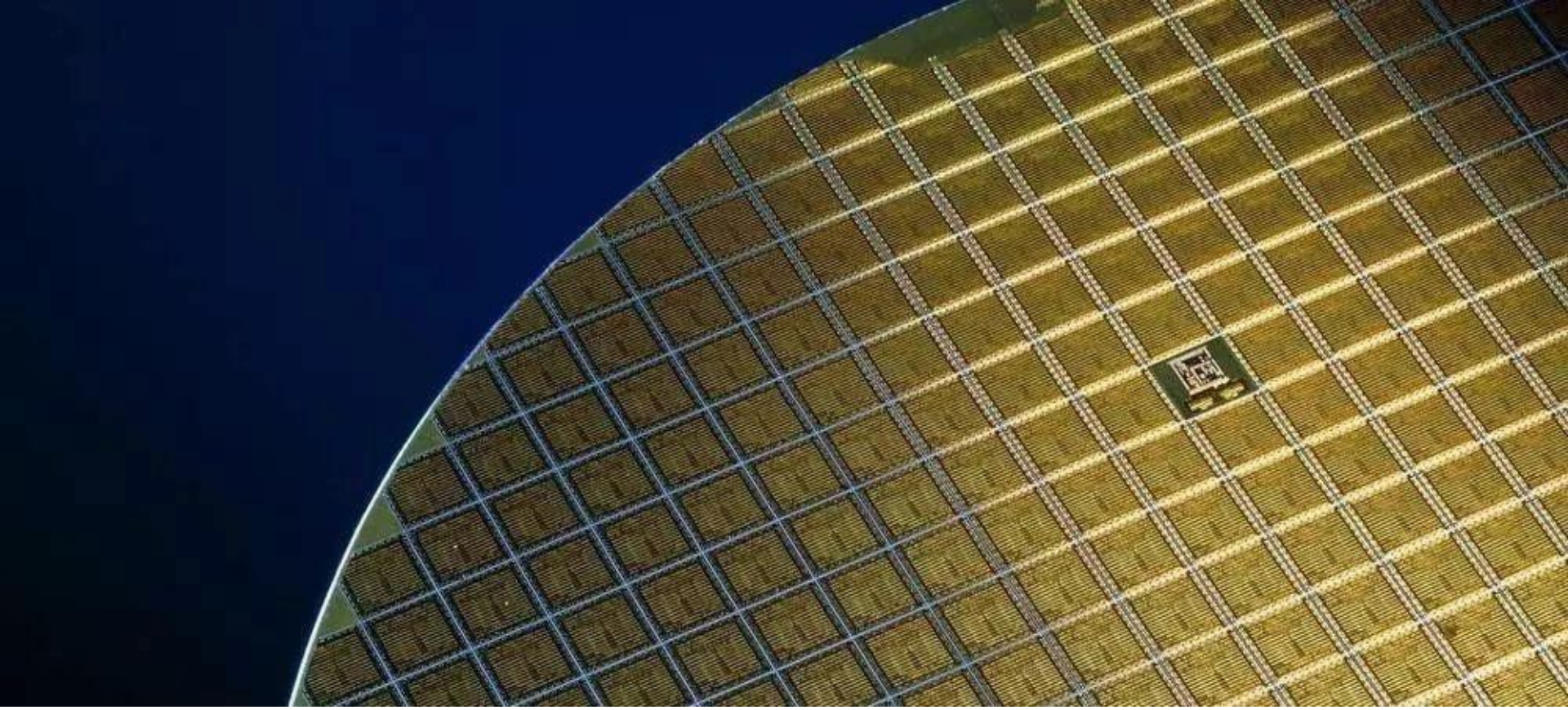


数字集成电路设计
Digital Integrated Circuit Design

付宇卓、孙亚男
上海交通大学电子信息与电气工程学院

What we talk about?

- How to qualify?
- What is the value of the course?
- IC history
- Why Integrated Circuit is the soul of IT?
- Why COST?
- China IC history

Lab assistant

- teacher：***Yuzhuo FU/Yanan SUN***
  - Email:yzfu@sjtu.edu.cn
  - Office Room：205 room Weidianzi Blgs
  - Office hours:1h/week
- Lab assistant: **XXX** Master Candidate
  - Email:**XXX**
  - weidianzi blgs 205 cofferoom
  - Office hours:?

