

# SEE Tolerant Standard Cell Based Design While Guaranteeing Specific Distance Between Memory Elements

*Sandeep Miryala*  
ASIC Group, Fermilab

# Outline

- **Introduction**
  - Single Events Effects (SEE)
  - Mitigation Techniques : Triple Modular Redundancy (TMR)
- **TMR automation in semi-custom flow**
  - Digital Logic Synthesis Flow (RC Compiler / Genus)
    - TMR insertion
  - Physical Design (Innovus)
    - Specific distance between memory elements
    - Clock delay insertion
- **Application**
  - RD53A Prototype Chip
- **Conclusion**

# Radiation Effects on Semiconductor Devices

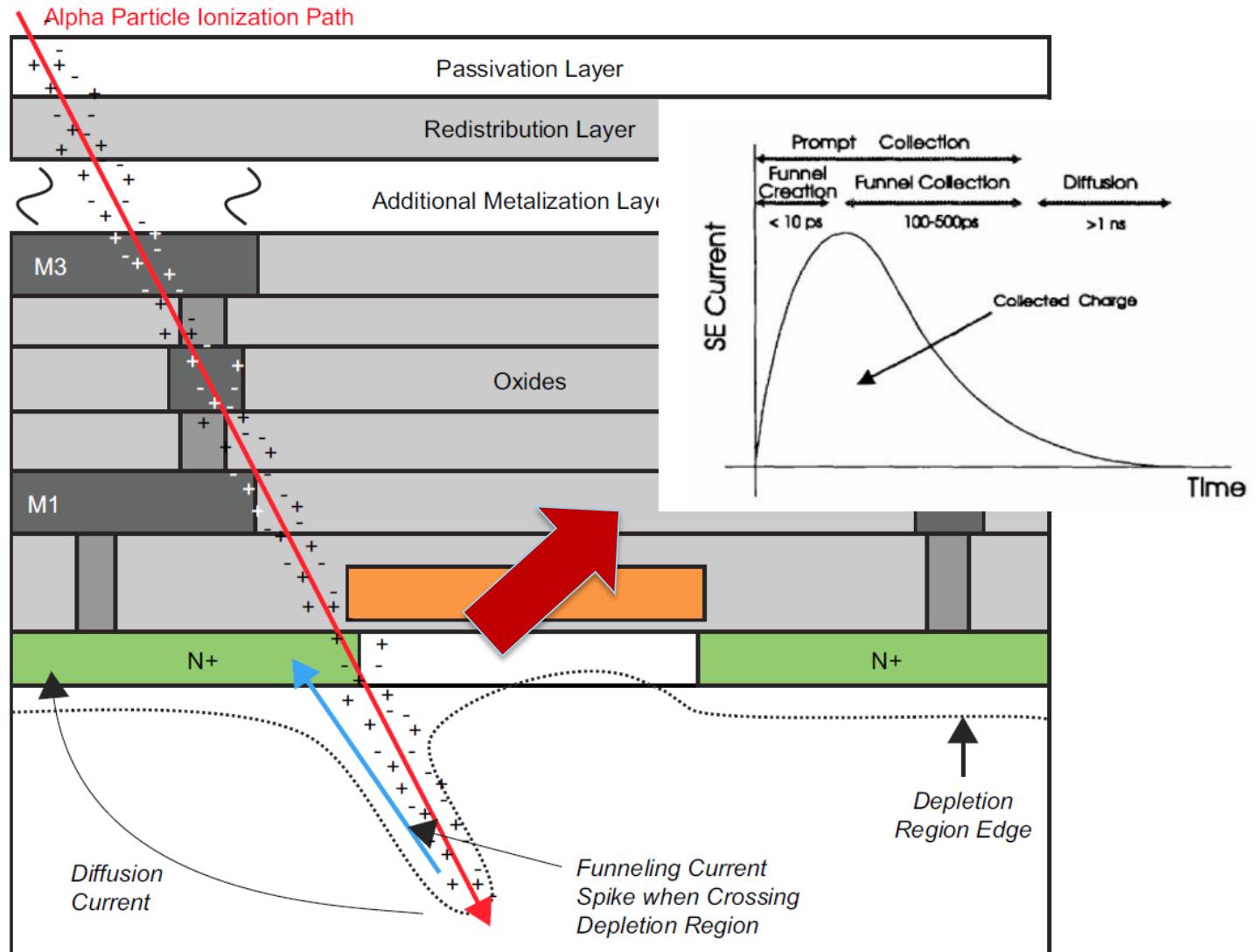


Image taken from *Introduction to SEU, Altera, white paper, Sep 2013*

# Single Event Effects (SEE) Classification

- Permanent (Hard Errors)

- Single Event Burnout
  - Single Event Gate Rupture
  - Single Event Latchup

- Soft Errors

- Transient

- Single Event Transient (SET)

- Static

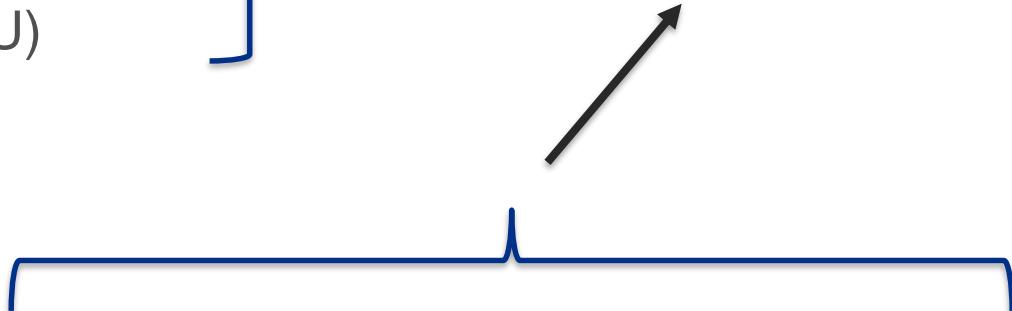
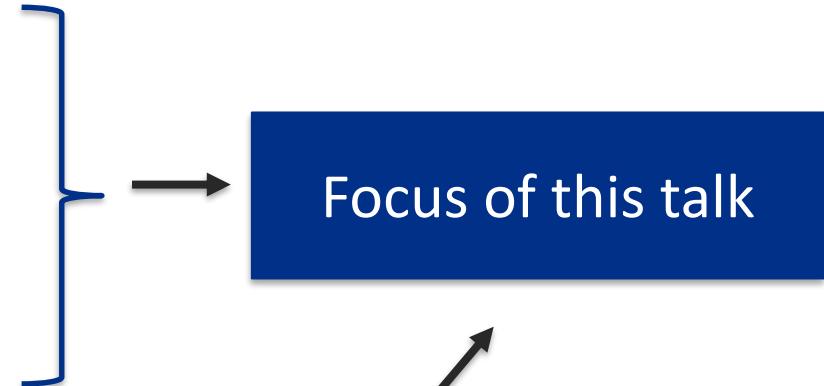
- Single Event Upset (SEU)

