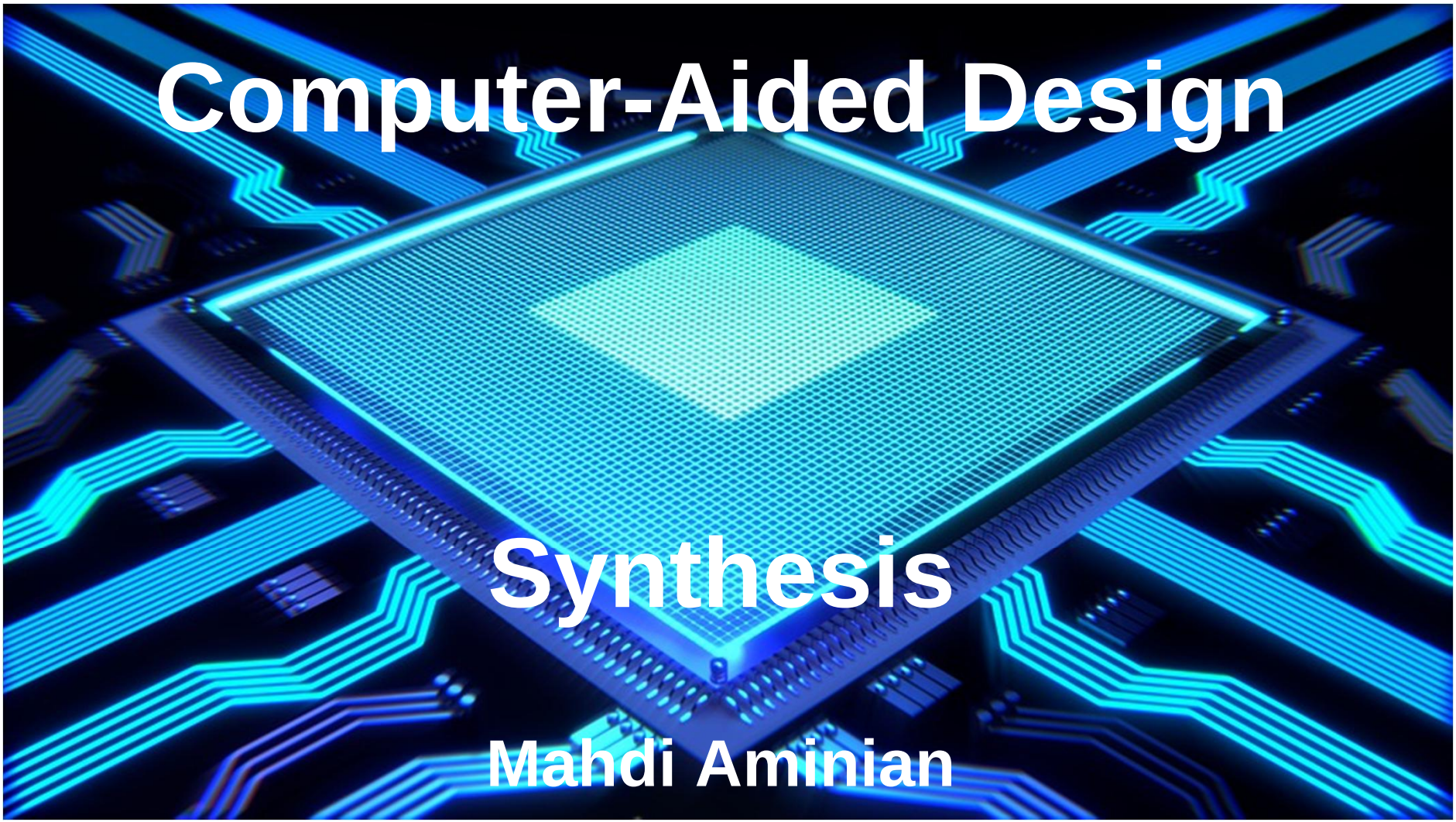


Computer-Aided Design

Synthesis

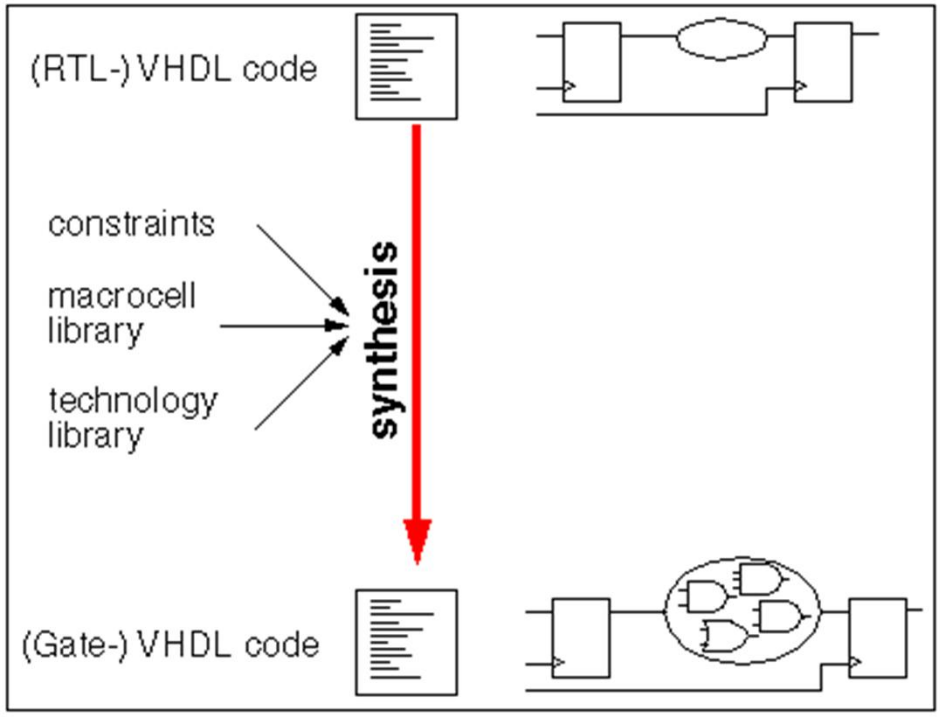
Mahdi Aminian

The University Of Guilan
Rasht - Iran

Some slides courtesy of:

- “Hardware Systems Modeling”, A. Vachoux, EPFL
- CAD slides from Dr. Saheb Zamani

What is Synthesis?



Transformation of an abstract description into a more detailed description

- "+" operator is transformed into a gate netlist
- "if (VEC_A = VEC_B) then" **à** a comparator which controls a multiplexer

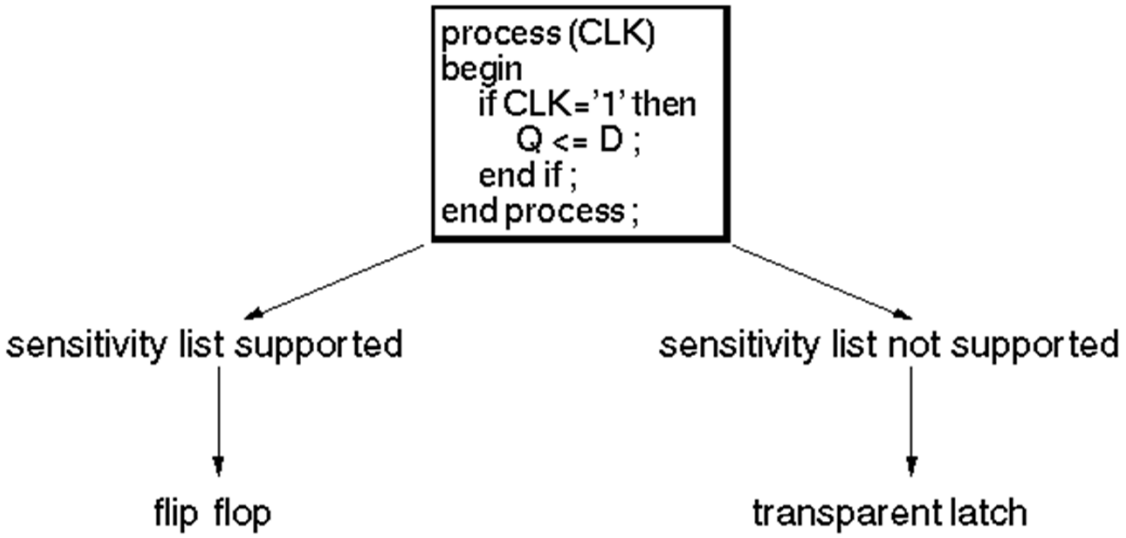
Transformation depends on several factors:

- **Algorithm, constraints, library**

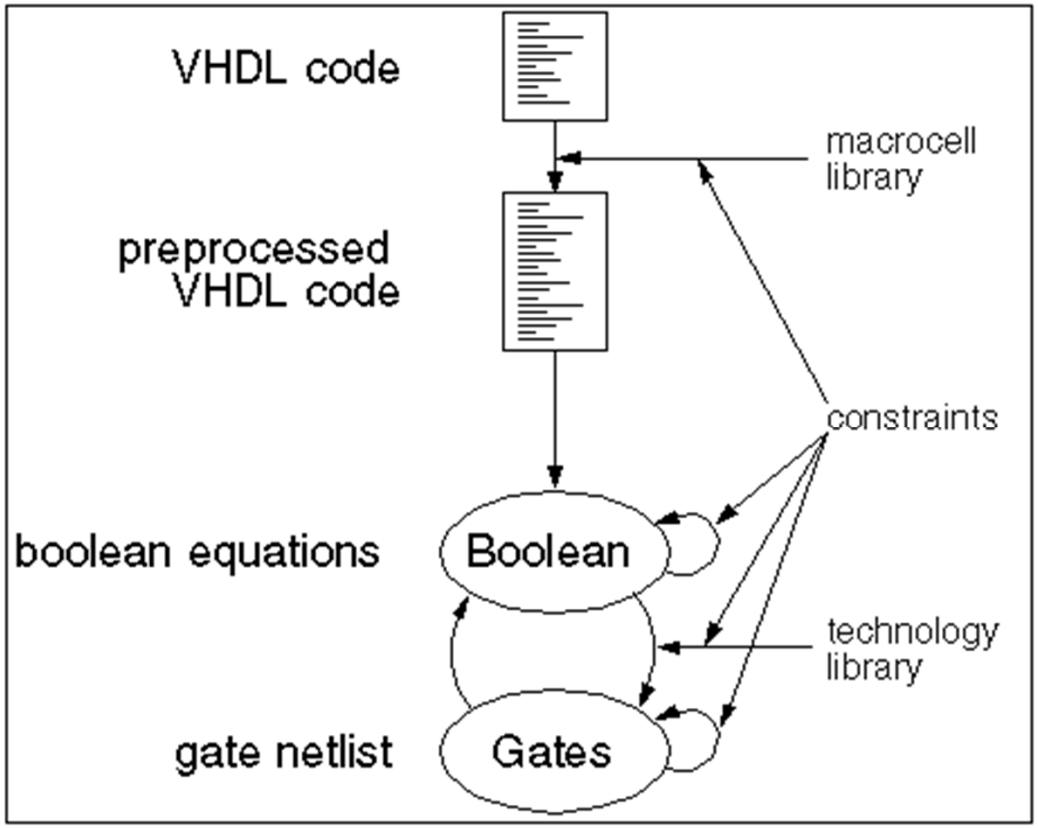
- **عملگرهای ساده (مثل AND، OR، مقایسه) به گیتهای مشخصی تبدیل می شوند اما عملگرهای پیچیده تر مثل ضرب ابتدا به ماکروسلهای خاص آن tool تبدیل می شوند.**

Synthesizability

- **Only a subset of VHDL is synthesizable**
- **Different Tools support different subsets**
  - records?
  - arrays of integers?
  - clock edge detection?
  - sensitivity list?
  - ...
- **Different Language Support for Synthesis**



How to Do?



