EE 531: ADVANCED VLSI DESIGN

Introduction

Nishith N. Chakraborty

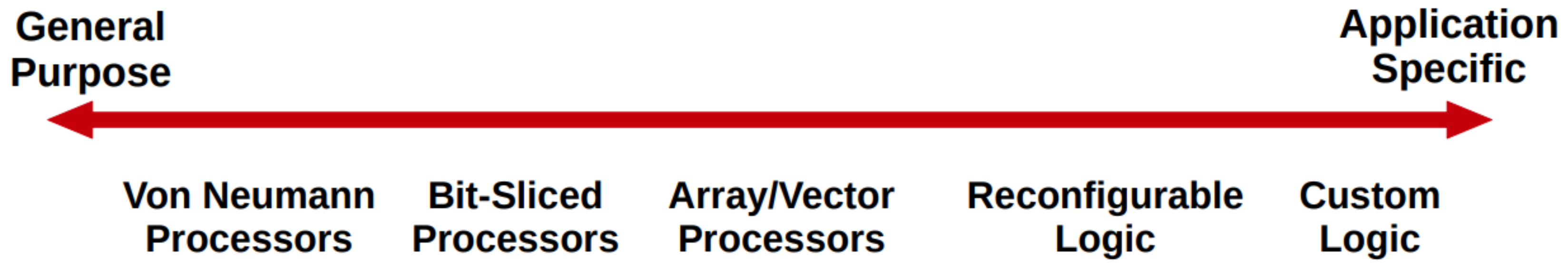
January, 2025

OUTLINE

- Review of IC design basics
- Design partitioning and design approaches
- Introduction to semi-custom design approach
- Introduction to automated approach

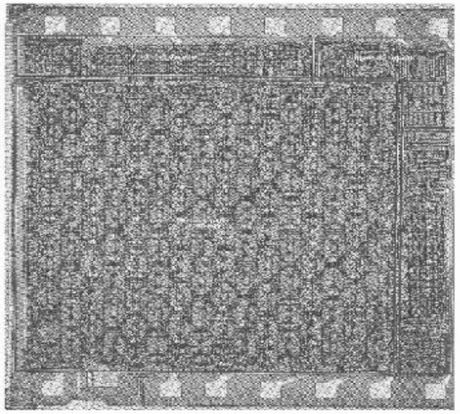
TYPES OF DIGITAL SYSTEMS

- ASIC: Application Specific Integrated Circuits
  - ➢ Usually fabricated in silicon, custom or synthesized
- Microprocessor
- FPGA: Field Programmable Gate Arrays
  - ➢ Reconfigurable logic blocks configured for use

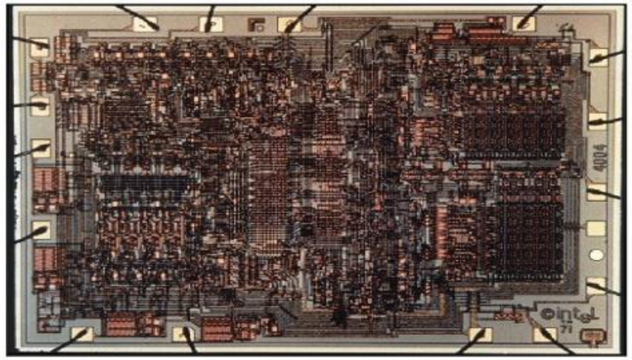


MOS INTEGRATED CIRCUITS

- 1970's processes usually had only nMOS transistors
  - ➢ Inexpensive, but consume power while idle
- 1980s-present: CMOS processes for low idle power