- Soft Error mitigation

- Technology level

- Cell level

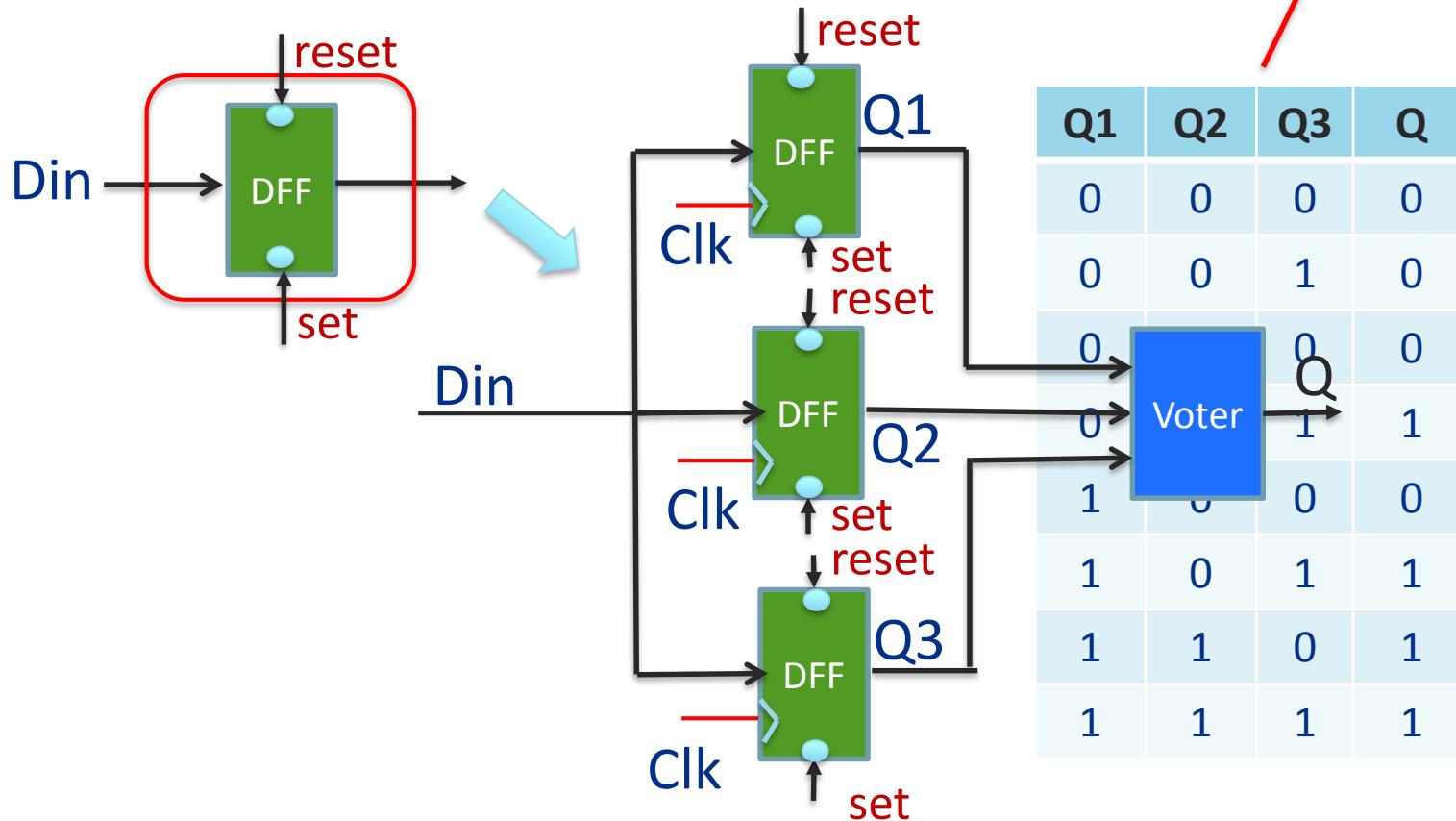
- System level redundancy : **Triple Modular Redundancy (TMR)**



# Triple Modular Redundancy (TMR)

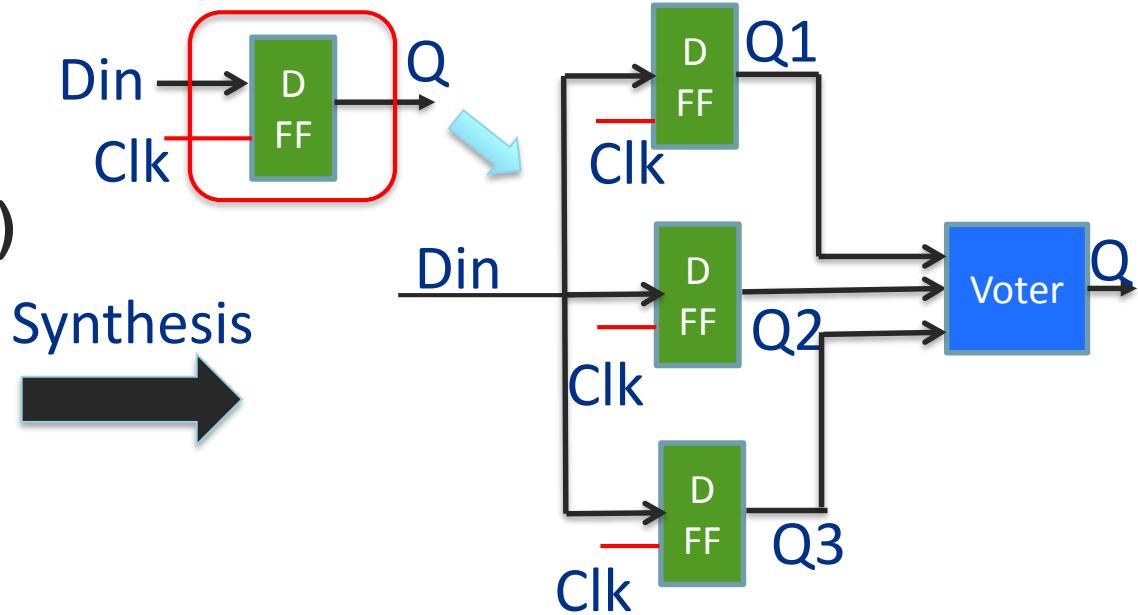
- Data is replicated on multiple nodes
- Additional area and power consumption

$$Q = (Q_1 \& Q_2) | (Q_2 \& Q_3) | (Q_3 \& Q_1)$$



# TMR Insertion by RTL designers

```
module TMR (Din, Q, clk);
Input Din;
output Q;
reg Q1, Q2, Q3;
always @(posedge clk)
begin
    -Q <= Din;
    Q1 <= Din;
    Q2 <= Din;
    Q3 <= Din;
end
assign Q = (Q1 & Q2) | (Q2 & Q3) | (Q3 & Q1)
endmodule
```

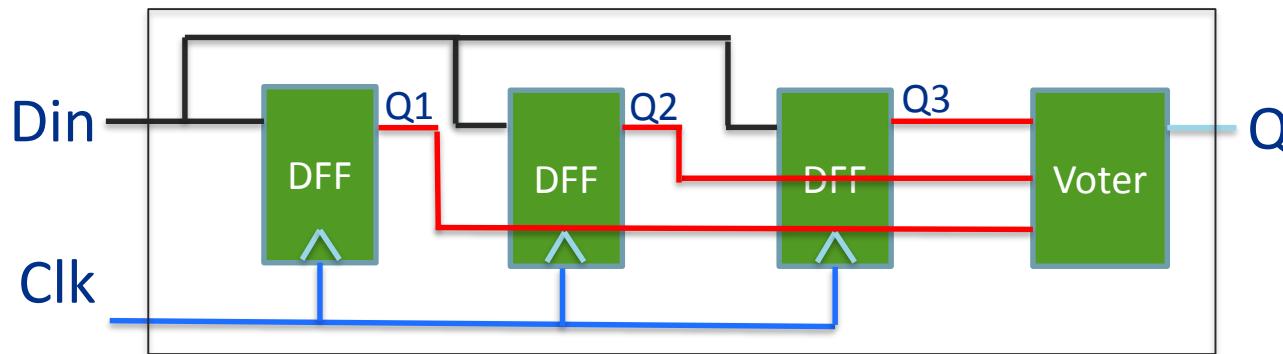


## Disadvantages:

1. Redundant logic is removed by synthesis tools
2. Additional don't touch commands
3. Complex for chips with huge # of registers