- **Macrocells**
  - adder
  - comparator
  - bus interface
  - counter
- **Constraints**
  - speed
  - area
  - power
- **Optimizations**
  - boolean: mathematic
  - LUTs: technological

Constraints

- محدودیتهای سخت: محدودیتهای ثابتی که در سنتزکننده وجود دارد:
  - مربوط به محدودیتهای target technology (مثل محدودیت fanout در Logic Blockها)
  - و مربوط به توانایی tool
- محدودیتهای نرم: محدودیتهایی که کاربر مشخص می کند:
  - مثل حداقل سرعت لازم

Non-functional requirements

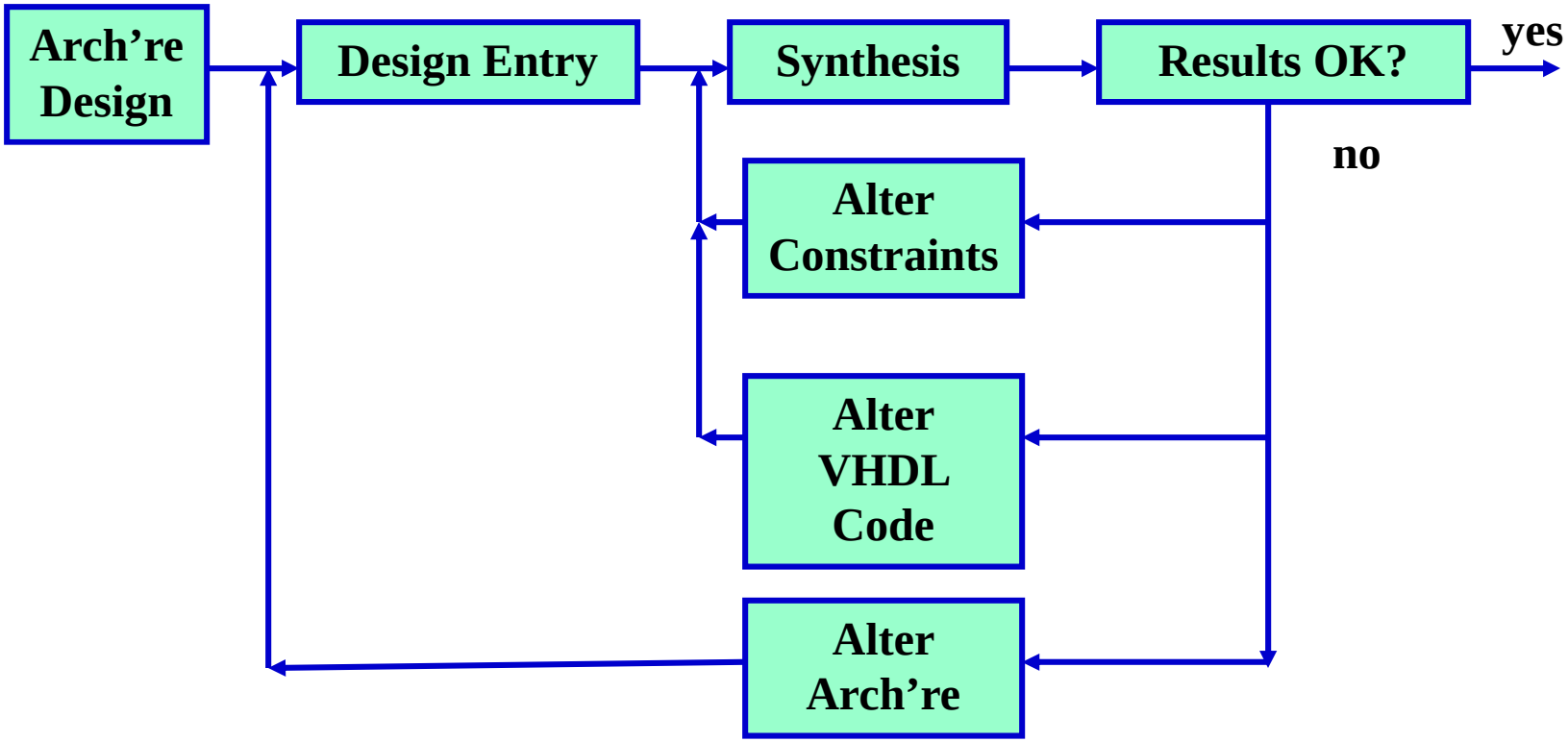
- Performance:
  - Clock speed is generally a primary requirement.
  - Usually expressed as a lower bound.
    - Design cycle and Timing Closure
- Size:
  - Determines manufacturing cost.
  - If your design doesn't fit into one size FPGA, you must use the next larger FPGA.
  - For very large designs: multi-FPGAs.
- Power/energy:
  - Power/Energy related to battery life and heat.
    - May have more cost:
      - More expensive packaging to dissipate heat.
      - More extreme measures (e.g. cooling fans).
  - Many digital systems are power- or energy-limited.

Mapping into an FPGA

- Must choose the FPGA:
  - Capacity.
  - Pinout/package type.
  - Maximum speed.
- Essential Information for Synthesis
  - Load values
  - Path delays
  - Driver strengths
  - Timing
  - Operating conditions (e.g. temperature)

Synthesis Process in Practice

- باوجود مکانیزمهای بهینه سازی، ممکن است بعد از سنتز، همهٔ محدودیتها برآورده نشده باشند ß تکرار



Problems with Synthesis Tools

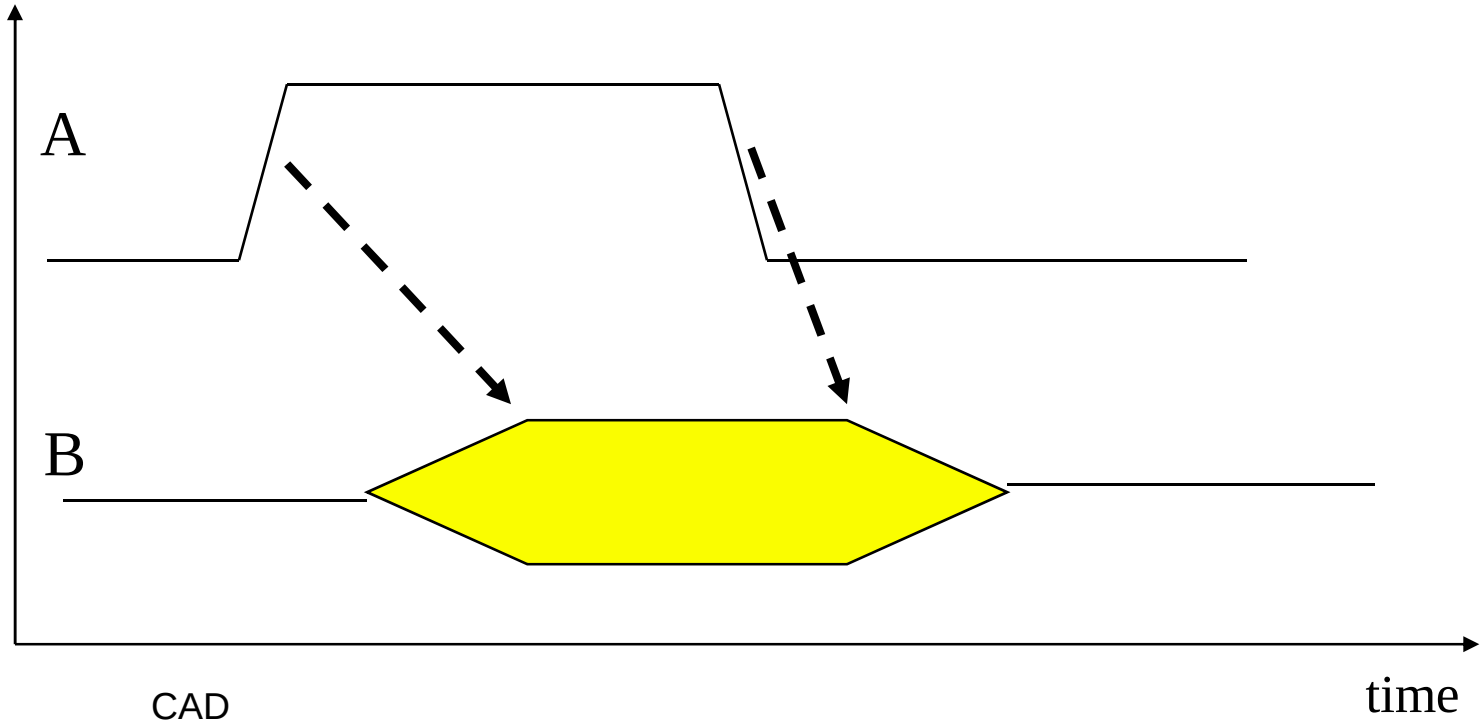
- **Timing issues**
  - layout information is missing during the synthesis process
  - clock tree must be generated afterwards
- **Complex clocking schemes**
  (inverted clocks, multiple clocks, gated clocks)
- **Memory**
  - synthesis tools are not able to replace register arrays with memory macro cells
- **Macro cells**
  - no standardized way for instantiation of existing technology macro cells

Synthesis Strategy

- **Consider the effects of different coding styles on the inferred hardware structures**
- **Appropriate design partitioning**
  - critical paths should not be distributed to several synthesis blocks
  - different optimization constraints may be used for separate blocks
- **Delay and Power Optimization**
  - Combinational network delay
  - Combinational network energy/power

Delay characteristics

- Measured from change in inputs to change in outputs.
- Data-dependent:
  - Some block delays depend on the value/waveform at the input ($t_{pHL} \neq t_{pLH}$) ($t_r \neq t_f$)
- May need to observe different paths through the network.

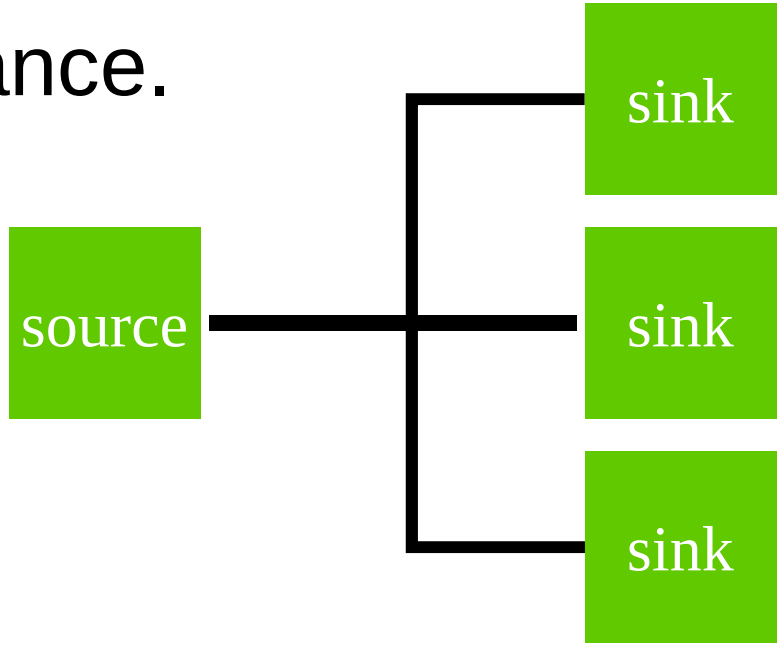


Sources of Delay

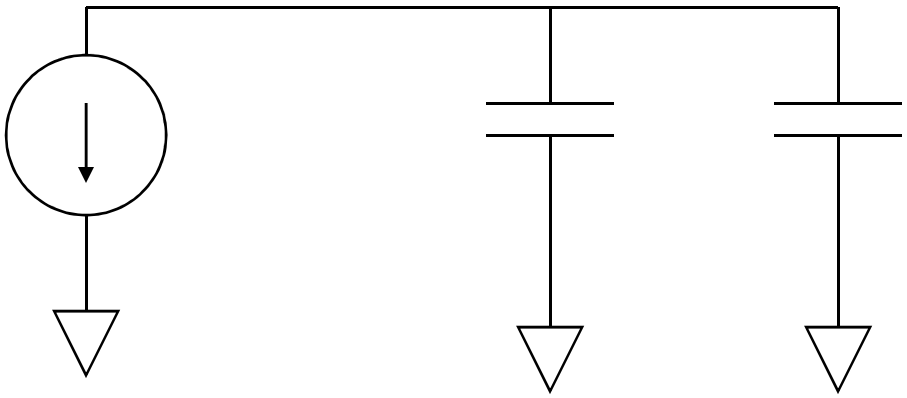
- Gate delay:
  - Little we can do about it
    - E.g. select another FPGA with faster logic blocks (LBs) or
    - Minimize the number of LBs in the critical path
- Wire delay:
  - Much we can do
    - E.g. select the proper path of the wire or
    - Select buffered paths.
  - Two types:
    - lumped load (for short wires which are modeled by a single capacitance)
      - Little we can do.
    - transmission line (for long wires).

Fanout

- Fanout adds capacitance.