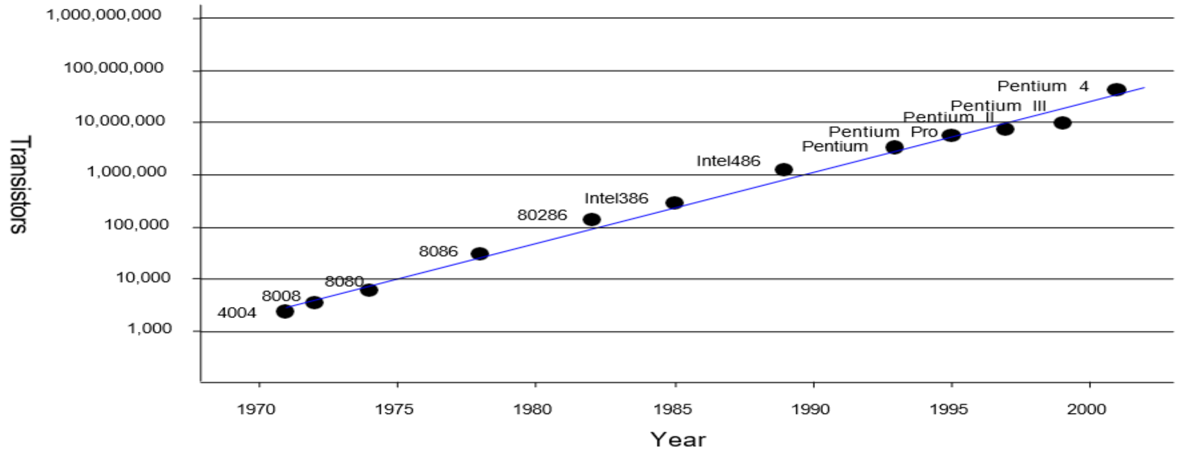
Intel 1101 256-bit SRAM



Intel 4004 4-bit μProc

MOORE’S LAW

- 1965: Gordon Moore plotted transistor on each chip
  - Fit straight line on semilog scale
  - Transistor counts have doubled every 26 months



Integration Levels

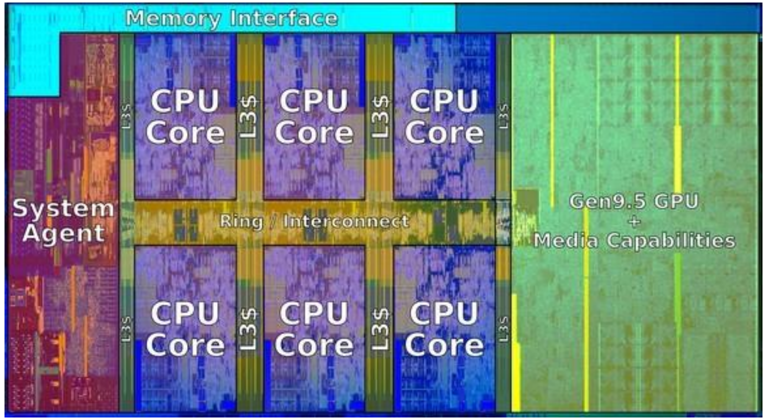
SSI:	10 gates
MSI:	1000 gates
LSI:	10,000 gates
VLSI:	> 10k gates

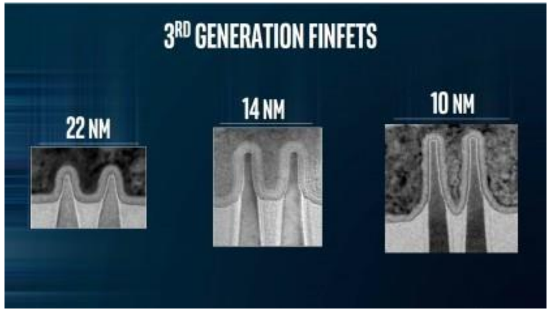
WHY SCALING?