Course Instruction

- Course named：Digital Integrated Circuit
- Course code：SOME-021
- Course type：**fundamental**
- credit：**64** hours/**4** credits
- Pre:digital logic, circuit basic

Teacher

Dr. Yuzhuo FU, Professor

- Phone: 13671916822
- Email: yzfu@sjtu.edu.cn
- Office Room：Rm 417/205, Building of Microelectronics
- Office hours: Thursday(all day),Friday(morning)

Dr. Yanan SUN, Associate Professor

- Phone: (021) 34204546-1041
- Email: sunyanan@sjtu.edu.cn
- Office Room：Rm 418, Building of Microelectronics

iCAT - Intelligent Computer Architecture and Technology

- 团队组成
  - 教授1人，讲师2人
  - 研究生15人左右（博士在读7人）
- 主要研究方向
  - 深度学习/机器学习应用及加速设计
  - 复杂系统故障量化分析及加固技术
- 合作单位
  - Intel、Google、阿里云、航天八院、上汽集团、中通客车等

SJTU-icat.github.io

参课教师介绍

- 孙亚男

副教授，博士

上海交通大学

- 2019/01-至今，上海交通大学微纳电子学系，副教授
- 2015/10-2018/12，上海交通大学微纳电子学系，讲师
- 2009/09-2015/06，香港科技大学电子与计算机工程学系，博士
- 2005/09-2009/07，上海交通大学微电子学院，学士

科研方向

- 基于新型纳米技术的高能效及高良率集成电路设计
- 存内计算电路及三维集成电路设计
- 神经网络加速器及边缘计算

成果简介

在IEEE JSSC、TCAS-I、IEEE TCAS-II、IEEE TVLSI、IEEE TED、IEEE TCAD等集成电路领域高水平期刊和IEEE ISSCC、DAC、DATE 、ISCAS等重要国际会议发表40余篇学术论文

科研项目

主持国家自然科学面上基金、青年自然科学基金、上海市扬帆计划等多项；参与国家重点研发计划等多项。

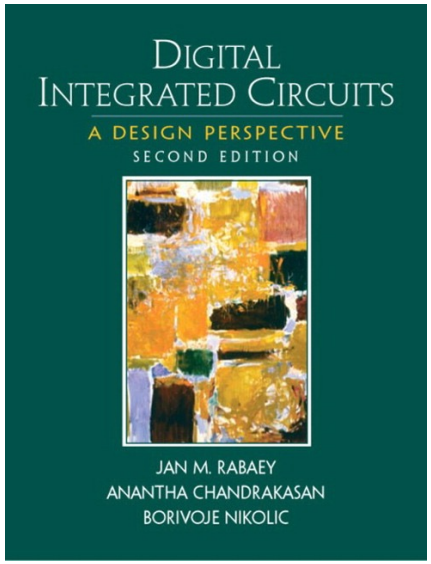
荣誉奖项

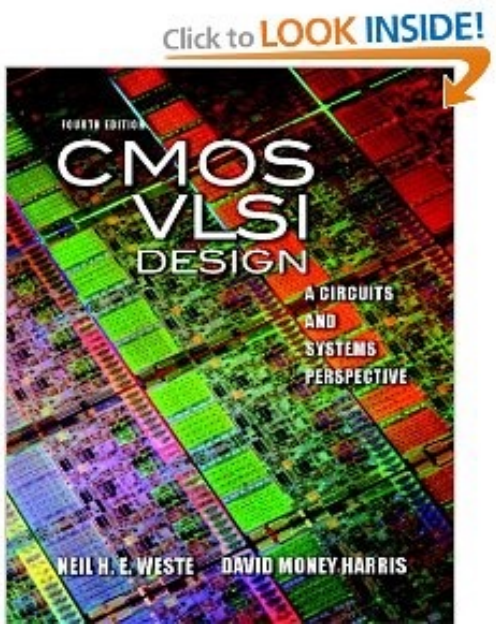
- IEEE欧洲设计自动化与测试会议（DATE）最佳论文提名奖（2020）
- “华为杯”第三届中国研究生创“芯”大赛，优秀指导教师奖（2020）
- 全国大学生集成电路创新创业大赛，优秀指导教师奖（2019）
- 上海市“青年科技英才扬帆计划“（2017）
- 北京大学生集成电路设计大赛暨全国邀请赛，优秀指导教师奖（2016）
- IEEE微电子国际会议（ICM）最佳论文奖（2014）

Textbook

- ***Digital integrated circuit*** Jan M.Rabaey
- ***CMOS VLSI design- a circuits and systems perspective***
  Meil H.E.Weste, addison wesley press

《CMOS超大规模集成电路设计》





Method of course

- Time, location
  - 1-16 weeks
  - Mon. 3-4[10:00-11:40] ,Room 东下302
  - Wed. 1-2[08:00-09:40], Room 东下302
- Materials
  - Please pre-read the slides before the lessons
- Quiz
  - Finish at the end of each lesson
- Homework
  - Close by the end of the week
- Review 10%, New Contents 90%