- Adding gates adds capacitance:

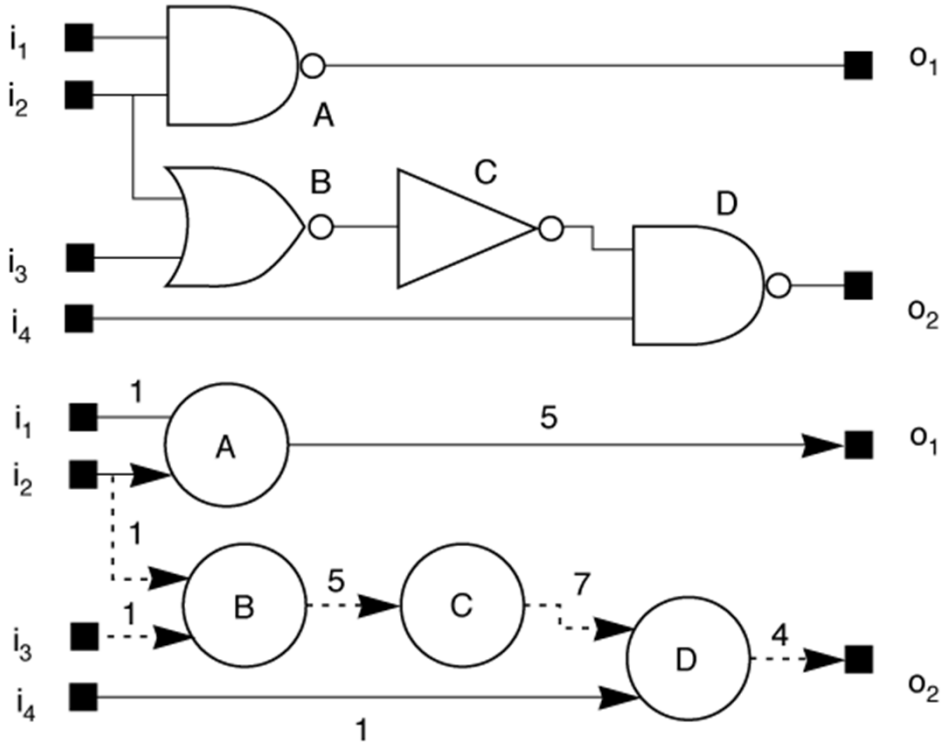


Path delay

- Combinational network delay is measured over paths through network.
- Can trace a causality chain from inputs to worst-case output.

network

graph model



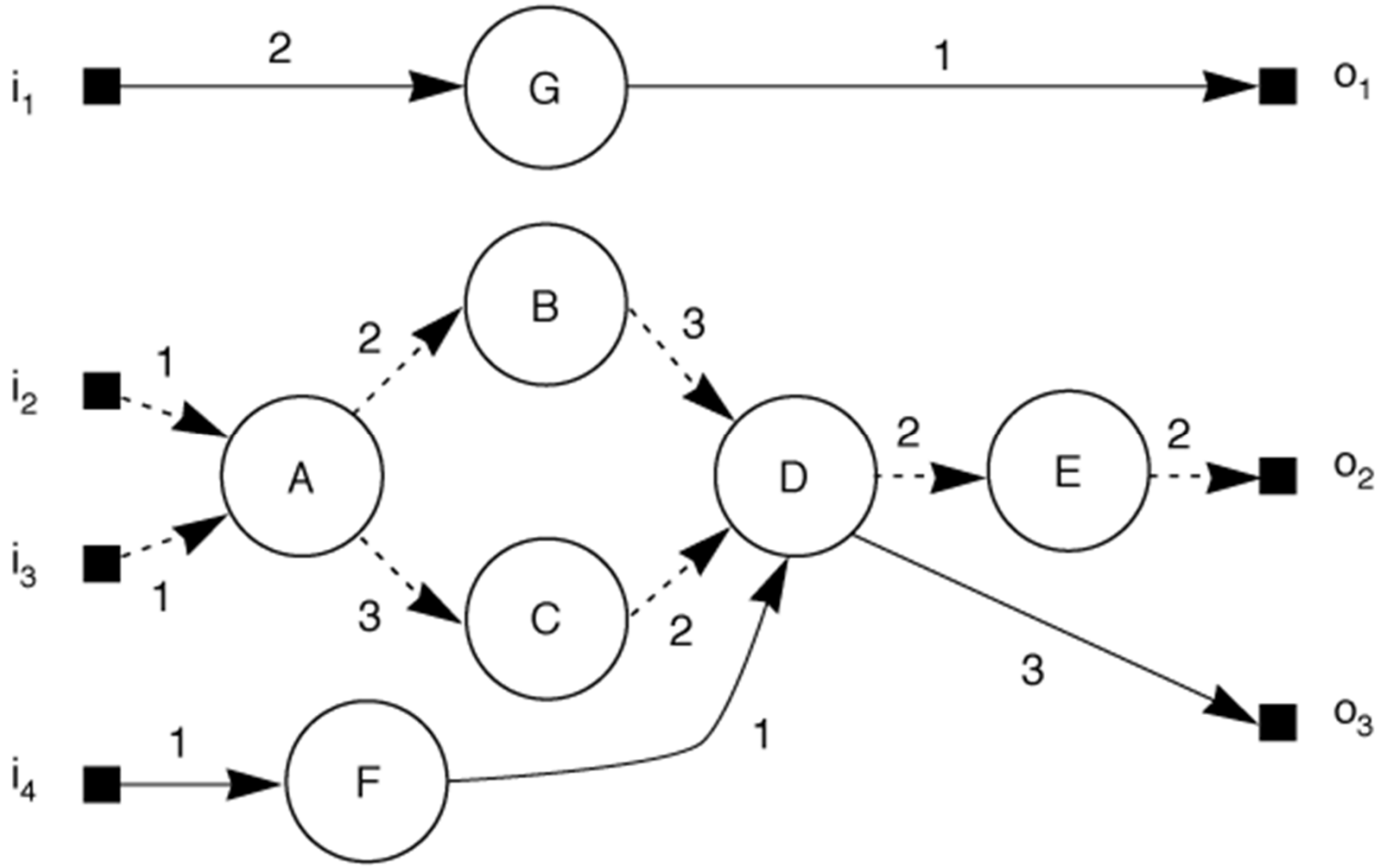
Critical path

- Critical path = path which creates longest delay.
- Can trace transitions which cause delays that are elements of the critical delay path.

Delay model

- Nodes represent gates.
- Assign delays to edges—signal may have different delay to different sinks.
- Lump gate and wire delay into a single value.

Critical path through delay graph



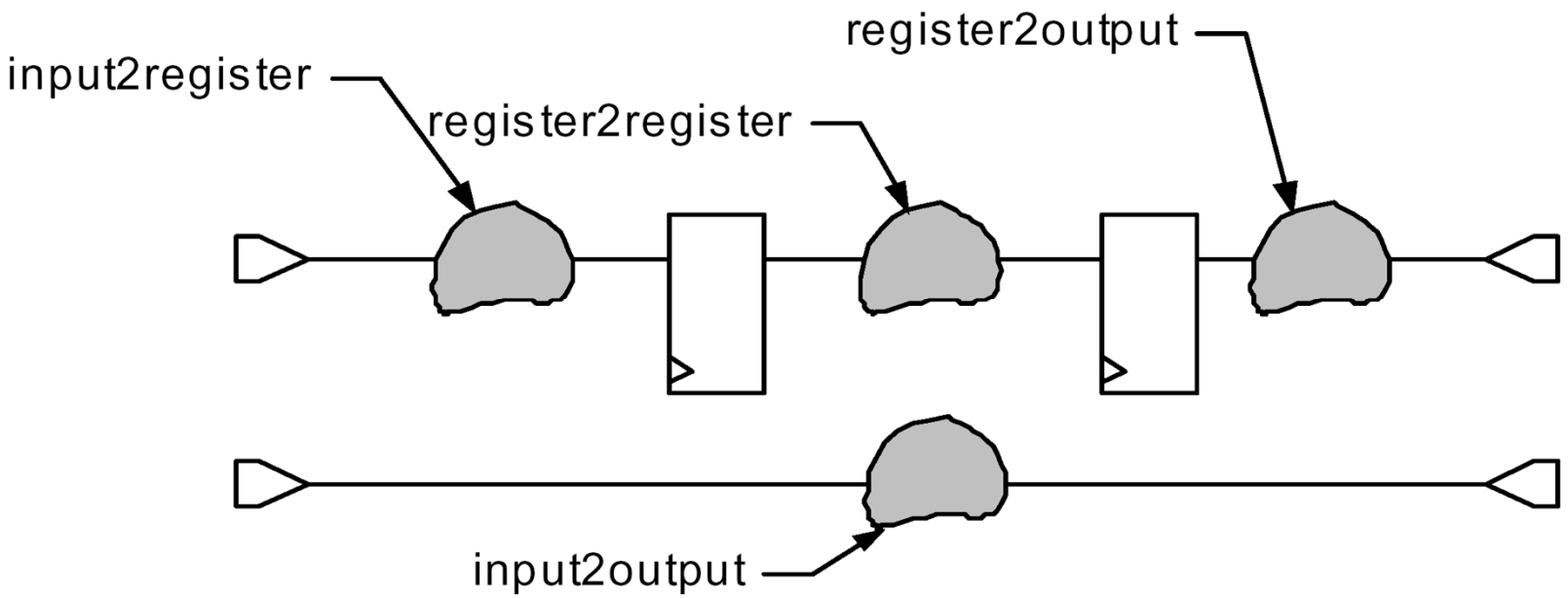
Reducing critical path length

- Must speed up the critical path
  - Reducing delay off the path doesn't help.

- There may be more than one path of the same delay.
  - **à** Must speed up all equivalent paths to speed up circuit.

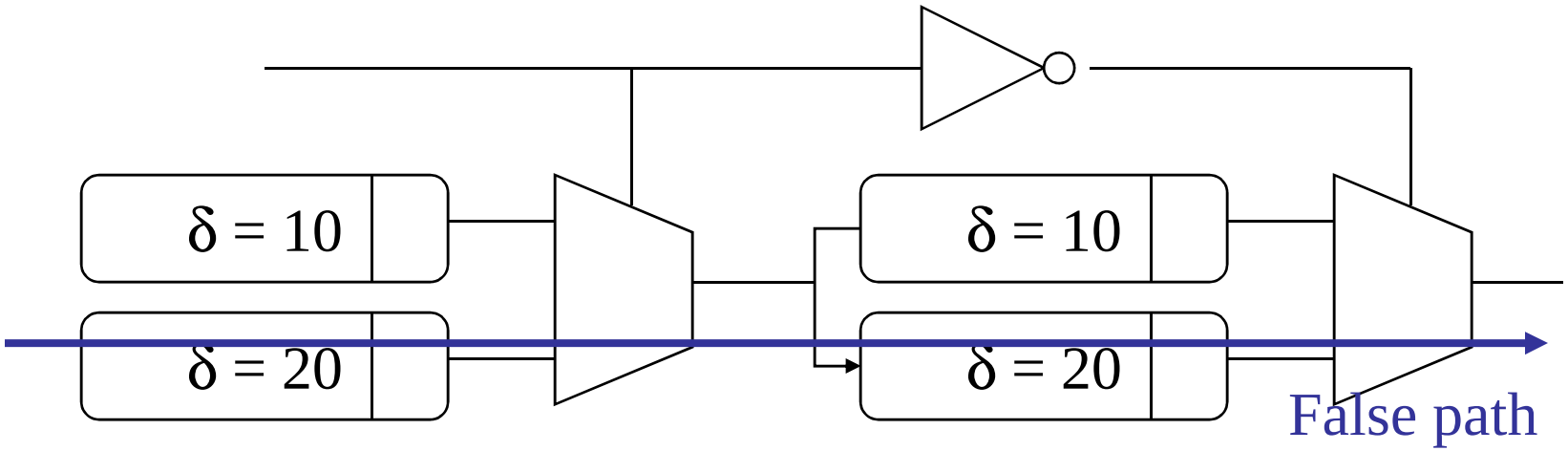
- Cutset: a set of edges that when removed, break the graph into two unconnected paths. (e.g. {(C,D), (B,D)} or {(D,E})
  - Must speed up cutset through critical path.

Delay Paths in a design



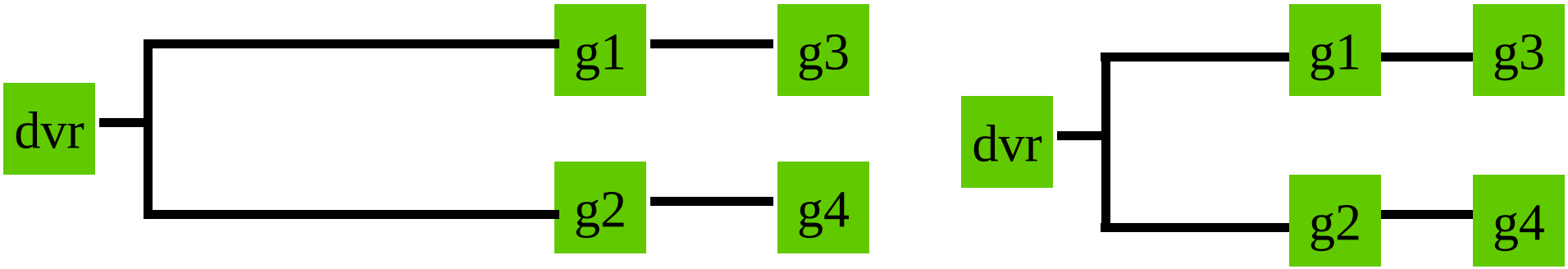
False paths

- Some input changes don’t cause output changes.
- A false path is a path which never happens due to Boolean gate conditions.
- False paths cause pessimistic delay estimates.