# TMR Macro

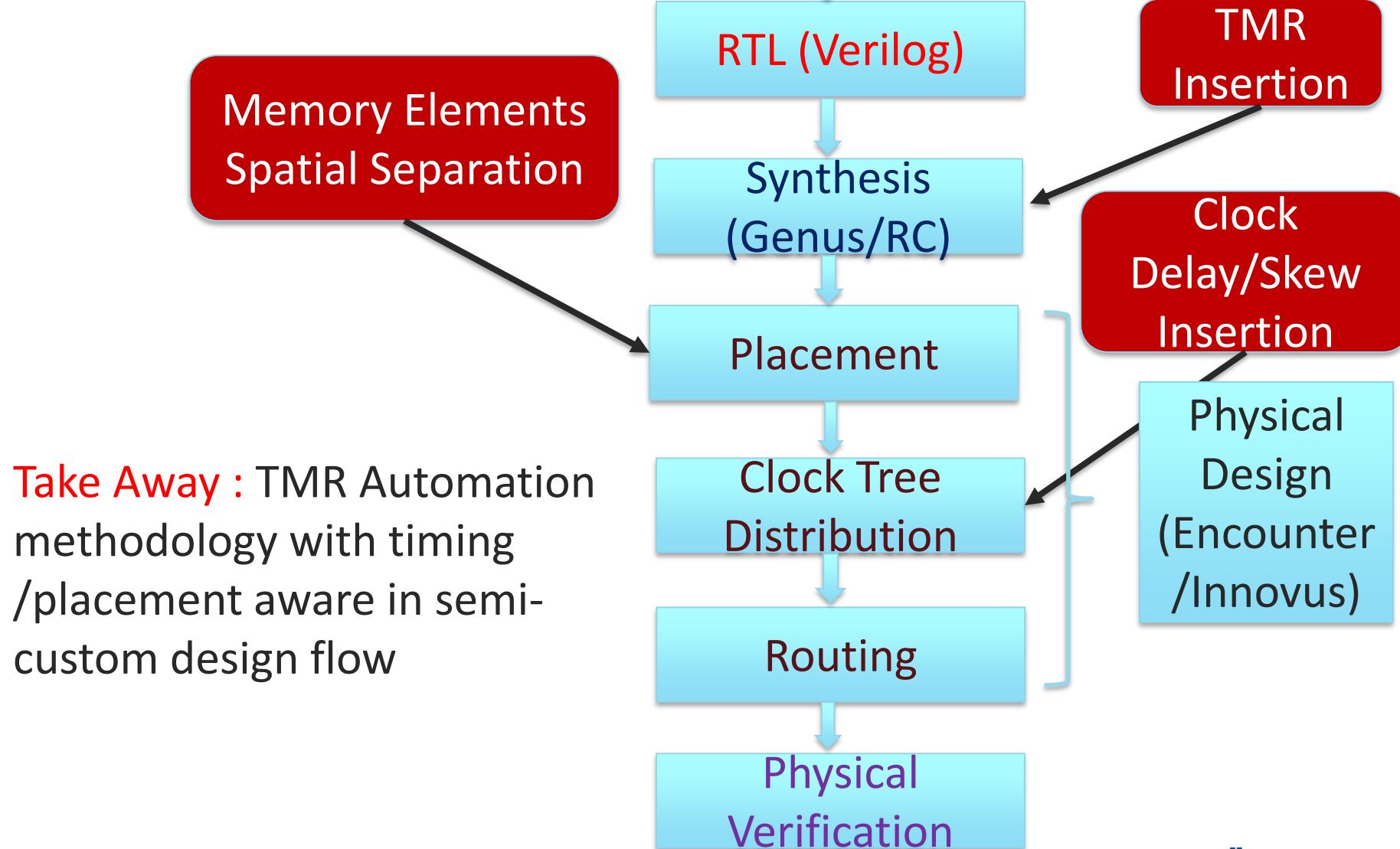
- Realize TMR cell as a macro
  - Single row implementation
  - Use this macro as a standard cell
- After synthesis, replace the registers with TMR Macro



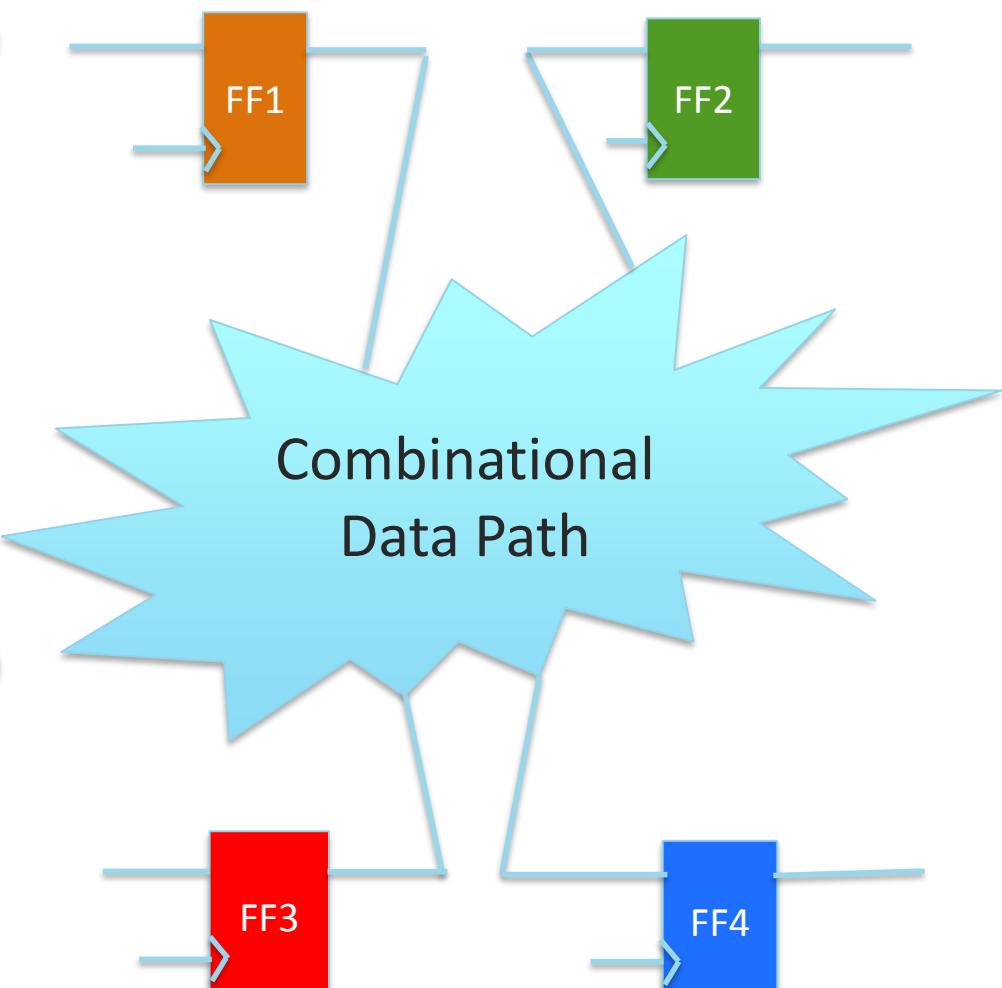
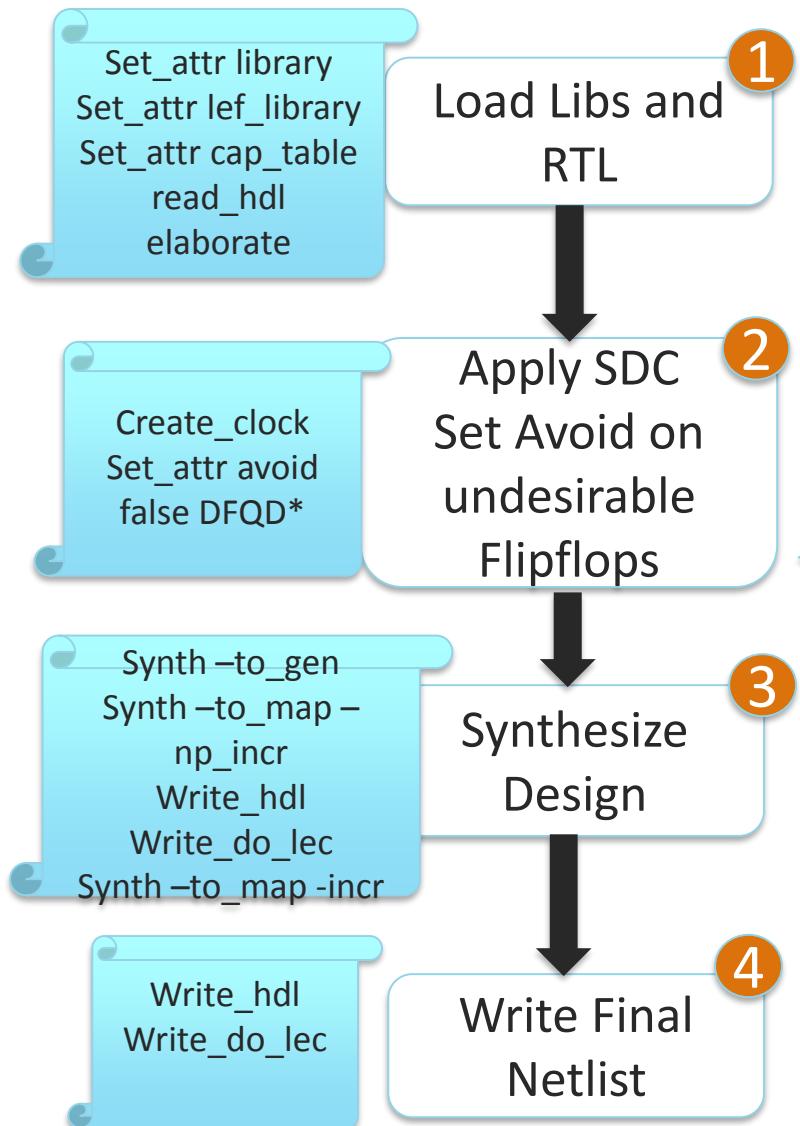
## Disadvantages:

1. Non-optimal placement
2. Routing issues

# Semi-Custom Flow



# Logic Synthesis Flow



# TMR Synthesis

Set\_attr library  
Set\_attr lef\_library  
Set\_attr cap\_table  
Read\_hdl  
elaborate

1 Load Libs and RTL

Create\_clock  
Set\_attr avoid false DFQD\*

2 Apply SDC  
Set Avoid on undesirable  
Flipflops

Synth-to\_gen  
Synth-to\_map -  
np\_incr  
Write\_hdl  
Write\_do\_lec  
Synth-to\_map -incr

3 Synthesize Design

```
Read_netlist LTMR.v
foreach i [filter -regexp libcell
DFQD* [find /designs/$DESIGN -
instance instances_seq/*tmr*]] {
Change_link -inst $i -design_name
/designs/tmr_flop
}
```

4 read TMR Netlist  
Use Change Link to convert  
desired flops to TMR

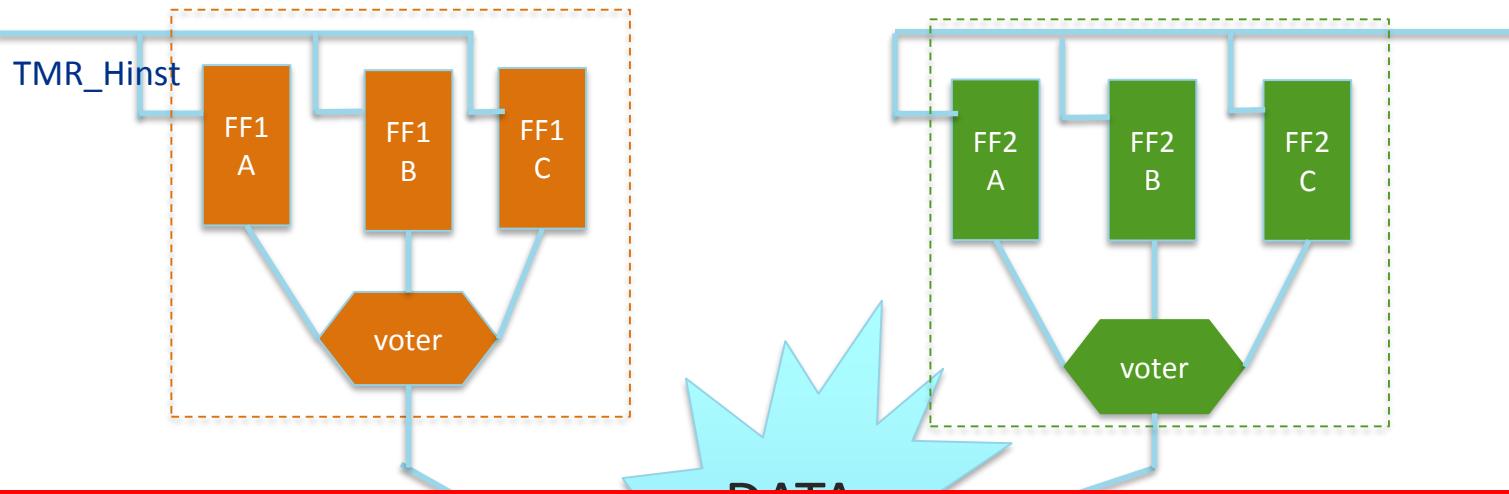
5 Remove TMR flop design  
Uniquify

```
rm /designs/LTMR
Uniquify $DESIGN
Set_attr preserve
size_ok [find / -
subdesign TMR*]
Synth-to_map incr
```

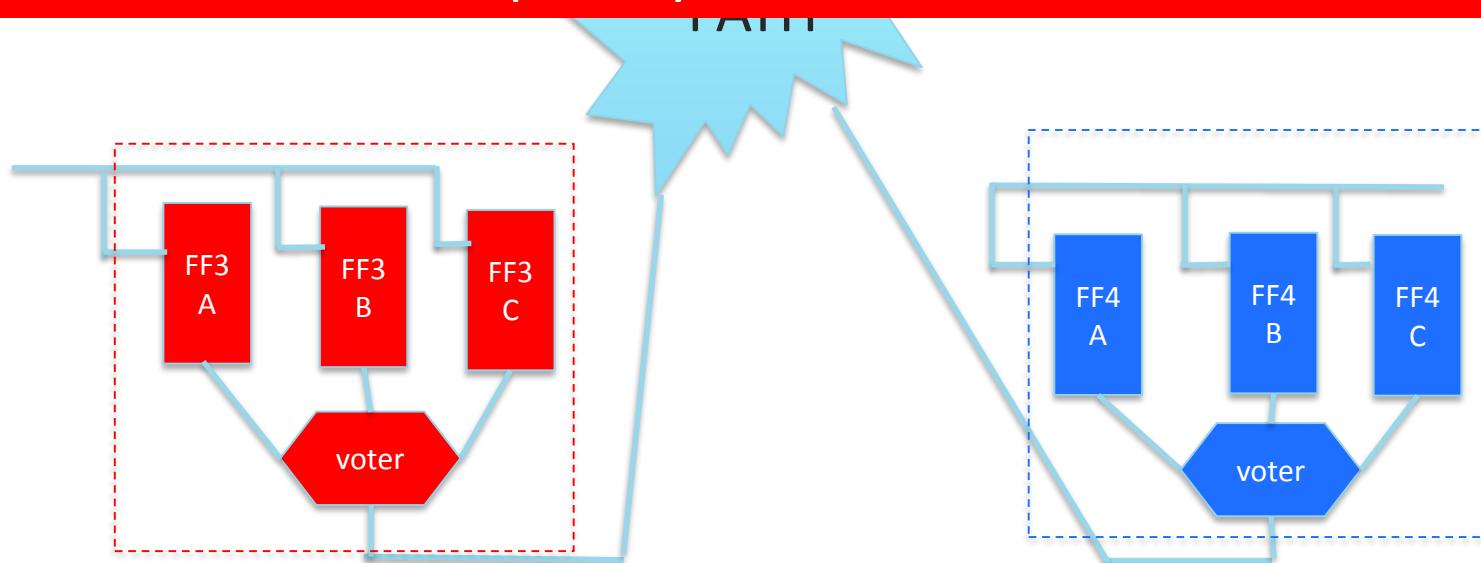
6 Write Final  
Netlist

Write\_hdl  
Write\_do\_lec

# TMR Insertion Automation



Additional steps in synthesis flow does TMR insertion



```
read_netlist TMR_DFQD.v  
foreach I [filter -regexp libcell DFQD* [find /designs/$DESIGN -instance instances_seq/*reg*]] {  
    Change_link -inst $i -design_name /designs/TMR_DFQD  
}
```