Qualification

授课方式 (课时)				作业 (份)	实验 (课时)		考试 (次)	
课堂教学	实验	讲座	习题课	作业	课内	课外	测验	考试
50	2	4	4	30	0	6	30	2

作业	实验报告	考试		
		测验	期中考试	期末考试
0.6_{30}	4_4	0.4_{30}	27	27

Some references

- http://www.itrs.net/
- http://www.lib.sjtu.edu.cn

  IEEE conferences and journals
- **IEEE Journal of Solid-State Circuits (JSSC)**
- **IEEE International Solid-State Circuits Conference(ISSCC)**
- **Symposium on VLSI Circuits (VLSI)**

Classroom Regulations

- No Breakfast
- No Chat

Who is **DIC**?

File Edit View Insert Cell Kernel Widgets Help 可信 Python 3

运行 代码

```
In [2]: a=10
        b=32
        a=a+b

In [4]: a,b
Out[4]: (42, 32)

In [7]: f1=1
        f2=-1
        for n in range(8):
            f1=f1+f2
            f2=f1-f2
            print(f1,f2)

        0 1
        1 0
        1 1
        2 1
        3 2
        5 3
        8 5
        13 8

In [ ]:
```

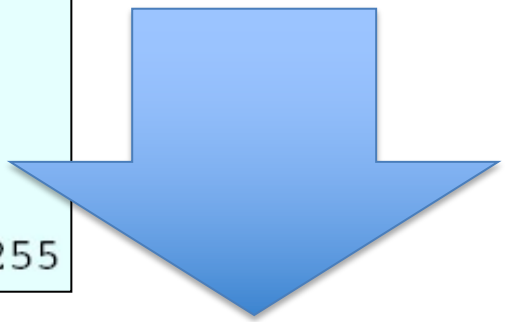
How?

Table 1.7 MIPS instruction set (subset supported)

Instruction	Function		Encoding	op	funct
`add $1, $2, $3`	addition:	$1 → $2 + $3	R	000000	100000
`sub $1, $2, $3`	subtraction:	$1 → $2 – $3	R	000000	100010
`and $1, $2, $3`	bitwise and:	$1 → $2 and $3	R	000000	100100
`or $1, $2, $3`	bitwise or:	$1 → $2 or $3	R	000000	100101
`slt $1, $2, $3`	set less than:	$1 → 1 if $2 < $3 $1 → 0 otherwise	R	000000	101010
`addi $1, $2,`	add immediate:	$1→ $2 + imm	I	001000	n/a
`beq $1, $2, imm`	branch if equal:	PC → PC + imm[a]	I	000100	n/a
`j destination`	jump:	PC_destination[a]	J	000010	n/a
`lb $1, imm($2)`	load byte:	$1 → mem[$2 + imm]	I	100000	n/a
`sb $1, imm($2)`	store byte:	mem[$2 + imm] → $1	I	110000	n/a

```
# fib.asm
# Register usage: $3: n $4: f1 $5: f2
# return value written to address 255
fib:  addi $3, $0, 8      # initialize n=8
      addi $4, $0, 1      # initialize f1 = 1
      addi $5, $0, -1     # initialize f2 = -1
loop: beq $3, $0, end     # Done with loop if n = 0
      add $4, $4, $5      # f1 = f1 + f2
      sub $5, $4, $5      # f2 = f1 - f2
      addi $3, $3, -1     # n = n - 1
      j loop              # repeat until done
end: sb $4, 255($0)       # store result in address 255
```