Placement and delay

- Placement helps determine gate distances.
- Gate distances determine routing.
- Routing determines wire length.
- Wire length determines capacitive load.
- Capacitive load determines delay.

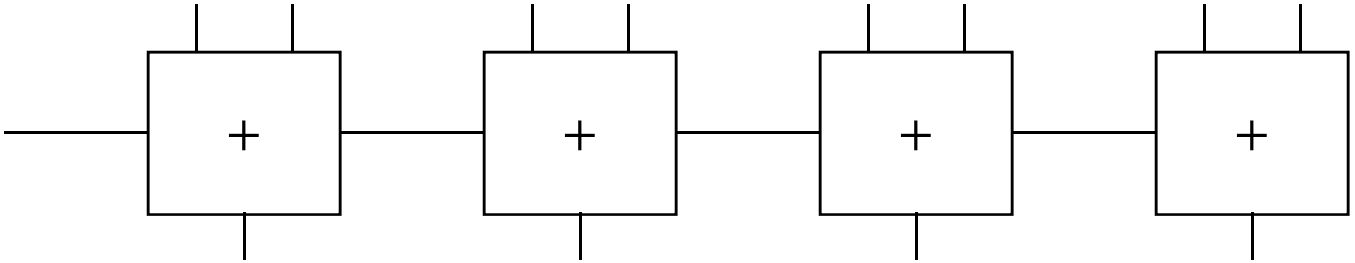


Optimizing network delay

- Identify the longest path(s).
- Improve delay along the longest path(s):
  - Driver delay
  - Wire delay
  - Logic restructuring

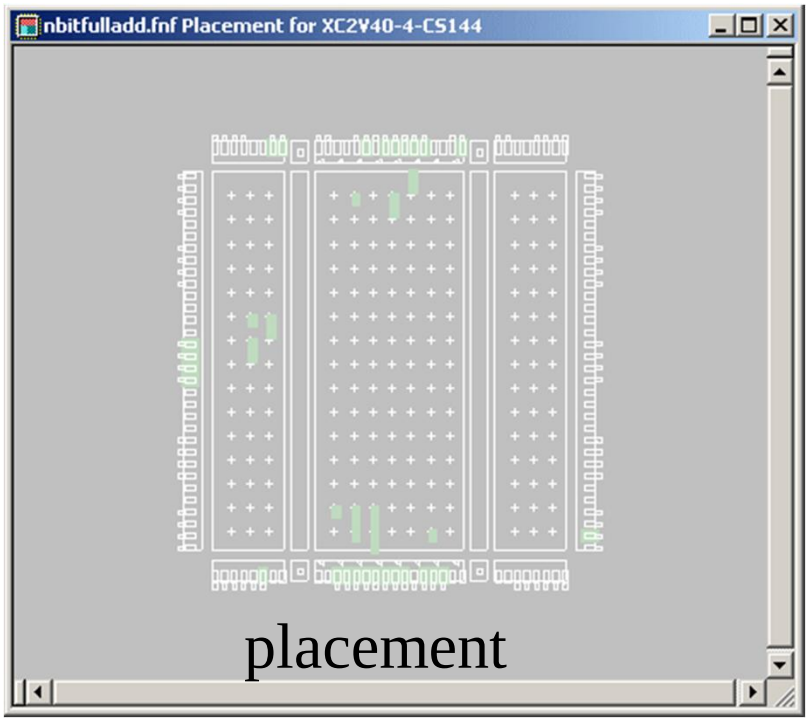
Example: Adder placement and delay

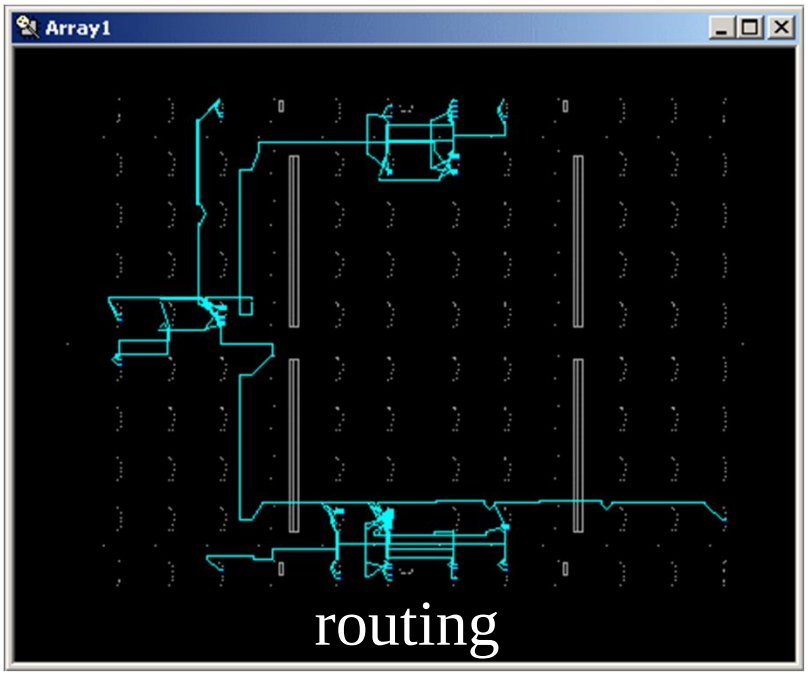
- N-bit adder: (optimal placement)



Bad placement and routing

With no delay constraints.

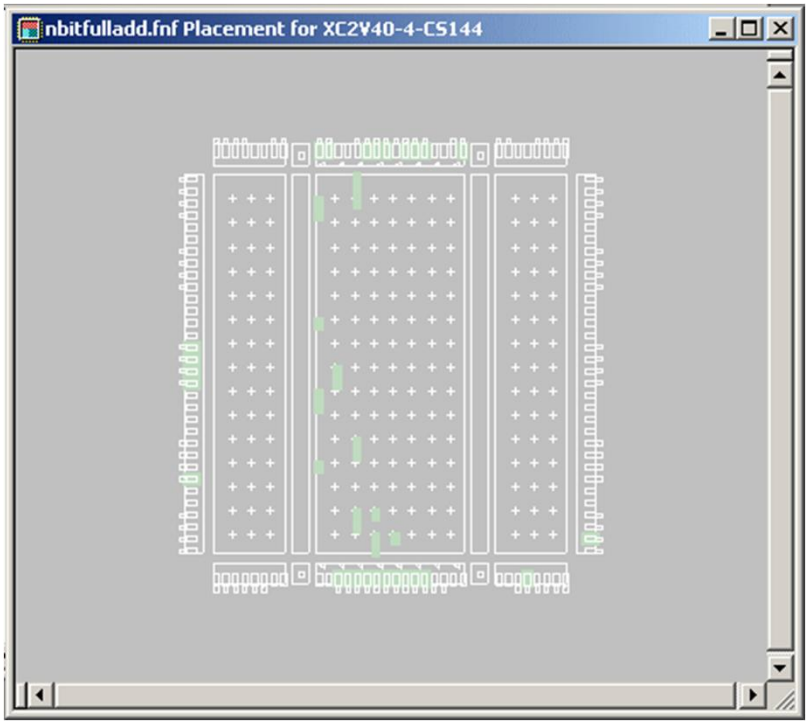


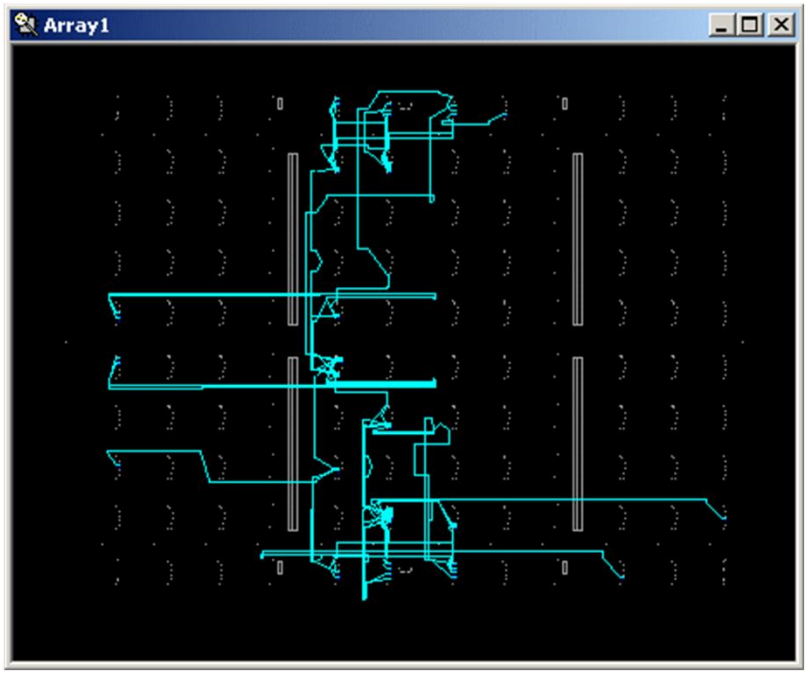


- Adder has been distributed throughout the FPGA.
- I/O pins have been spread around the chip.
- **à** P&R algorithms do not catch on to regularity.

Better placement and routing

With delay constraints.





• Better but far from optimal (less spread out horizontally but spread out vertically)

How to improve?

- Use macros (optimized),
- Put constraints on the placement of objects,
- Hand place objects

The FPGA design process

- Xilinx ISE (Integrated Synthesis Environment)
  - Translation from HDL
  - Logic synthesis
  - Placement and routing
  - Configuration generation

Design experiments

- Synthesize with no constraints.
- Synthesize with timing constraint.
  - Tighten timing constraint.
- Synthesize with placement constraints.
- Power:
  - Many tools don’t allow us to directly specify power consumption
  - Some tools allow us to specify power as an objective
  - May need to rewrite our h/w description for better power consumption characteristics.

Commercial Tools

- XST “-power” option reduces dynamic power consumption.
- Xilinx MAP and PAR“-power” option reduces dynamic power
  - But increases runtime and decreases design performance.
- Quartus-II has Power-Driven Synthesis and Place & Route.

Post-translation simulation model

- No timing or area constraints
- HDL model in terms of FPGA primitives.
- Example:

```
X_LUT4 \p12_Madd__n0015_Mxor_Result_Xo<1>1  (
   .ADR0(x_7_IBUF),
   .ADR1(y_13_IBUF),
   .ADR2(c12[7]),
   .ADR3(row12[8]),
   .O(row13[7])
 );
```

Mapping report

```
Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Logic Utilization:
  Number of 4 input LUTs:           501 out of   1,024   48%
Logic Distribution:
  Number of occupied Slices:         255 out of     512   49%
  Number of Slices containing only related logic:     255 out of     255  100%
  Number of Slices containing unrelated logic:         0 out of     255    0%
       *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:          501 out of   1,024   48%

  Number of bonded IOBs:             64 out of      92   69%

Total equivalent gate count for design:  3,006
Additional JTAG gate count for IOBs:  3,072
Peak Memory Usage:  64 MB
```

Static timing analysis report

Timing constraint: TS_P2P = MAXDELAY FROM TIMEGRP "PADS" TO TIMEGRP "PADS" 99.999 uS ;

20135312 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)

Maximum delay is 20.916ns.