- Technology shrinks by 0.7/generation
- Every generation can integrate 2x more functions per chip; chip cost does not increase significantly
- Cost of a function decreases by 2x
- But…
  - How to design chips with more and more functions?
  - Design engineering population does not double every two years…
- Hence, a need for more efficient design methods
  - Exploit different levels of abstraction

A MICROPROCESSOR EXAMPLE

- Intel Coffee Lake
  - Up to 6 cores (12 threads)
  - 2.9 to 4.8GHz clock (Core i9)
  - Still 14nm technology...
    - Cannon Lake (shrink) is 10nm
    - Moore's Law at an end?
  - Over 1 billion transistors 12MB SmartCache
  - Built-in GPU – a “system on chip”
  - Some custom (“neat” areas); many synthesized components (“messy” areas)



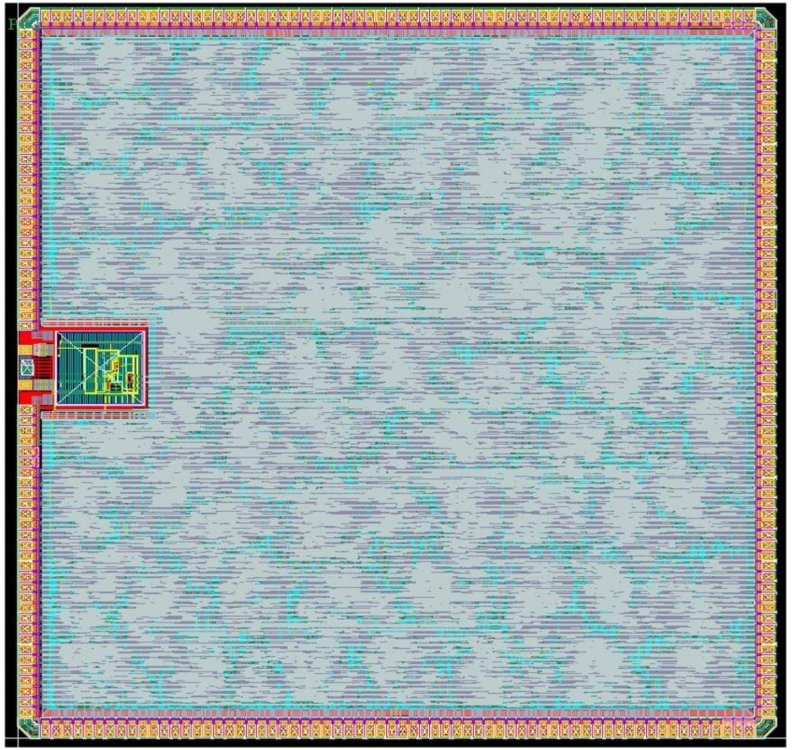


DESIGNING A MICROPROCESSOR

- Requires a large team and a lot of time
- Some components can be synthesized
  - Design written in code (e.g. VHDL)
  - Synthesis tool generates netlist from code
- Other components must be custom designed
  - Blocks on critical path (i.e. critical to system performance)

AN ASIC EXAMPLE: AVALON BITCOIN MINING ASIC

- Application very specific: device that mines bitcoins
- Bitcoin mining can be time consuming – ASIC works fast
- Almost entirely synthesized from standard cells
  - “messy” look of layout
  - Not as fast as full custom