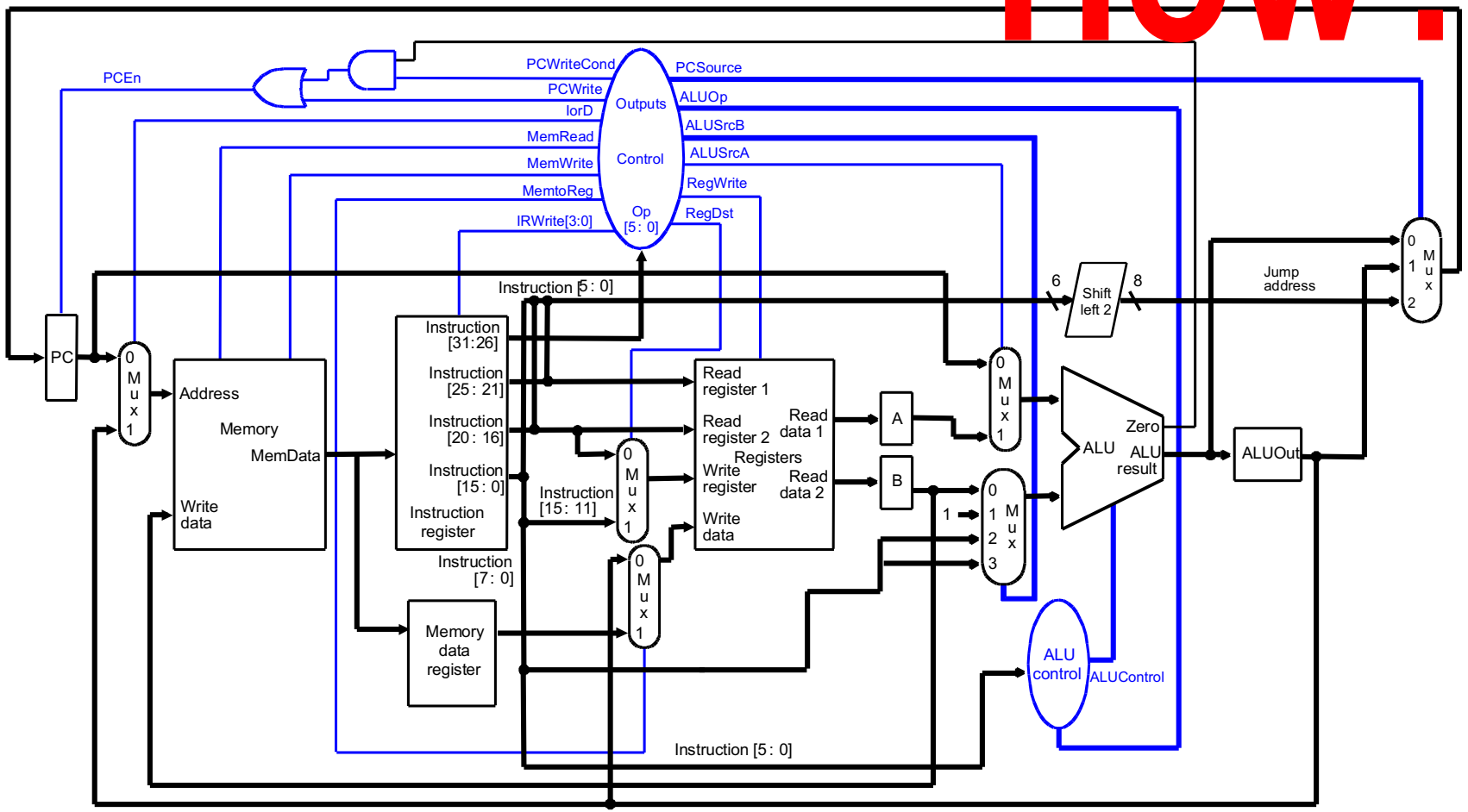
How?



