow$   $i-1$  on North
  - East – South:  $i$  on East  $\leftrightarrow$   $i+1$  on South
- 
- $F_s$  = switch box flexibility, # of wires each wire can connect to

Ex.:  $F_s = 3$



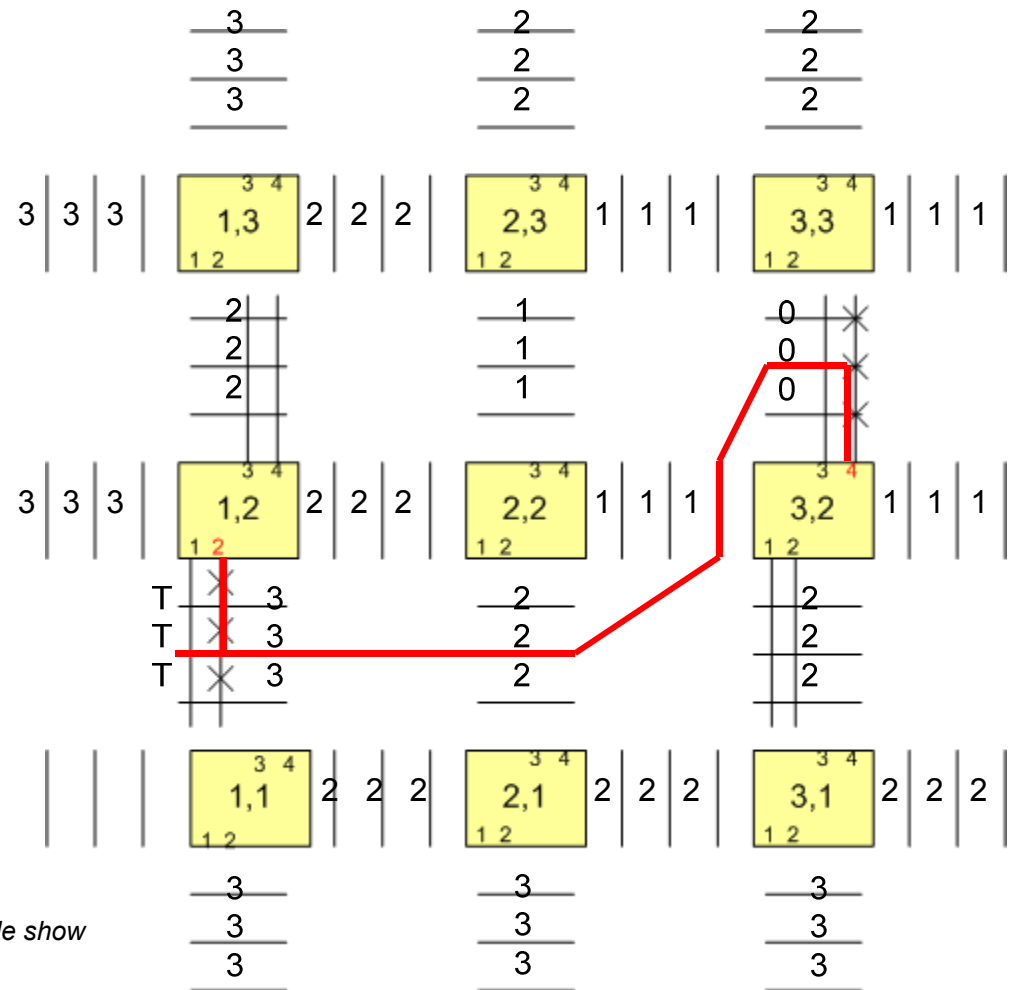
# Lee-Moore Maze Routing Algorithm

- Used for FPGA and ASICs
- Similar to Dijkstra's shortest path algorithm
- Given
  - ▣ Pins to connect
  - ▣ FPGA architecture ( $W$ ,  $F_s$ ,  $F_c$ )
- There are many connections (source/load pin pairs) to route, route one at a time
  - ▣ Create a data structure that has one entry for each track segment
  - ▣ Label all segments available 'A'
  - ▣ Route one connection at a time
  - ▣ Each routed connection then uses segments that are made unavailable 'U' to subsequent connection (i.e. they are blockages)

# Example

- Route from source/driver pin 4 at (3,2) to sink/load pin 2 at (1,2)

1. Label neighboring tracks starting from the source (0)
2. Label the neighboring tracks to the target as T
3. Expand the labeling until target tracks (T) are reached
4. Backtrack from target to source through descending labels



\*Click the mouse to show routing animation during slide show

# Lee-Moore Maze Routing Algorithm

- For all connections to route, we want to find path between 2 pins  $(x_1, y_1)p_1 \Leftrightarrow (x_2, y_2)p_2$ 
  1. First pin is source (S), 2<sup>nd</sup> pin is target (T)
  2. Mark all track segments that are available and connect to target pin labeled as “T”
  3. Mark all track segments connected to source that are available with a mark ‘0’, put these segments on an expansion list  
(expansion list is a FIFO queue)

# Lee-Moore Maze Routing Algorithm

```
4. While expansion list is not emptied {
    for each segment j on list with smallest label L {
        for each segment k connected to j through programmable switch {
            if k == target
                exit while loop
            if k is unavailable or already labeled
                ignore
            else k should be available
                label k with L+1
                put k onto expansion list (end of FIFO queue)
        }
    }
    remove j from expansion list
}
```

# Lee-Moore Maze Routing Algorithm

- If the while loop ends without hitting target
    - No path found to target pin
    - Declare failure for connection
  - Is it possible to miss a path that was available?
    - No, if the path exists and is available, the maze router will find it
  - If a connection fails, one heuristic approach is to remove the routing for all connections, and iterate another time by routing the failing connection first
5. If the target is hit, then trace back along switches/segment through descending #'s to source (thereby defining a routing path from the source to the target pin).
- Mark path as 'U' (unavailable)
  - Note choices when there are 2 equal segments (ex.: fewer turns or use congestion metrics [e.g. choose less congested channel])

# Assignment 1



- Implement the maze router described above:
  - Use Wilton switch box
  - $W$  and the array size of logic blocks, are parameters to your program
  - Find minimum  $W$  to route each circuit
  - Given 4 test circuits
  - Display routing with graphics, show FPGA, routing and tracks
  - Explore approaches to route multi-fanout nets using a minimal amount of routing wires.