STRUCTURED DESIGN

Hierarchy: Divide and Conquer

- Recursively system into modules

Regularity

- Reuse modules wherever possible
- Ex: Standard cell library

Modularity: well-formed interfaces

- Allows modules to be treated as black boxes

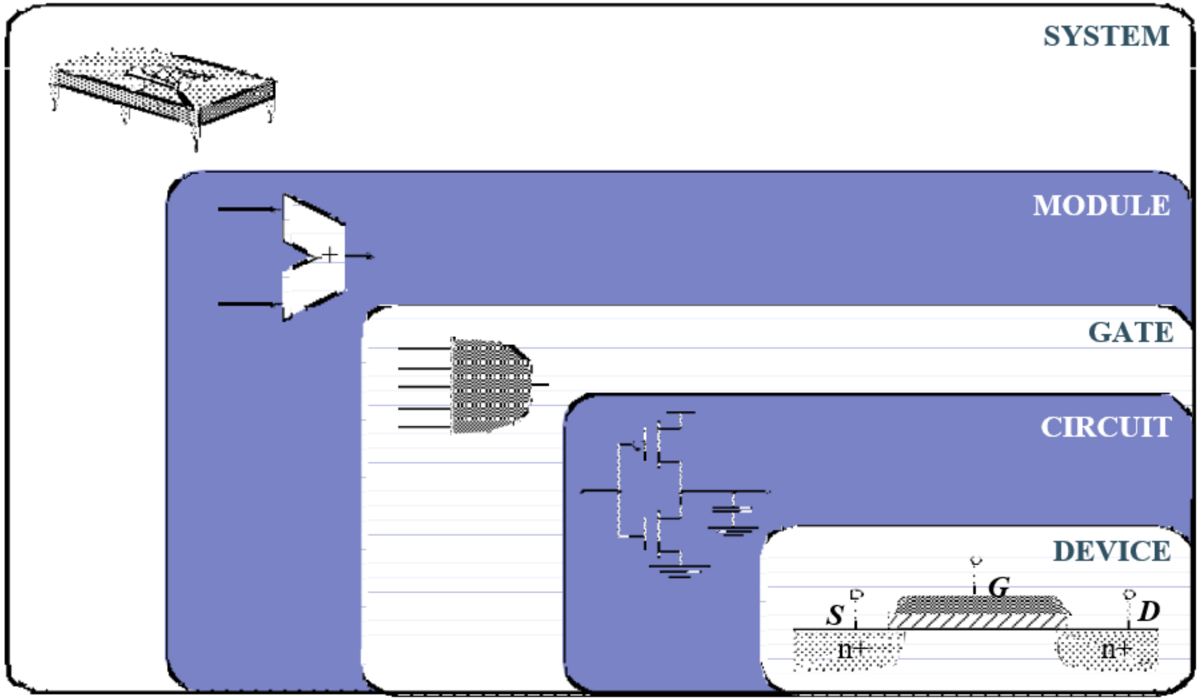
Locality

- Physical and temporal

DESIGN PARTITIONING

- Architecture: User’s perspective, what does it do?
  - Instruction set, registers
  - MIPS, x86, Alpha, PIC, ARM, …
- Microarchitecture
  - Single cycle, multi-cycle, pipelined ?
- Logic: how are functional blocks constructed
  - Ripple carry, carry lookahead, carry select adders
- Circuit: how are transistors used
  - Complementary CMOS, pass transistors, domino
- Physical: chip layout
  - Datapaths, memories, random logic