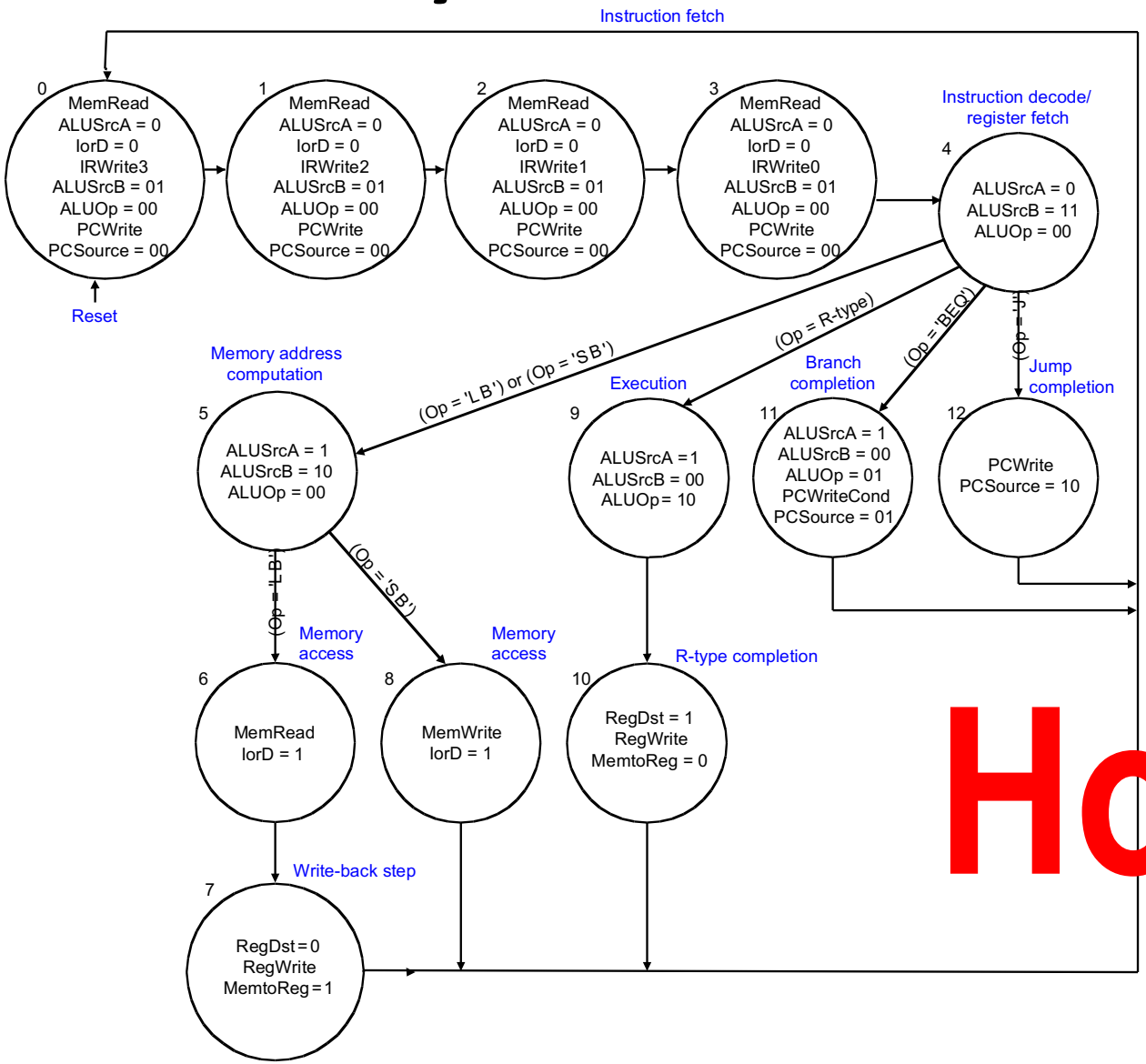
Instruction	Binary Encoding	Hexadecimal Encoding
addi $3, $0, 8	001000 00000 00011 0000000000001000	20030008
addi $4, $0, 1	001000 00000 00100 0000000000000001	20040001
addi $5, $0, -1	001000 00000 00101 1111111111111111	2005ffff
beq $3, $0, end	000100 00011 00000 0000000000000101	10600005
add $4, $4, $5	000000 00100 00101 00100 00000 100000	00852020
sub $5, $4, $5	000000 00100 00101 00101 00000 100010	00852822
addi $3, $3, -1	001000 00011 00011 1111111111111111	2063ffff
j loop	000010 00000000000000000000000011	08000003
sb $4, 255($0)	110000 00000 00100 0000000011111111	a00400ff

MIPS Microarchitecture

How?



Multicycle Controller



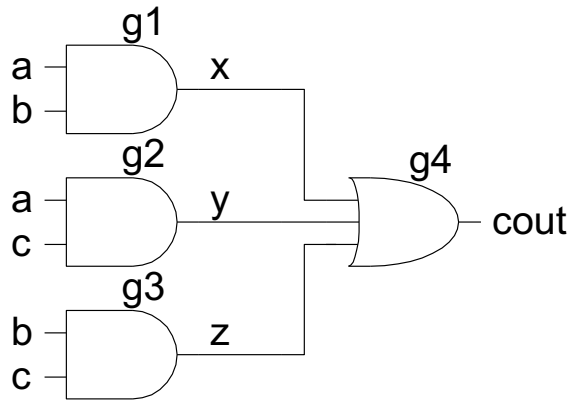
How?

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
```



I have learned

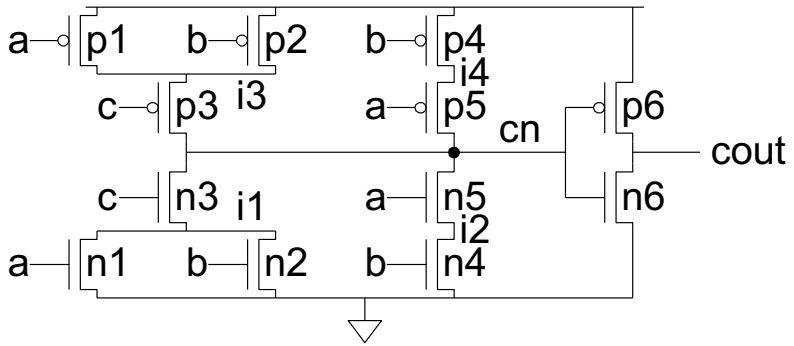
Transistor-Level Netlist

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

      wire     i1, i2, i3, i4, cn;

      tranif1 n1(i1, 0, a);
      tranif1 n2(i1, 0, b);
      tranif1 n3(cn, i1, c);
      tranif1 n4(i2, 0, b);
      tranif1 n5(cn, i2, a);
      tranif0 p1(i3, 1, a);
      tranif0 p2(i3, 1, b);
      tranif0 p3(cn, i3, c);
      tranif0 p4(i4, 1, b);
      tranif0 p5(cn, i4, a);
      tranif1 n6(cout, 0, cn);
      tranif0 p6(cout, 1, cn);
endmodule
```