After Mapping: à estimated delays (no information about interconnects)

Static timing report: delays along paths

Data Sheet report:

All values displayed in nanoseconds (ns)

Pad to Pad

Source Pad	Destination Pad	Delay
x<0>	p<0>	5.824
x<0>	p<10>	10.675
x<0>	p<11>	11.214
x<0>	p<12>	11.753

Routing report

Phase 1: 1975 unrouted; REAL time: 11 secs

Phase 2: 1975 unrouted; REAL time: 11 secs

Phase 3: 619 unrouted; REAL time: 12 secs

Phase 4: 619 unrouted; (0) REAL time: 12 secs

Phase 5: 619 unrouted; (0) REAL time: 12 secs

Phase 6: 619 unrouted; (0) REAL time: 12 secs

Phase 7: 0 unrouted; (0) REAL time: 12 secs

The NUMBER OF SIGNALS NOT COMPLETELY ROUTED for this design is: 0

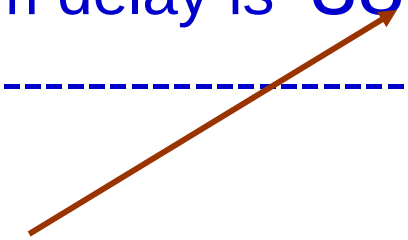
- **REAL time: Routing algorithm run time.**

Static timing after routing

Timing constraint: TS_P2P = MAXDELAY FROM TIMEGRP "PADS" TO TIMEGRP "PADS" 99.999 uS ;

20135312 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)

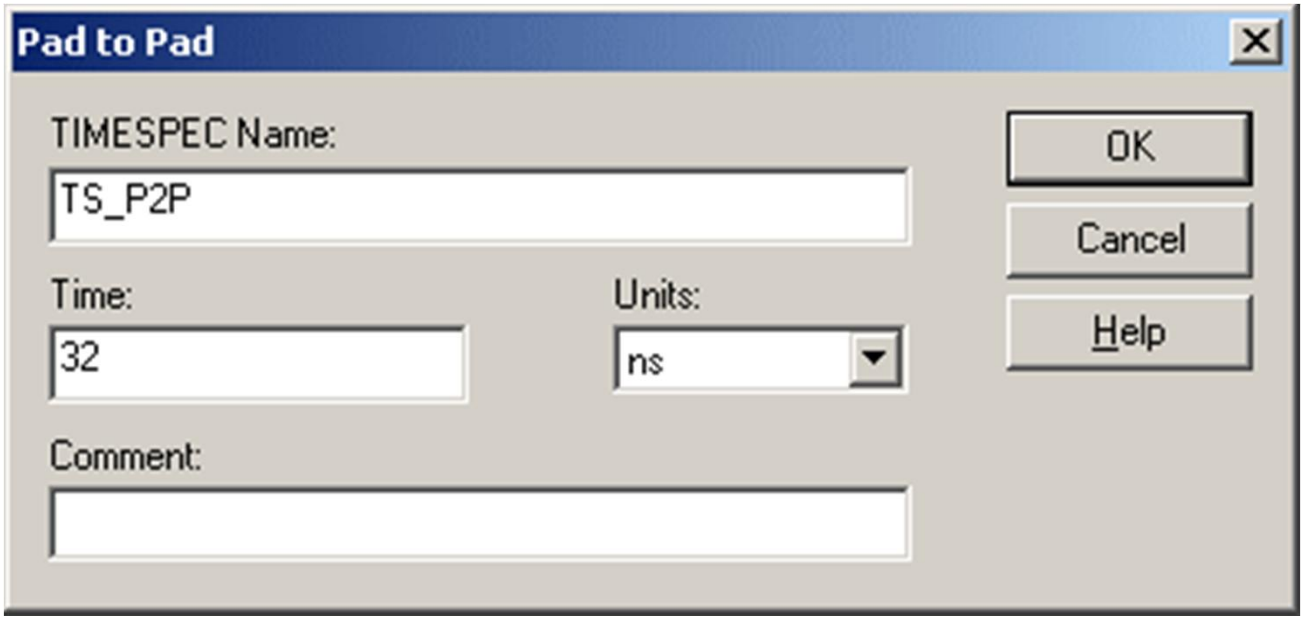
Maximum delay is 38.424ns.



- **(vs. 20.916 ns in mapping report) Because of interconnect delays.**

Timing constraint

- Use timing constraint editor:



Post-routing static timing report

Timing constraint: TS_P2P = MAXDELAY FROM

TIMEGRP "PADS" TO TIMEGRP "PADS" 32 nS ;

20135312 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)

Maximum delay is 31.984ns.

Tools generally try to meet the delay goal as closely as possible to minimize area.

Tighter timing constraints

- Tighten requirement to 25 ns.
- Post-place-route timing report:

Timing constraint: TS_P2P = MAXDELAY FROM TIMEGRP "PADS" TO TIMEGRP "PADS" 25 nS ;

20135312 items analyzed, 11 timing errors detected. (11 setup errors, 0 hold errors)

Maximum delay is 31.128ns.

Report on a violated path

Slack: -6.128ns (requirement - data path)
Source: y<0> (PAD)
Destination: p<30> (PAD)
Requirement: 25.000ns
Data Path Delay: 31.128ns (Levels of Logic = 31)

Modify the logic and/or physical design to improve the delay.

Power report

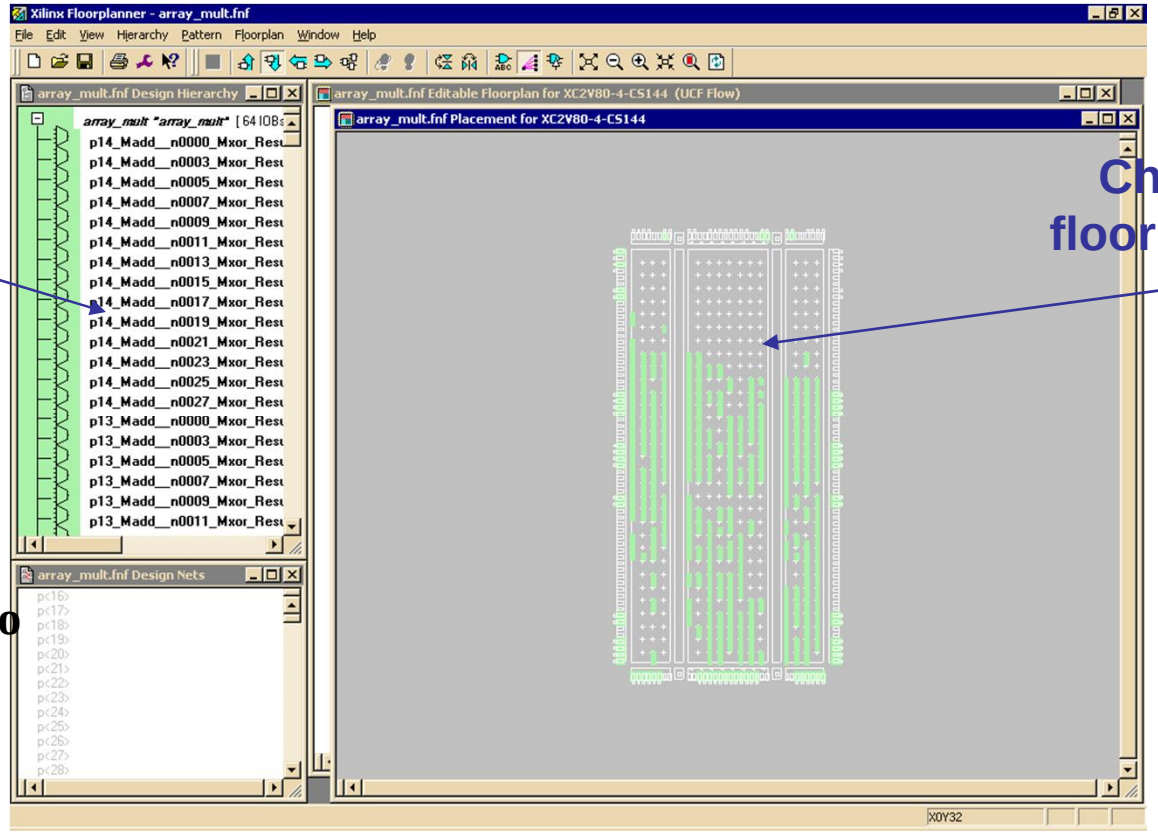
```
Power summary:                  I(mA)    P(mW)
------------------------------------------------------------------
Total estimated power consumption:              333
                    ---
              Vccint 1.50V:       0        0
              Vccaux 3.30V:     100      330
              Vcco33 3.30V:       1        3
                    ---
                Inputs:         0        0
                 Logic:         0        0
               Outputs:
                 Vcco33         0        0
               Signals:         0        0
                    ---
      Quiescent Vccaux  3.30V:     100      330
      Quiescent Vcco33  3.30V:       1        3

Thermal summary:
------------------------------------------------------------------
  Estimated junction temperature:               36C
            Ambient temp:  25C
              Case temp:  35C
              Theta J-A:  34C/W
```

Helps us determine whether we need additional cooling.