DESIGN ABSTRACTION LEVELS



PERFORMANCE METRICS

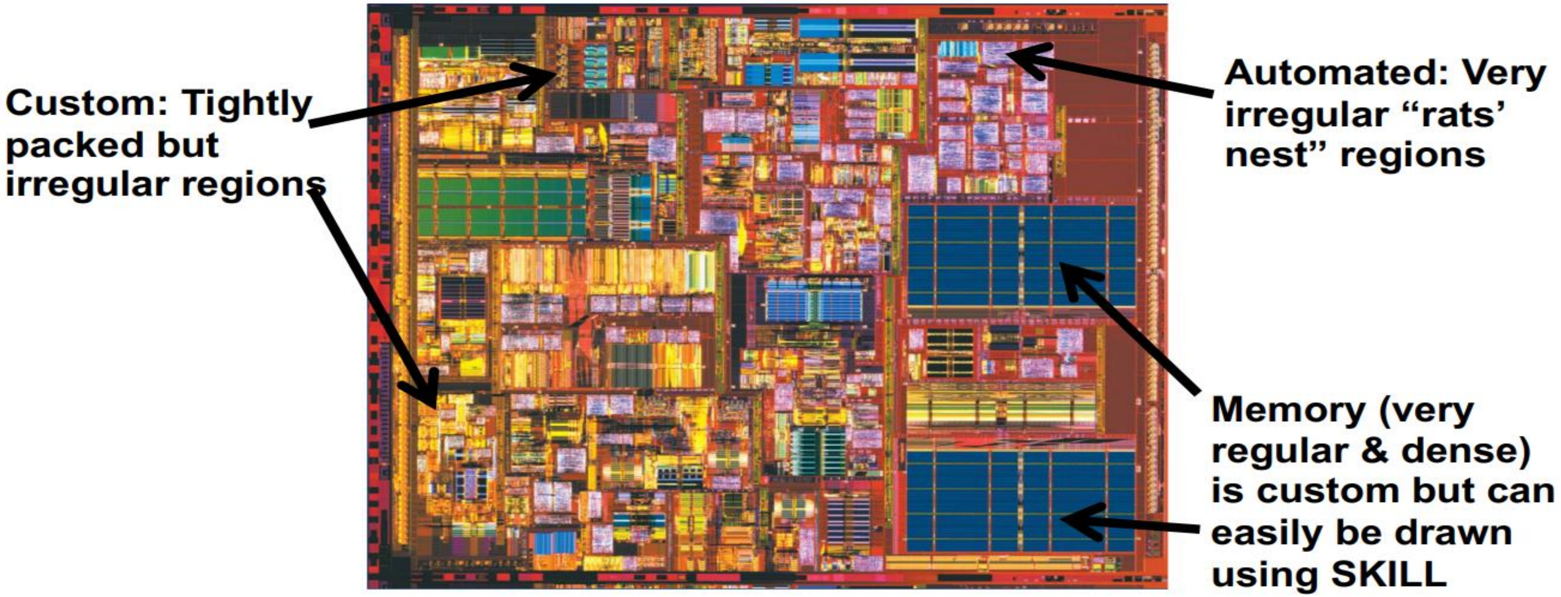
How do we evaluate performance of a digital circuit (gate, block, …)?

- Cost
- Reliability
- Scalability
- Speed (delay, operating frequency)
- Power dissipation
- Energy to perform a function

VLSI DESIGN APPROACHES

- **Custom:** A design that has been carefully crafted by the designer including manual layout.
- **Semi-custom:** Some custom crafted blocks including custom blocks using automation (i.e., SKILL) along with some fully automated blocks. Usually makes use of predefined, well optimized blocks that cannot be changed.
- **Automated:** The use of a hardware description language (VHDL, Verilog, SystemC, etc.) and CAD tools to synthesize a design from code to Silicon.