TMR\_DFQD.v

```
module TMR_DFQD (CP, D, Q);  
input CP, D;  
output Q; // Each flipflop type must have its corresponding TMR netlist  
wire CP, D1, D2, D3, Q, Q1, Q2, Q3, n_0;  
    DFQD1 DFFQ2_reg(.CP (CP), .D (D), .Q (Q2));  
    DFQD1 DFFQ1_reg(.CP (CP), .D (D), .Q (Q1));  
    DFQD1 DFFQ3_reg(.CP (CP), .D (D), .Q (Q3));  
    MAOI222D0 p4324D(.A (Q1), .B (Q2), .C (Q3), .ZN (n_0));  
    CKND0 Fp4461A(.I (n_0), .ZN (Q));  
endmodule
```

4

```

read_netlist TMR_DFQD.v
foreach i [filter -regexp libcell DFQD* [find /designs/$DESIGN/* -
instance instances_seq/*reg*]] {
    change_link -inst $i -design_name /designs/TMR_DFQD
}

```

Filter all the \*reg\* instances mapped to DFQD cell in the technology library

Change\_link, built-in command in RC compilers which Replaces the cell with LTMR cell

Search for all the sequential instances with instance name consisting of \*reg\*

5

```

rm /designs/TMR_DFQD
Uniquify $DESIGN
Set_attr preserve_size_ok [find / -subdesign TMR*]
Synth -to_map incr

```

Delete independent designs

Incremental Synthesis

# TMR Insertion Scenarios

- Triplicating only the registers having \*tmr\* in RTL

```
module TOPModule ();  
    ...  
    reg in_tmr;  
    ...  
endmodule
```

```
read_netlist TMR_DFQD.v  
foreach I [filter -regexp libcell DFQD* [find  
/designs/$DESIGN/* -instance instances_seq/**tmr*]] {  
    Change_link -inst $i -design_name  
        /designs/TMR_DFQD  
}
```

- Triplicating all the registers in the RTL

```
module TOPModule ();  
    ...  
    reg in;  
    ...  
endmodule
```

```
read_netlist TMR_DFQD.v  
foreach I [filter -regexp libcell DFQD* [find  
/designs/$DESIGN/* -instance instances_seq/**reg*]] {  
    Change_link -inst $i -design_name  
        /designs/TMR_DFQD  
}
```

- Triplicating in one of the hierarchical module in RTL

```
module submodule();  
    reg test;  
    ....  
endmodule  
module TOPModule ();  
    ...  
    submodule hierInst();  
endmodule
```

```
read_netlist TMR_DFQD.v  
foreach I [filter -regexp libcell DFQD* [find  
/designs/$DESIGN/instances_hier/hierInst -instance  
    instances_seq/**reg*]] {  
    Change_link -inst $i -design_name  
        /designs/TMR_DFQD  
}
```

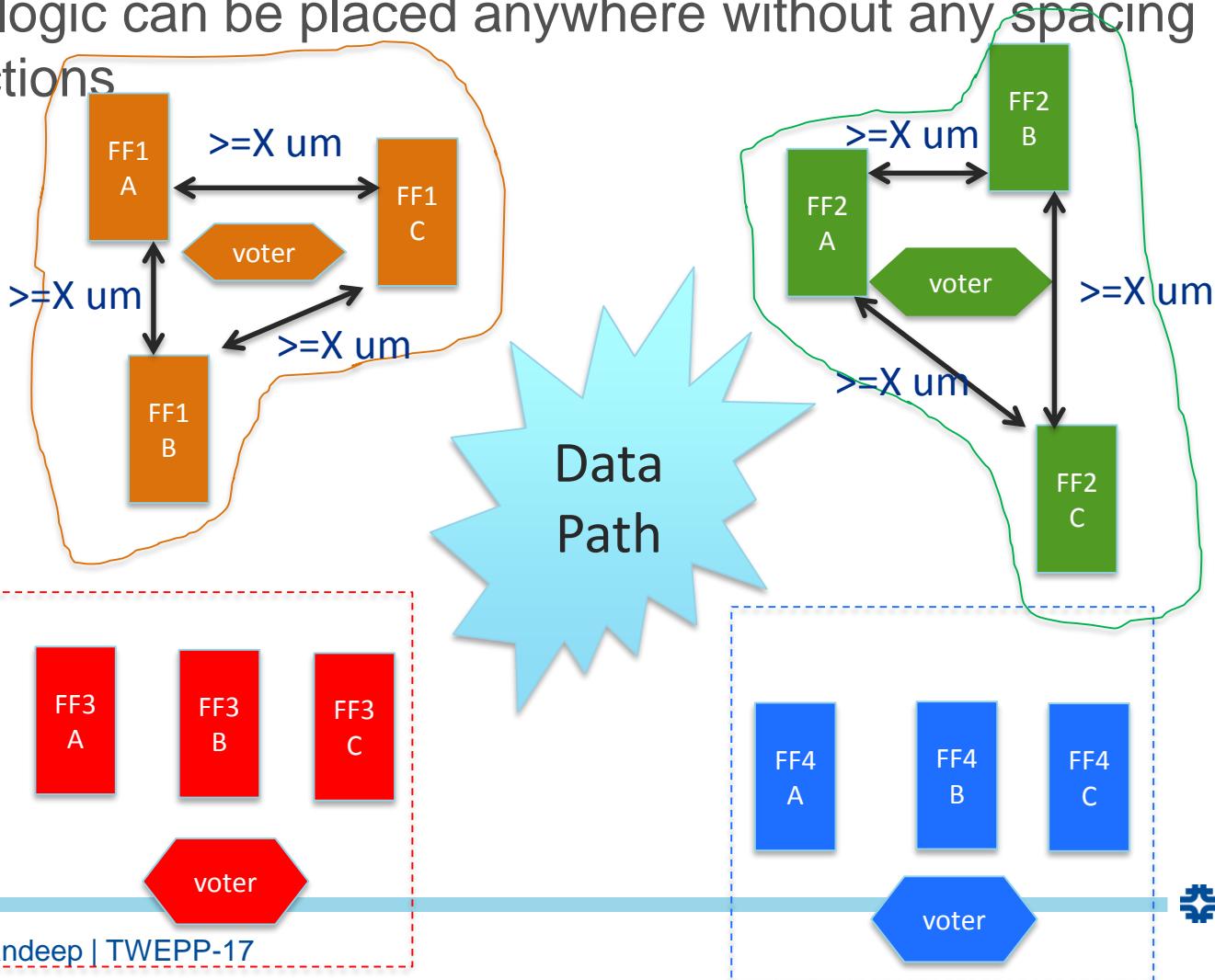
# Summary : TMR Synthesis Flow

- After the synthesis, convert the flops that require redundancy to a TMR flop using “**change\_link**” command
- The TMR modules are coded by hand and read into synthesis tool(RC/Genus) as a netlist
- One TMR module to be coded for each flop type (D-flop, Async Set, Async reset etc)
- “**avoid**” attribute can be used to limit the flop types chosen during synthesis