Improving area

- Floorplanner window:
  - Floorplanner **à** View/edit placed design



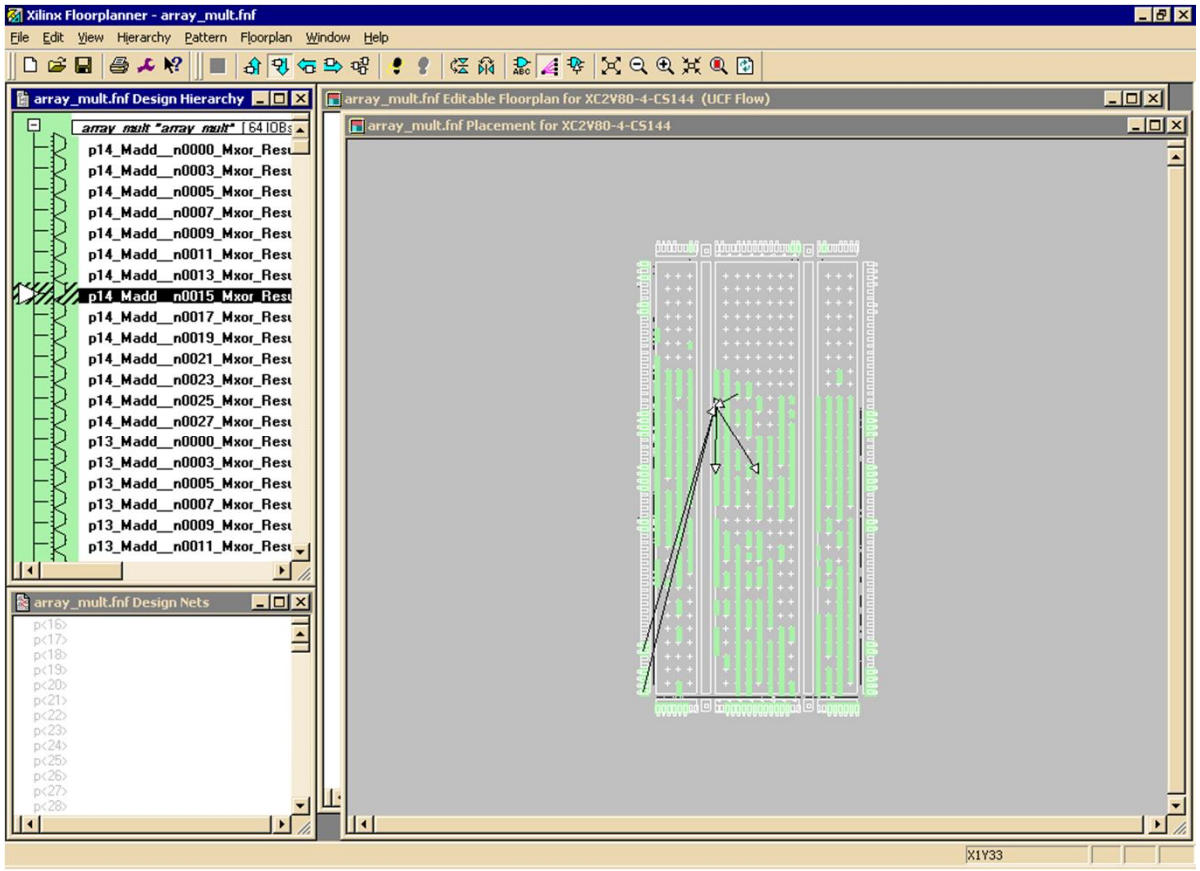
LEs

Chip floorplan

- **Green rectangles: mapped components to CLBs**

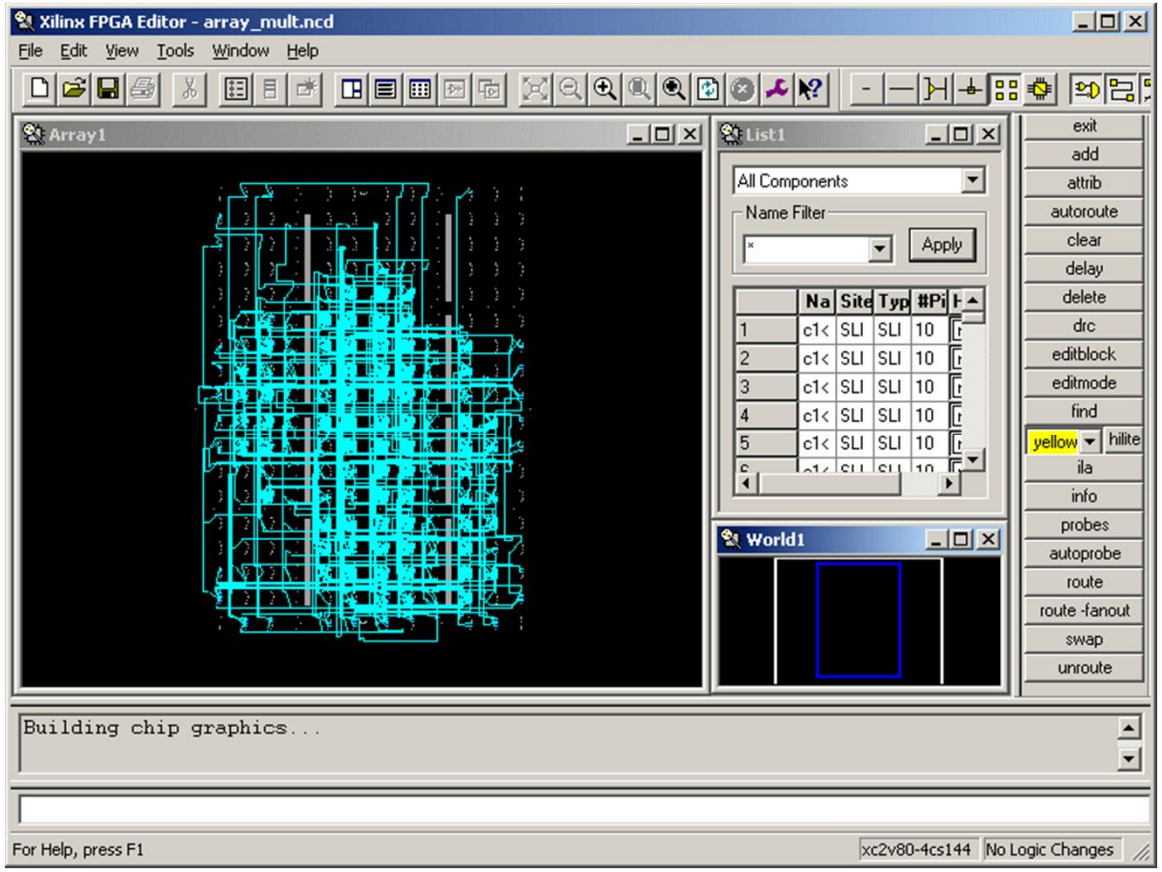
Rat's nest wiring

- **If you click on a component in the deign hierarchy window, its rat's nest is shown.**



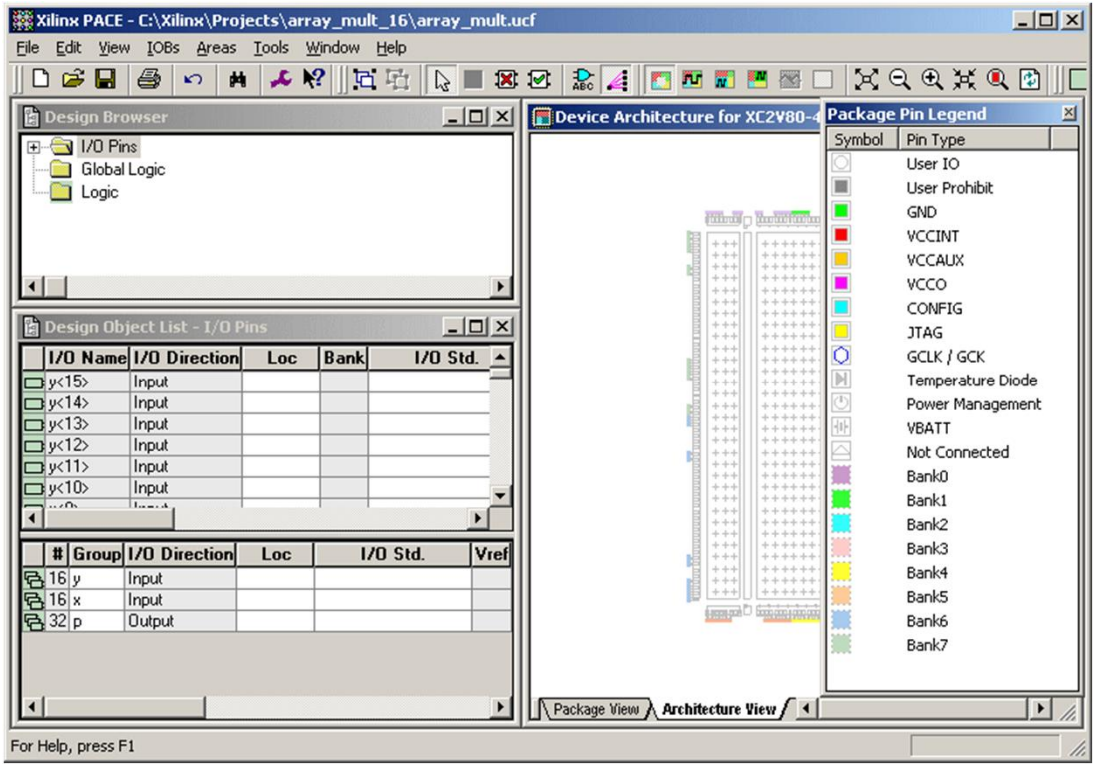
Routing editor view

- **FPGA Editor à View/Edit Routed Design**

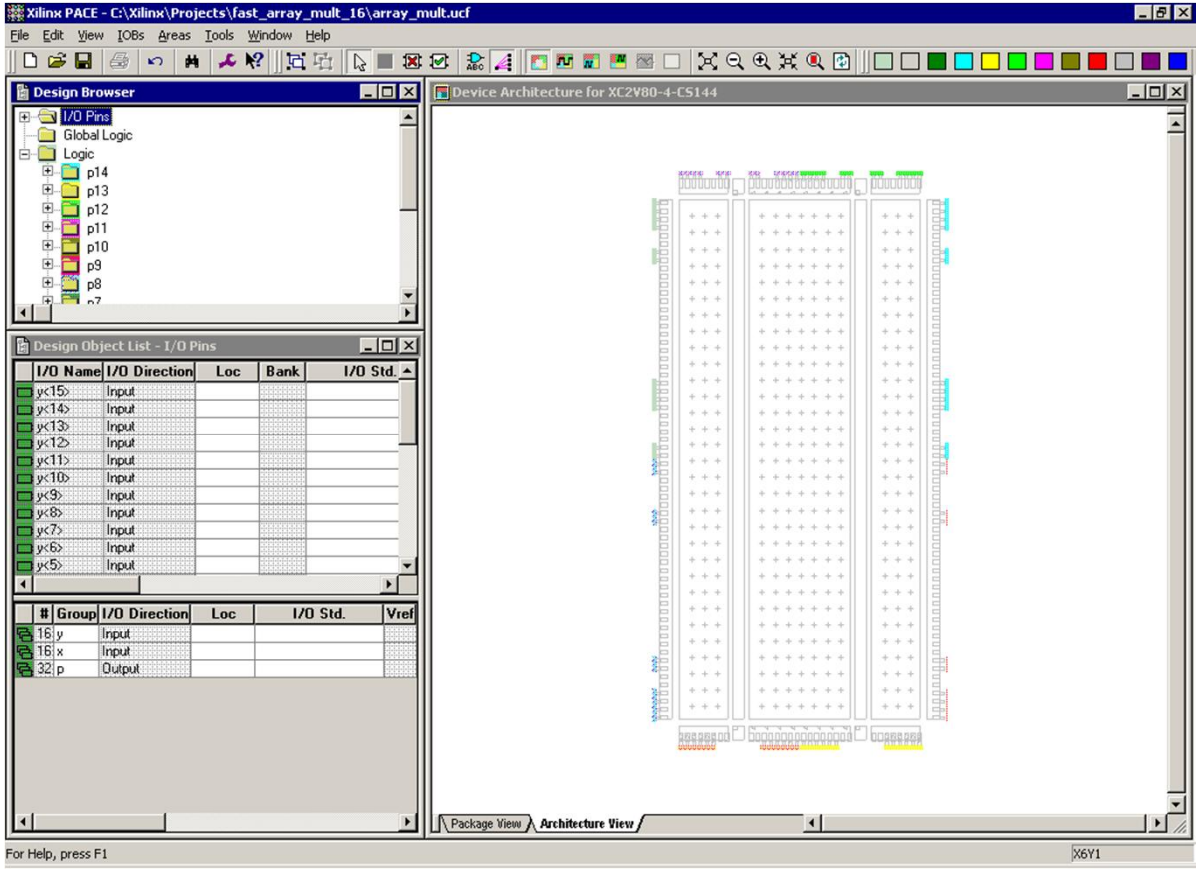


Editing constraints

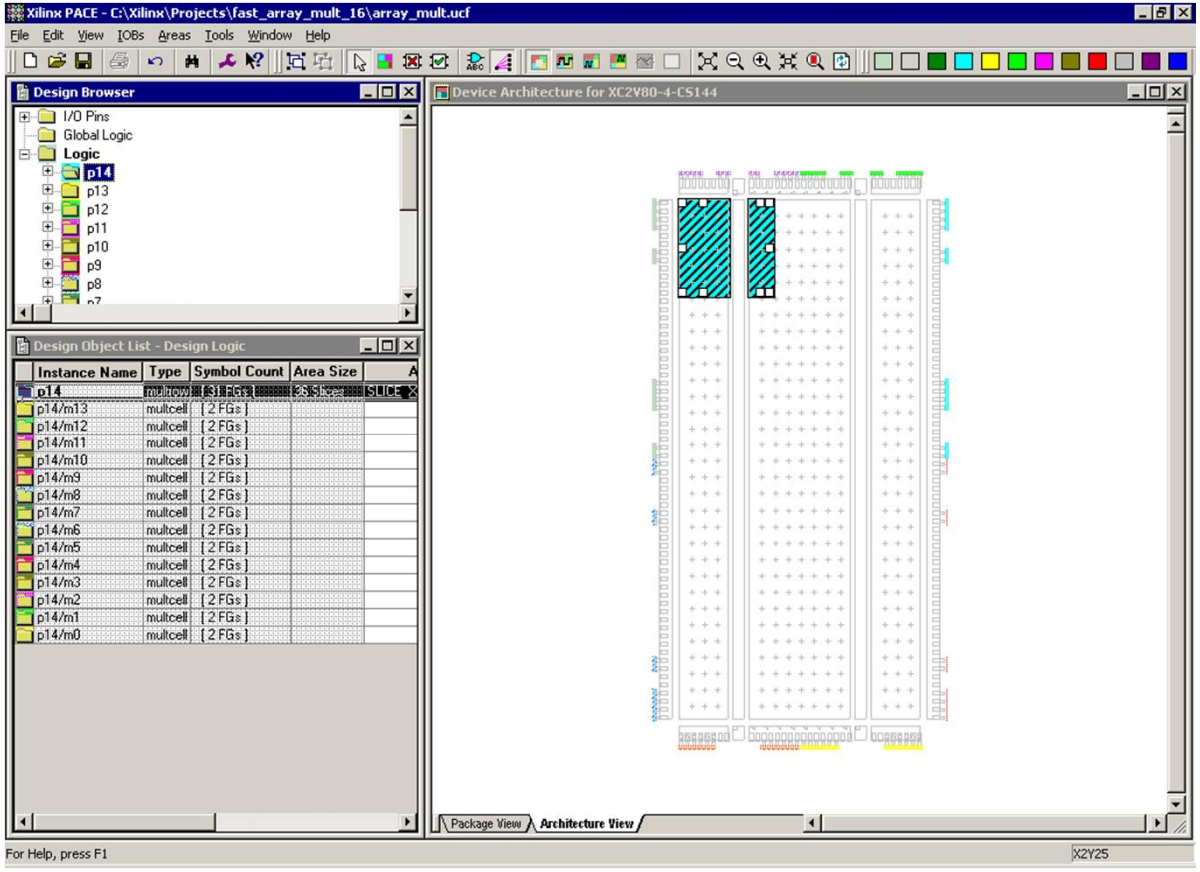
- Use constraints editor to place constraints:
  - This tool allws you to constrain
    1. placement of logic
    2. assignment of chip I/Os to IOBs (e.g useful for PCB design)