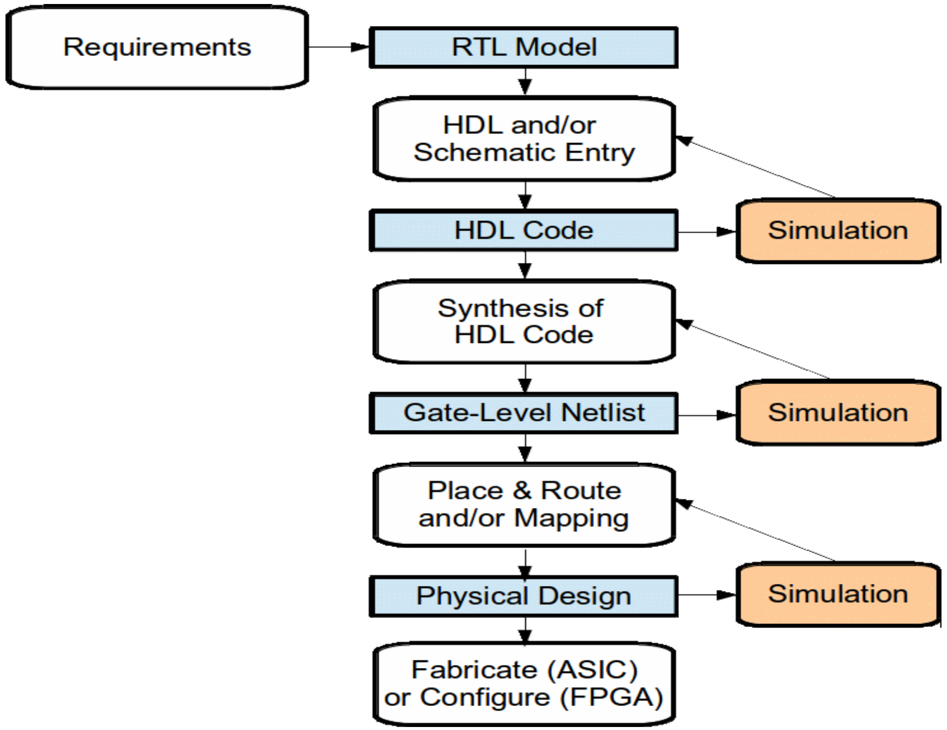
A QUICK LOOK AT INTEL PENTIUM


Custom: Tightly packed but irregular regions
Automated: Very irregular "rats' nest" regions
Memory (very regular & dense) is custom but can easily be drawn using SKILL

ASIC/SOC DESIGN FLOW

- SoC design should always start with a clear design flow
- An integrated circuit design flow shows the particular steps taken for each design stage and highlights CAD tools used

ASIC/SOC DESIGN FLOW



AUTOMATED DESIGN

- Typically, an application specific integrated circuit (ASIC) is designed using commercial tools
- A standard cell library is provided by a vendor (i.e., fab facility) which includes a set of basic gates
- The design is specified using an HDL (such as Verilog or VHDL) and synthesized to a gate-level netlist in terms of the gates in the library
- Finally, this netlist is used for floorplanning and place & route using another tool to produce layout
- Custom blocks can be included but much of the flow is automated starting with the HDL

STANDARD CELL LIBRARY

- The standard cell library is usually provided a vendor
- A lot is required for a library:
  - Optimized schematic and layout for each gate
  - Timing files
  - Input/output pin definitions
- Typical standard cells: INV, AND2, AND3, OR2, OR3, XOR2, AOI21, AOI22, etc.

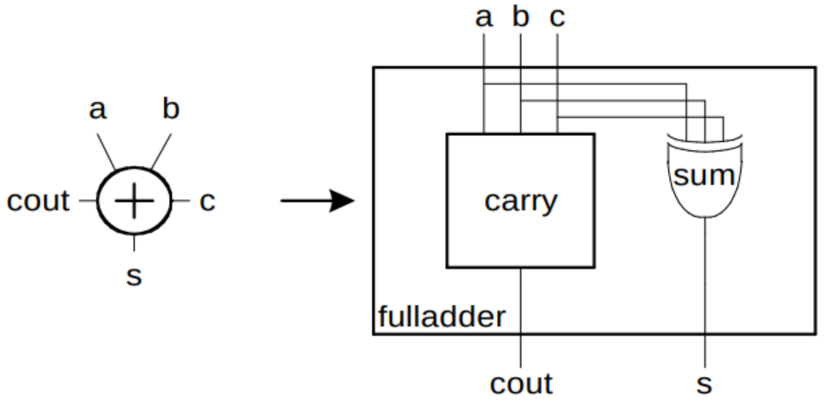
HDLS

- Hardware Description Languages
  - Widely used in logic design
  - Verilog and VHDL
- Describe hardware using code
  - Document logic functions
  - Simulate logic before building
  - Synthesize code into gates and layout
    - Requires a library of standard cells

CAL POLY
SAN LUIS OBISPO

VERILOG EXAMPLE

```
module fulladder(input  a, b, c,
                 output s, cout);
    sum    s1(a, b, c, s);
    carry  c1(a, b, c, cout);
endmodule


module carry(input  a, b, c,
             output cout)
    assign cout = (a&b) | (a&c) | (b&c);
endmodule
```