CMOS?pseudo nMOS?
Pass logic? Dynamic logic?



Speed? Power? Area?

DIC is coming

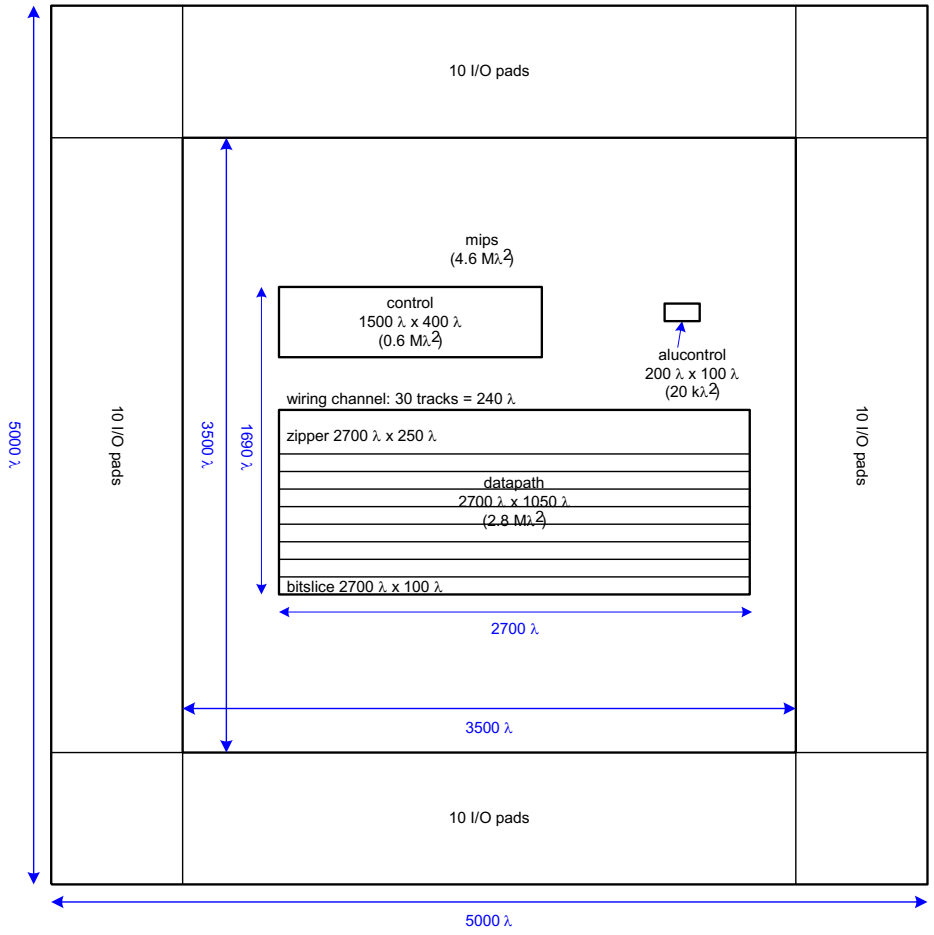
SPICE Netlist

```
.SUBCKT CARRY A B C COUT VDD GND
MN1 I1 A GND GND NMOS W=1U L=0.18U AD=0.3P AS=0.5P
MN2 I1 B GND GND NMOS W=1U L=0.18U AD=0.3P AS=0.5P
MN3 CN C I1 GND NMOS W=1U L=0.18U AD=0.5P AS=0.5P
MN4 I2 B GND GND NMOS W=1U L=0.18U AD=0.15P AS=0.5P
MN5 CN A I2 GND NMOS W=1U L=0.18U AD=0.5P AS=0.15P
MP1 I3 A VDD VDD PMOS W=2U L=0.18U AD=0.6P AS=1 P
MP2 I3 B VDD VDD PMOS W=2U L=0.18U AD=0.6P AS=1P
MP3 CN C I3 VDD PMOS W=2U L=0.18U AD=1P AS=1P
MP4 I4 B VDD VDD PMOS W=2U L=0.18U AD=0.3P AS=1P
MP5 CN A I4 VDD PMOS W=2U L=0.18U AD=1P AS=0.3P
MN6 COUT CN GND GND NMOS W=2U L=0.18U AD=1P AS=1P
MP6 COUT CN VDD VDD PMOS W=4U L=0.18U AD=2P AS=2P
CI1 I1 GND 2FF
CI3 I3 GND 3FF
CA A GND 4FF
CB B GND 4FF
CC C GND 2FF
CCN CN GND 4FF
CCOUT COUT GND 2FF
.ENDS
```