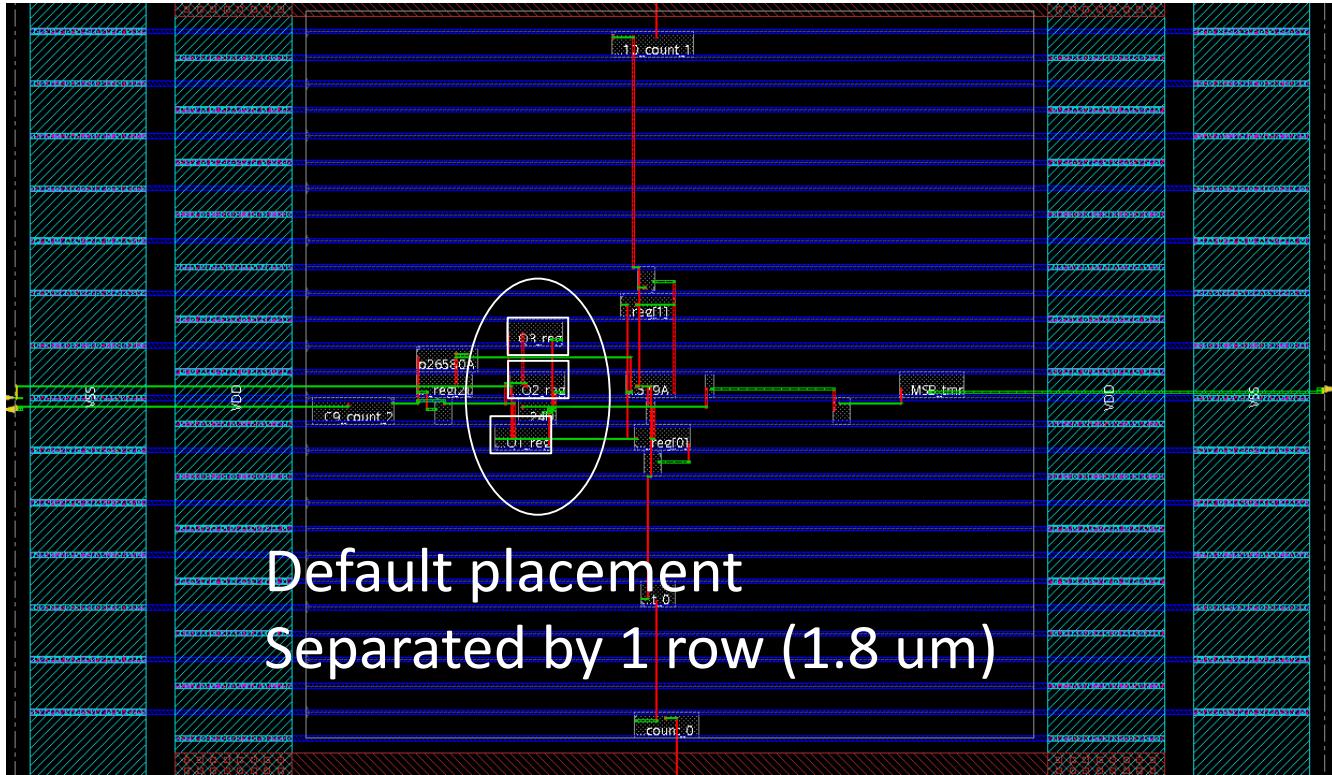
# Physical Design

- Requirement

- Flops within the same TMR module must be at least  $\geq X$  um apart
- Voter logic can be placed anywhere without any spacing restrictions



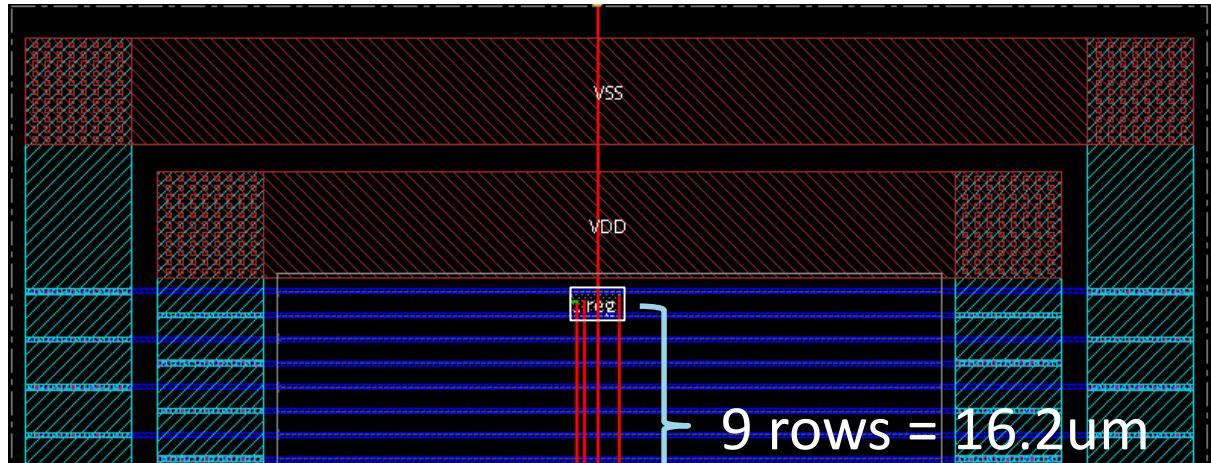
# Encounter Implementation



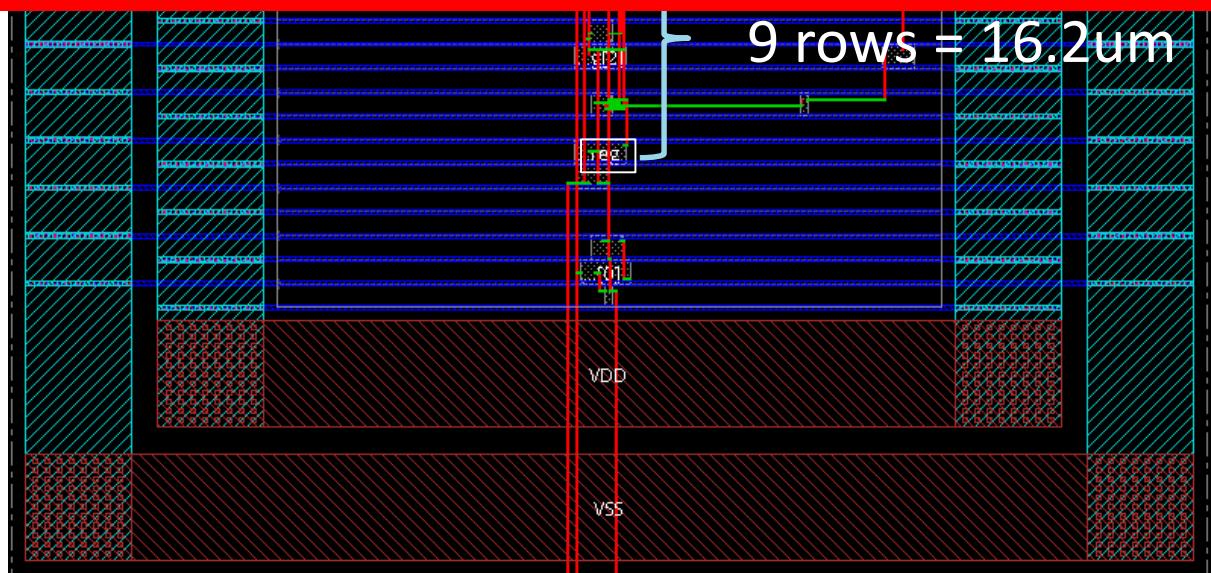
- Innovus 16.X has native space group constraints
  - Create space group constraint
  - Enable space group constraint during placement
  - Place design