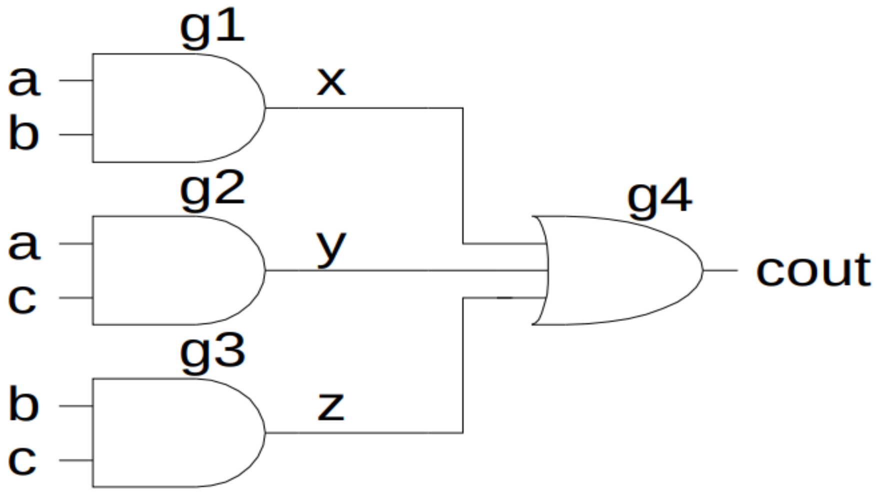


CIRCUIT DESIGN

- How should logic be implemented?
  - NANDs and NORs vs. ANDs and ORs?
  - Fan-in and fan-out?
  - How wide should transistors be?
- These choices affect speed, area, power
- Logic synthesis makes these choices for you
  - Good enough for many applications
  - Hand-crafted circuits are still better

EXAMPLE: CARRY LOGIC

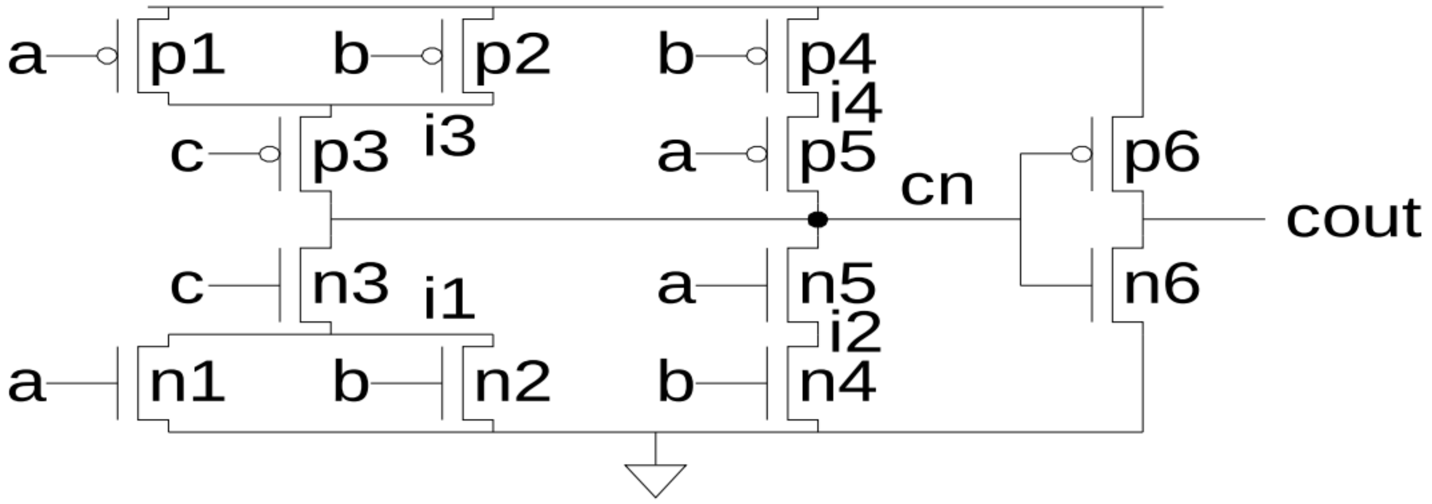
```
assign cout = (a&b) | (a&c) | (b&c);
```



Transistors? Gate Delays?