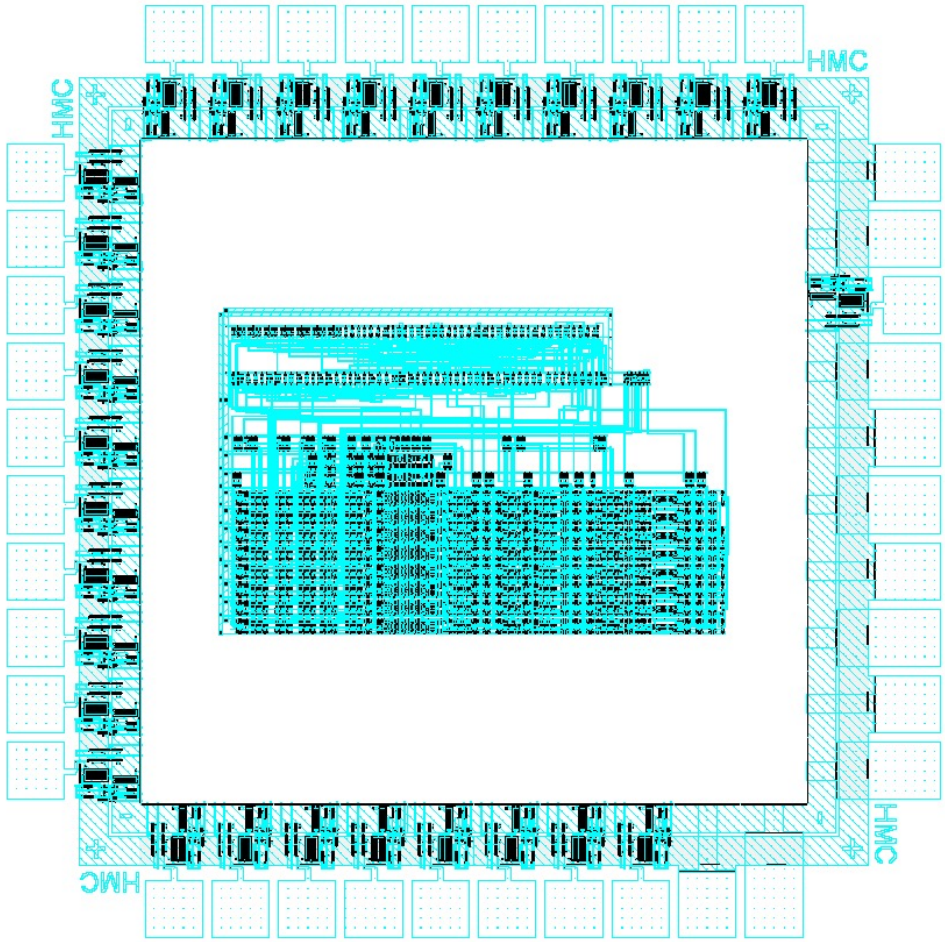
Physical Design

- Floorplan
- Standard cells
  - Place & route
- Datapaths
  - Slice planning
- Area estimation