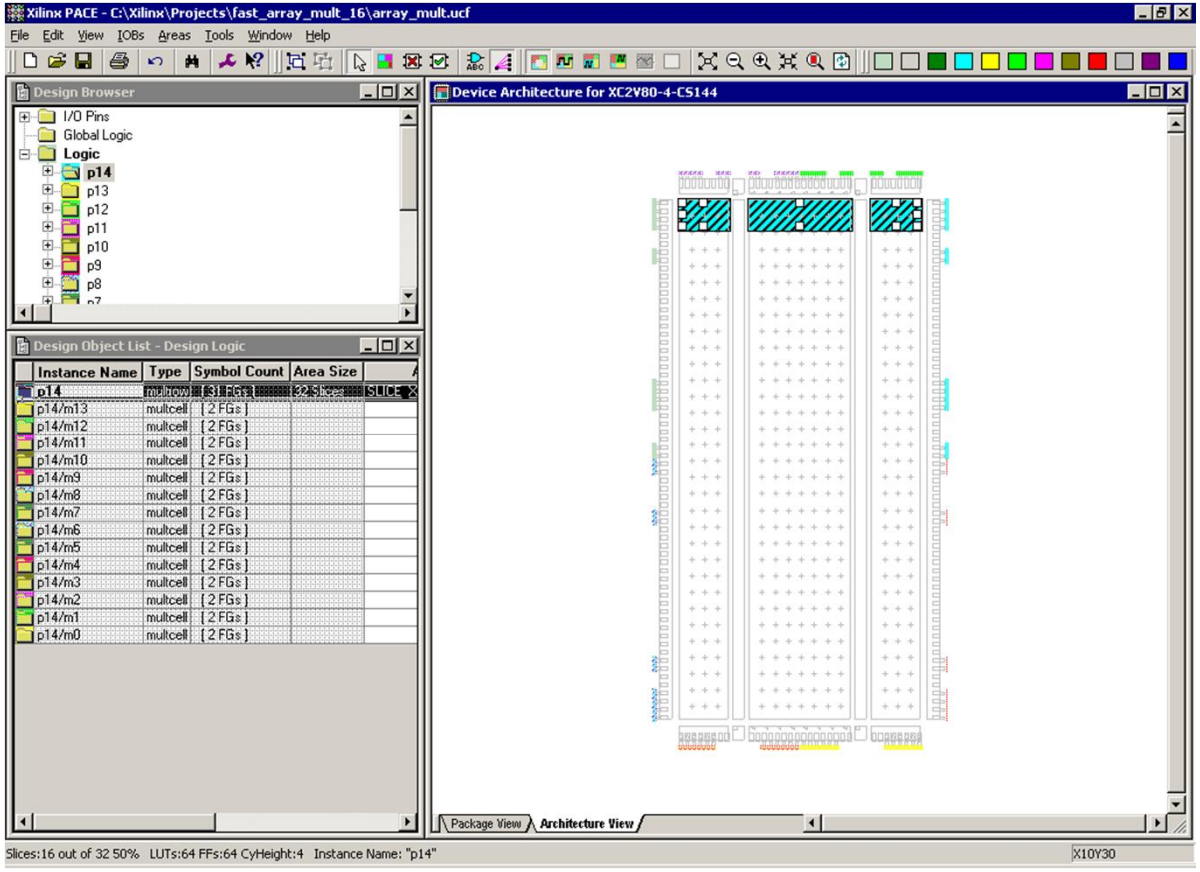
Design browser pane



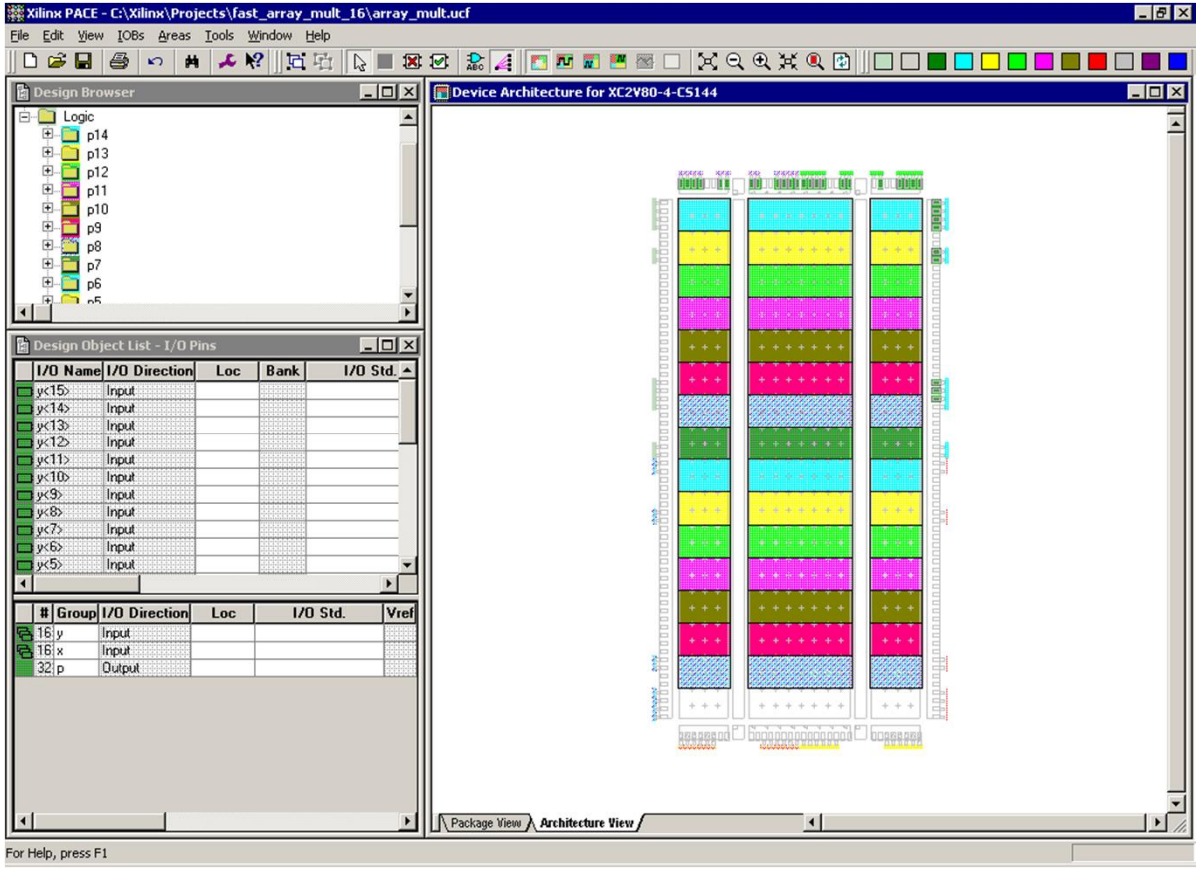
Drag and drop constraints



Change the shape of constraints

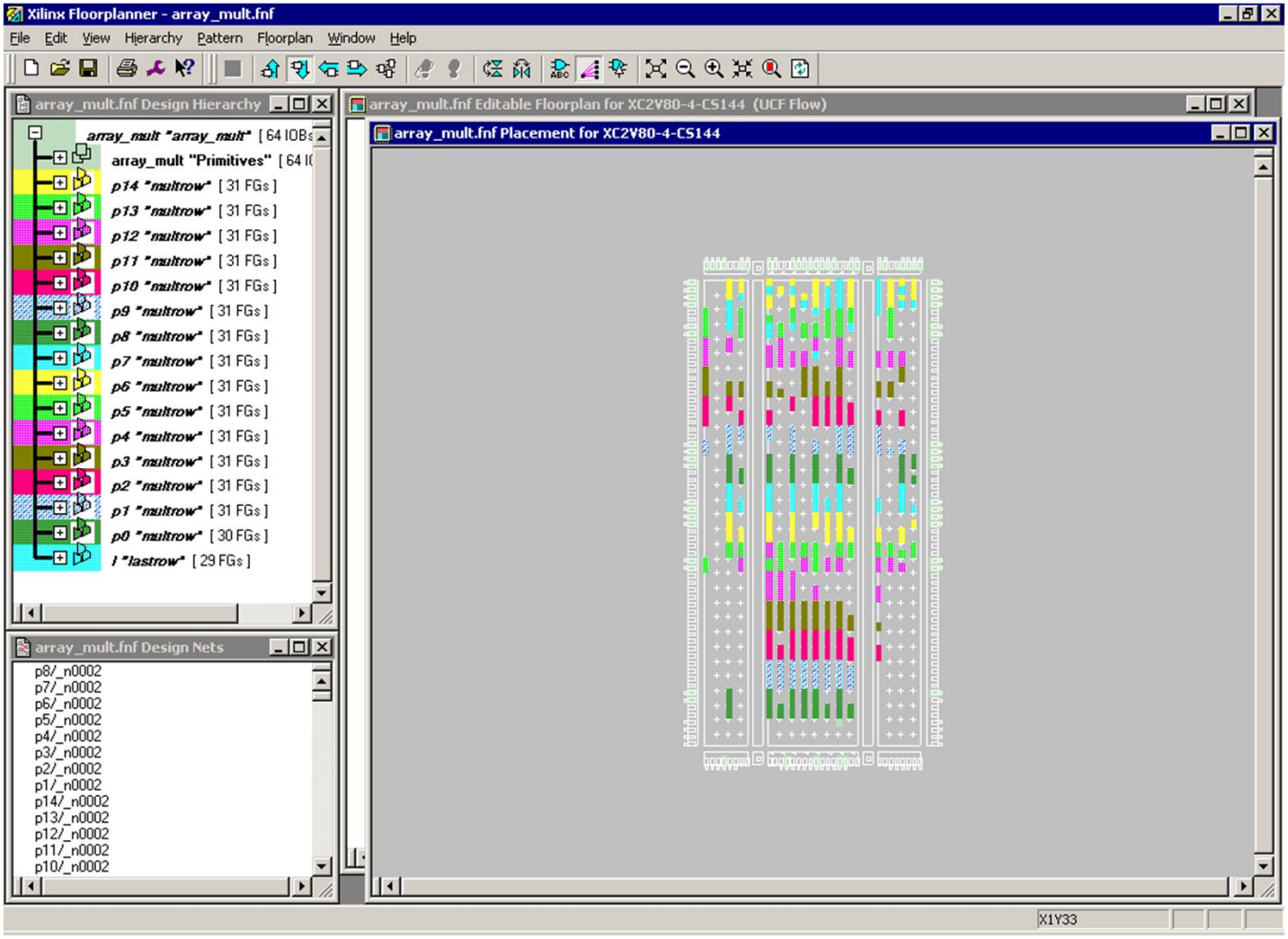


Full set of placement constraints



- **We place the rows of the multiplier one below the other to create the row structure of the floorplan.**

Placement results



New timing report

- After placement constraints:

  19742142 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)

  Maximum delay is 29.934ns.

- Compares to 31.128 ns for unconstrained placement.

Power Optimization

Power optimization

- Transitions cause power consumption.
- Logic network design helps control power consumption:
  - minimizing capacitance;
  - eliminating unnecessary glitches.

$$P_{dyn} = \sum a.f.C_L.V_{dd}^2$$

Power estimation tools

- Power estimator approximates power consumption from:
  - Netlist;
  - Primary input transition probabilities;
  - Capacitive loading.
- Two Types:
  - Switch/logic simulation-based
  - Statistical models

Power optimization

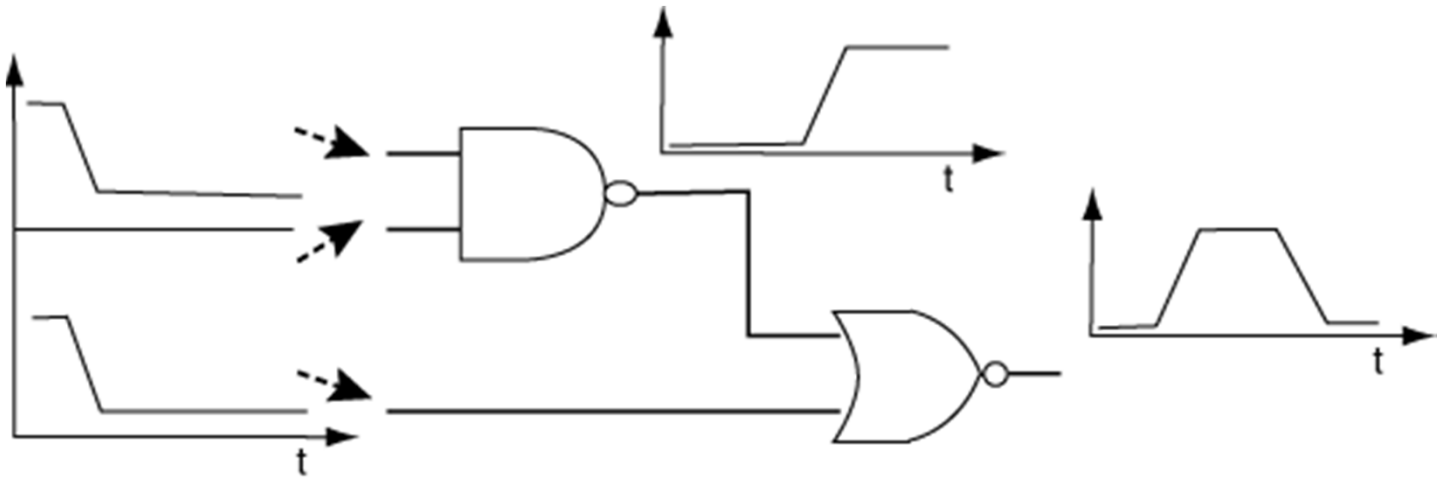
- Leakage in more advanced processes.
  - Even when logic is idle.
  - Solution:
    - Reducing junction temperature by using fans, heat sinks, or design modifications
    - Reducing voltage levels of the IOs.
      - Each bank requires a separate power supply.