# Innovus Implementation (v16.1)

- New built-in command  
`create_inst_space_group`

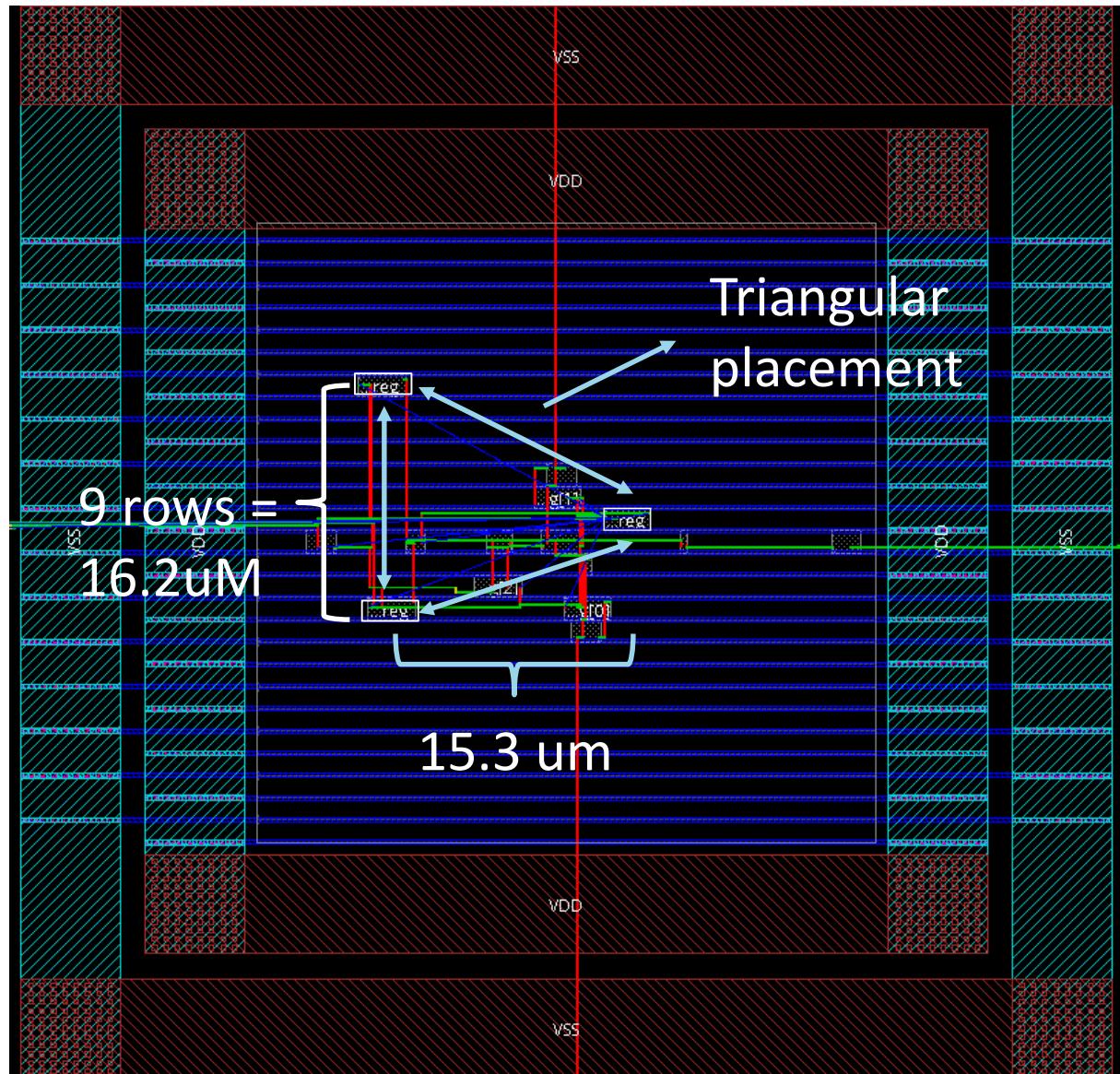


INN 16.1 can only separate cells vertically but does not explore horizontally



# INNOVUS 16.2

- Takes horizontal and vertical spacing during place & route
- ```
create_inst_space_group tmrSpaceGrp${n} -inst [ dbget [dbget$thPtr.allInsts.cell. isSequential 1 - p2].name ] -spacing_x $reg_spacing - spacing_y $reg_spacing
```



```
### Spacing distance
```

```
set LEN 15
```

```
### TMR Module Name Prefix
```

```
selectInstByCellName TMR*
```

built-in command  
in Innovus

```
### Foreach TMR Module Create Spacing Group & Assign the Corresponding Flops To The Spacing Group
```

```
foreach thPtr [dbGet selected] {  
    create_inst_space_group tmrSpaceGrp${n} -inst [  
    dbget [dbget $thPtr.allInsts.cell.name DFQD* -p2].name ] -  
    spacing_x $LEN -spacing_y $LEN  
}
```

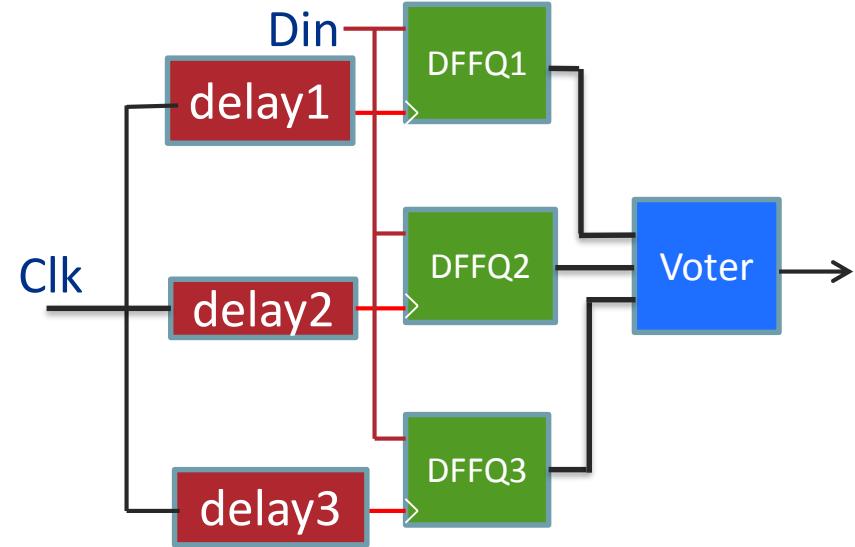
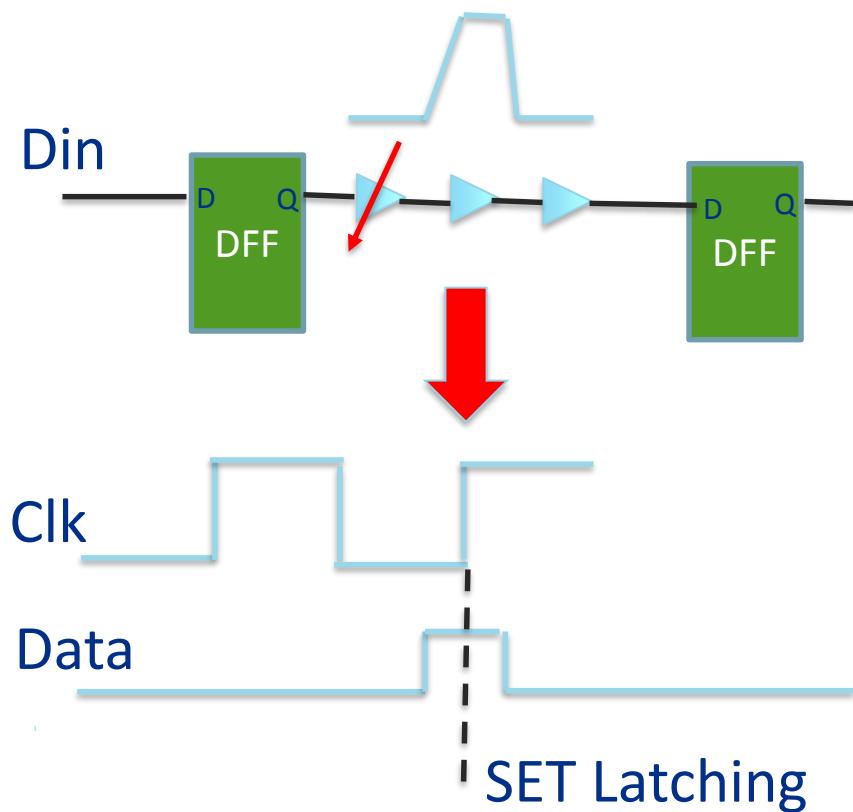
```
### Enable Spacing Constraint During Placement
```

```
setPlaceMode -place_detail_check_inst_space_group true
```

```
### Place Design
```

```
place_opt_design
```