EXAMPLE: CARRY LOGIC

```
assign cout = (a&b) | (a&c) | (b&c);
```

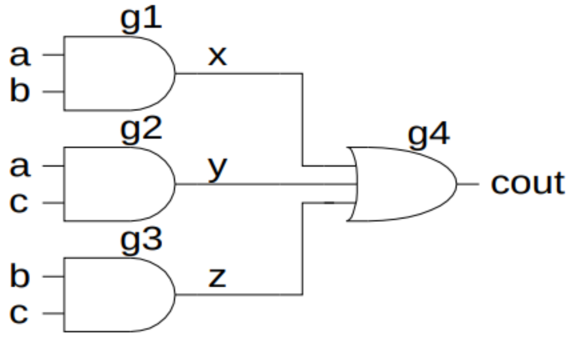


Transistors? Gate Delays?

CAL POLY
SAN LUIS OBISPO

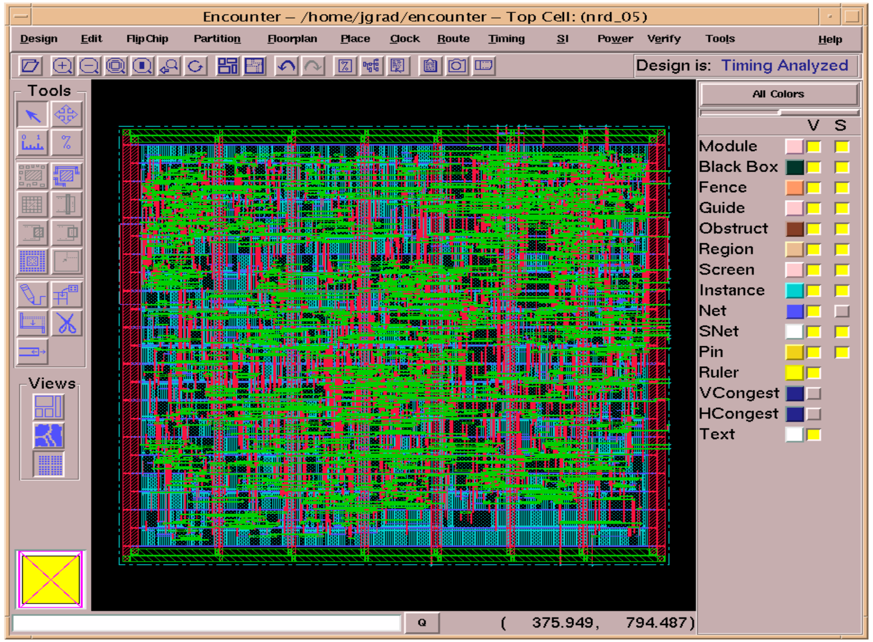
GATE-LEVEL NETLIST

```
module carry(input  a, b, c,
             output cout)

     wire   x, y, z;

     and g1(x, a, b);
     and g2(y, a, c);
     and g3(z, b, c);
     or  g4(cout, x, y, z);
endmodule
```



PLACE & ROUTE

- Floorplanning
- Placement
- Routing


Encounter – /home/jgrad/encounter – Top Cell: (nrd_05)
Design Edit FlipChip Partition Floorplan Place Clock Route Timing SI Power Verify Tools Help
Design is: Timing Analyzed
Tools
Views
All Colors
V S
Module
Black Box
Fence
Guide
Obstruct
Region
Screen
Instance
Net
SNet
Pin
Ruler
VCongest
HCongest
Text
Q
(375.949, 794.487)

SUMMARY

- VLSI design occurs at several abstraction layers
- Can design manually (custom) or using automated CAD tools
- Most VLSI systems are in a “gray area” – some custom and some automation
- This class focuses mostly on top-down, automated approach
- We will study ASIC/SOC design in general and will use FPGAs as our primary platform for prototyping designs

Welcome... Let’s have fun!