Power optimization

- Maximizing the number of unused and unpowered banks.
- Clock gating:
  - Clocks that drive logic, embedded memory, SerDes, and other FPGA primitives can be disabled when their operation is not required.

Glitching example

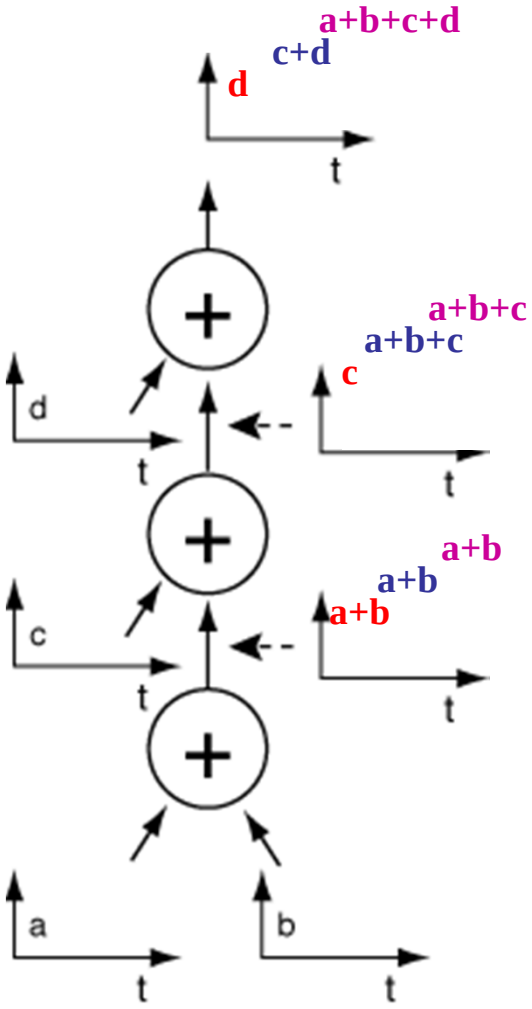
- Gate network:



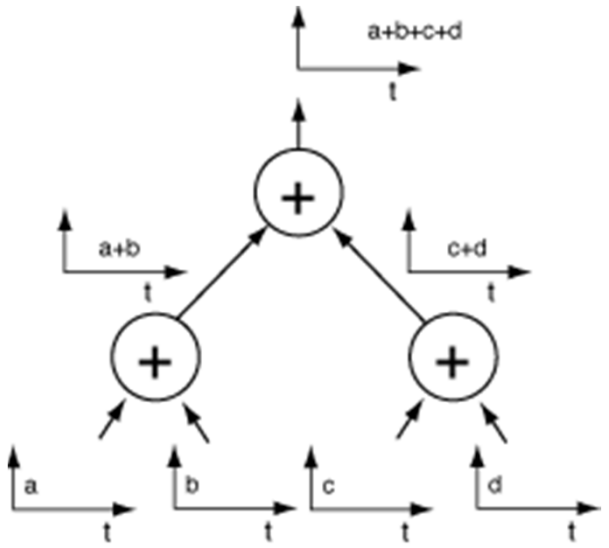
Glitching example behavior

- NOR gate produces 0 output at beginning and end:
  - beginning: bottom input is 1;
  - end: NAND output is 1;
- Difference in delay between application of primary inputs and generation of new NAND output causes glitch.

Adder Chain Glitching



bad



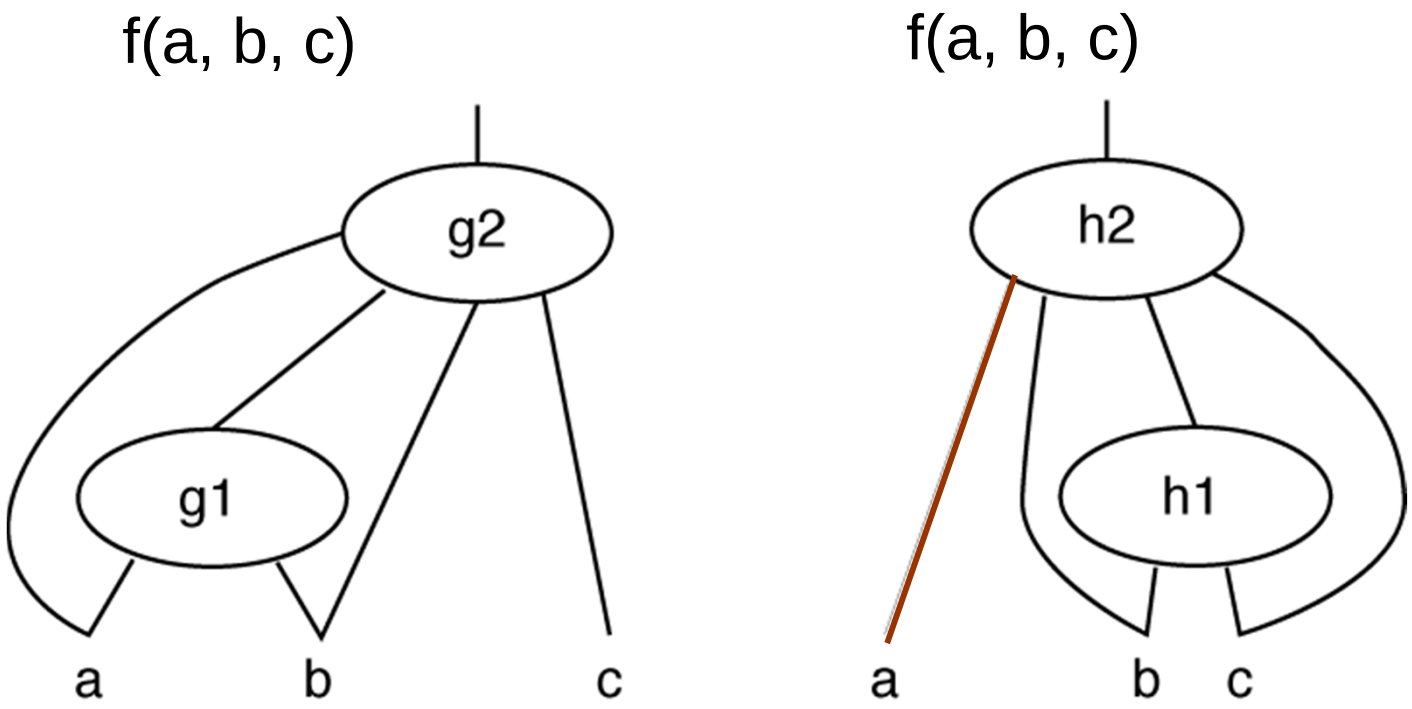
good

Explanation

- Unbalanced chain has signals arriving at different times at each adder.
  - A glitch downstream propagates all the way upstream.
- Balanced tree introduces multiple glitches simultaneously, reducing total glitch activity.

Factorization for low power

- Proper factorization reduces glitching.
- **'*a*' has high transition probability, '*b*' and '*c*' low probabilities**



Factorization techniques

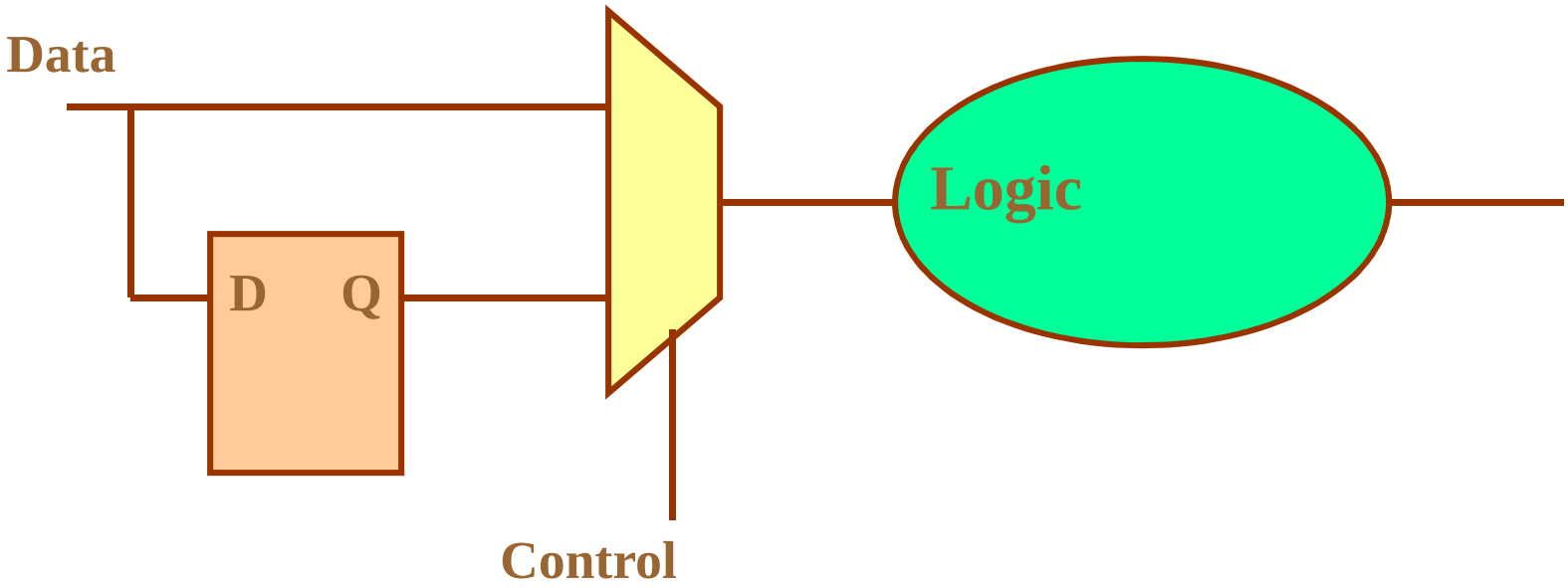
- Reduce number of logic levels through which high-probability signals must travel
  - **à** Reduce propagation of glitches.

Example (ALU)

- ALU output is not used for every cycle
  - **à** If ALU inputs change, the energy is needlessly consumed

Example (ALU)

- Control Signal selects whether data is allowed to pass the logic or the previous value is held to avoid transitions.



Low Power Design Techniques

- Reduce wiring capacitance
  - Get post-layout feedback for better capacitance estimations
- Place and route to minimize capacitance of nodes with high glitching activity.
- Reducing FPGA operating voltages.
- Reducing operating frequencies.
- Reducing the overall toggle rate of the design.

State assignment

- One-hot:
  - usually a good choice for optimizing speed or reducing power.
- Gray:
  - appropriate for controllers exhibiting long paths without branching.
  - Minimizes hazards and glitches.
  - Gives good results when implementing the State Register with T Flip-Flops.
  - Can be used to minimize power
- Compact:
  - Minimizes the number of bits in the state register.
  - Minimizes next state equations.

Sample Report

```
=========================================
*  HDL Synthesis
=========================================
Synthesizing Unit <state_machines>
    Found 4-bit register for signal <state_outputs>.
    Found 3-bit register for signal <state_cur>.
    Found finite state machine <FSM_0> for signal <state_cur>.
-----------------------------------------
| States             | 4
| Transitions        | 8
| Inputs             | 4
| Outputs            | 4
| Clock              | clk (rising_edge)
| Reset              | reset (positive)
| Reset type         | synchronous
| Reset State        | 000
| Encoding           | auto
| Implementation     | LUT
-----------------------------------------
Summary:
inferred   4 D-type flip-flop(s).
inferred   1 Finite State Machine(s).
```

```
HDL Synthesis Report
Macro Statistics
# Registers                          : 1
 4-bit register                      : 1
# FSMs                  : 1
=====================================
Low Level Synthesis
=====================================
Analyzing FSM <MFsm> for best encoding.
Optimizing FSM <state_cur> on signal <state_cur[1:2]> with gray encoding.
------------------
 State | Encoding
------------------
 000   | 00
 001   | 01
 010   | 11
 011   | 10
------------------
```

- Changed to gray automatically