# SET Mitigation



Simplest solution,  
Different delay values

# SET Mitigation : Clock Delay Insertion

```
set CkDel1 0.50
```

```
set CkDel2 1
```

*# Clock Insertion Delays on the TMR flops for SET Mitigation*

```
deselectAll
```

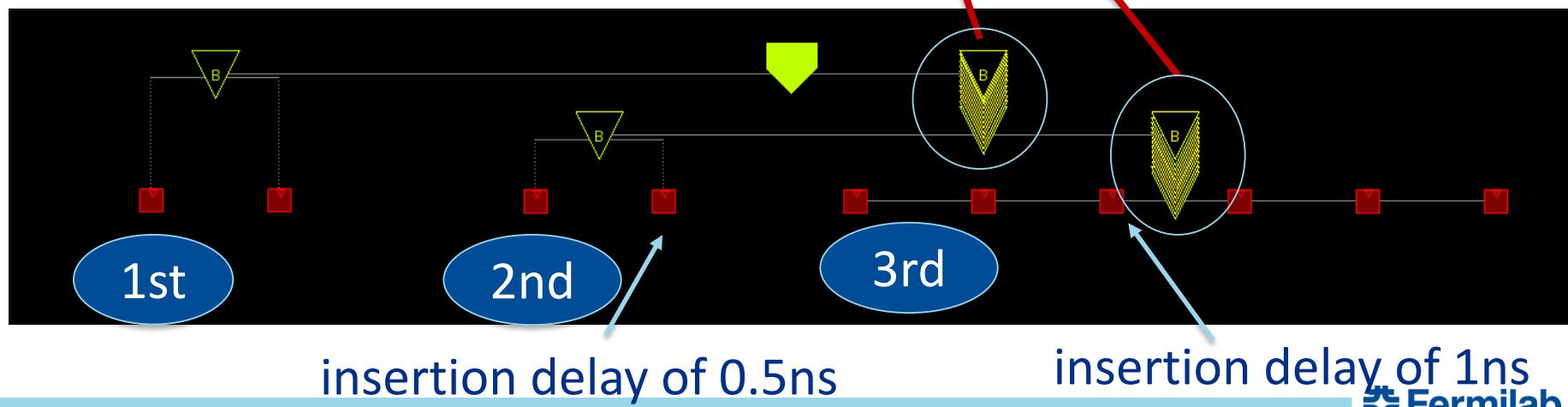
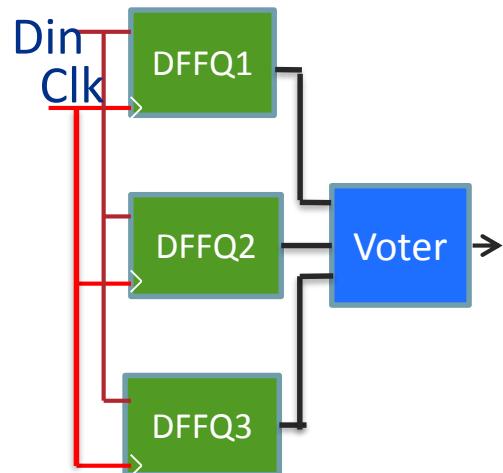
```
selectInstByCellName TMR*
```

```
foreach thPtr [dbGet selected] {
```

```
    foreach n [dbget $thPtr.allInsts.name *DFFQ2_reg* ] {  
        set_ccopt_property insertion_delay -delay_corner {av_typ_dc} $CkDel1 -pin ${n}/CP  
    }
```

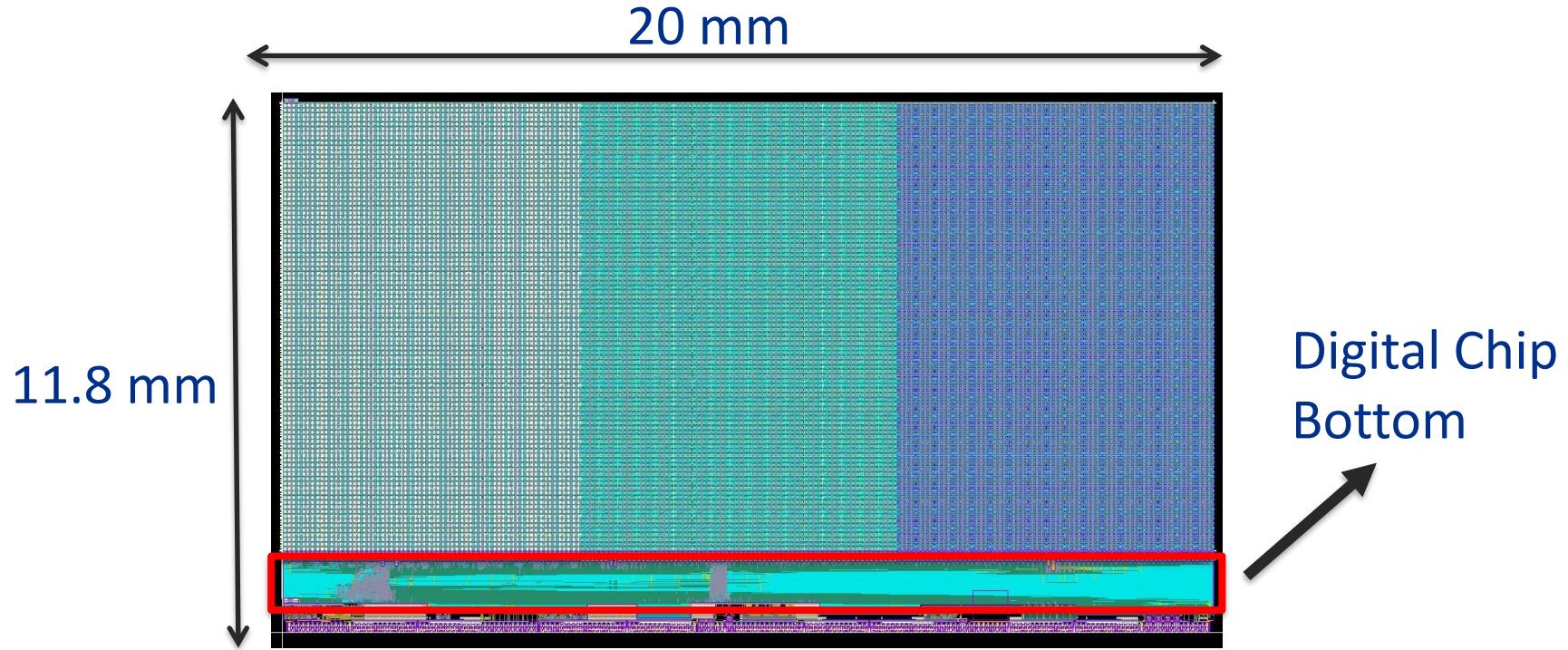
```
    foreach n [dbget $thPtr.allInsts.name *DFFQ3_reg* ] {  
        set_ccopt_property insertion_delay -delay_corner {av_typ_dc} $CkDel2 -pin ${n}/CP  
    }
```

```
}
```



# RD53A Context

- Joint effort between Atlas & CMS groups
- SEEs in global configuration
- SEEs along data path registers
- Spatial separation: 15um and SET pulse width: ~0.5ns



# Conclusions

- TMR insertion during synthesis
  - Works for Genus/RC
- Specific distance between memory elements during physical design
  - INN 16.X version
- SET mitigation with clock delay insertion
- Timing/Placement aware TMR insertion methodology

# Acknowledgements

- RD53 Collaboration : Special thanks to
  - Jorgen Christiansen (CERN)
  - Tomasz Hemperek (UBONN)
- ASIC group, Fermilab
- Cadence Support

!!! THANK YOU FOR YOUR ATTENTION !!!