MIPS Floorplan

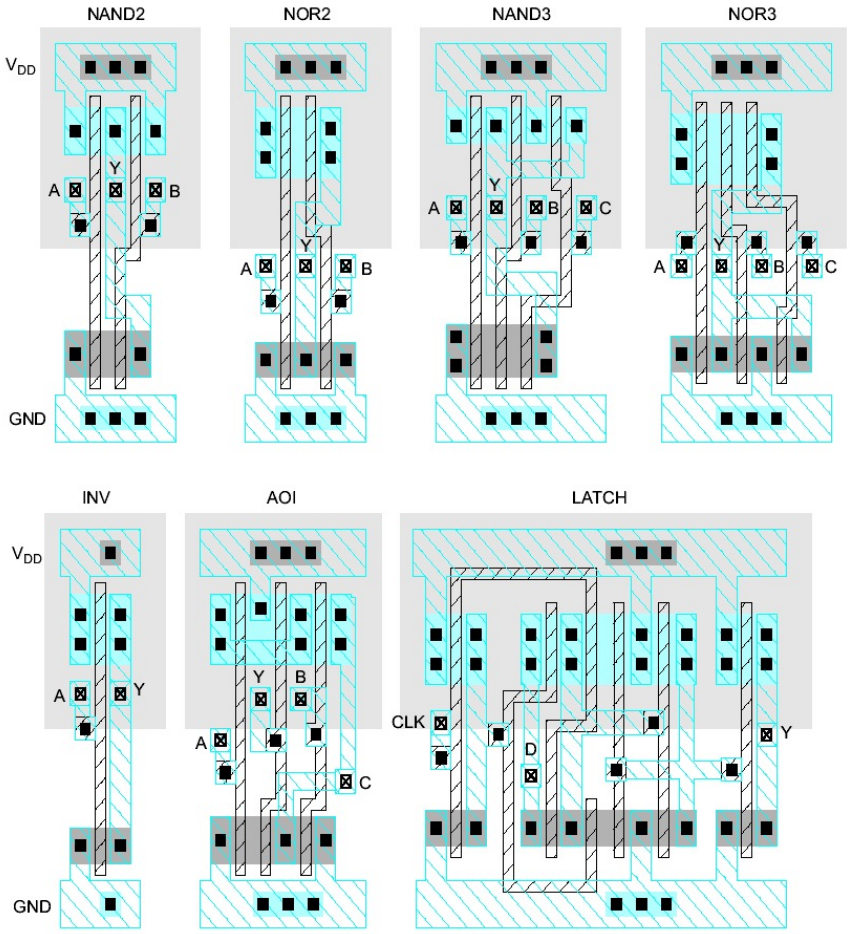


MIPS Layout



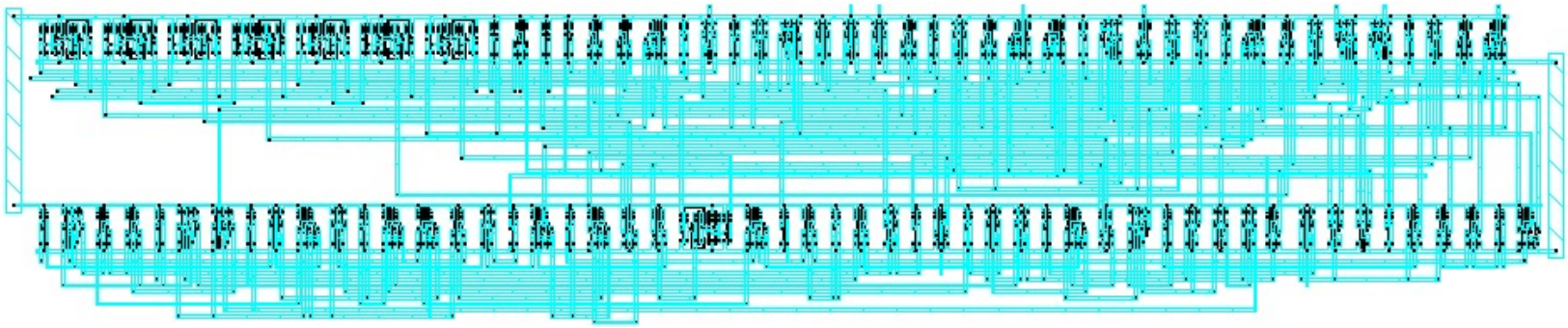
Standard Cells

- Uniform cell height
- Uniform well height
- M1 V_{DD} and GND rails
- M2 Access to I/Os
- Well / substrate taps
- Exploits regularity



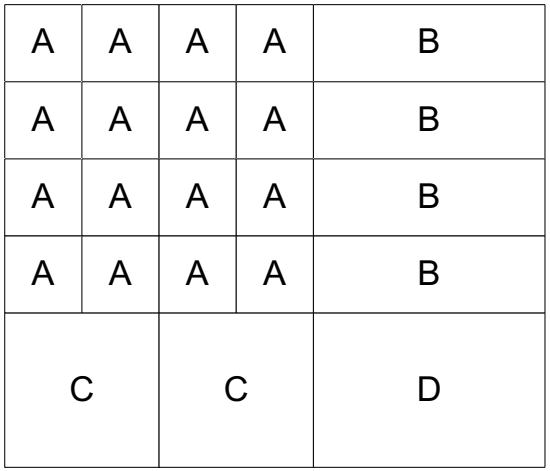
Synthesized Controller

- Synthesize HDL into gate-level netlist
- Place & Route using standard cell library



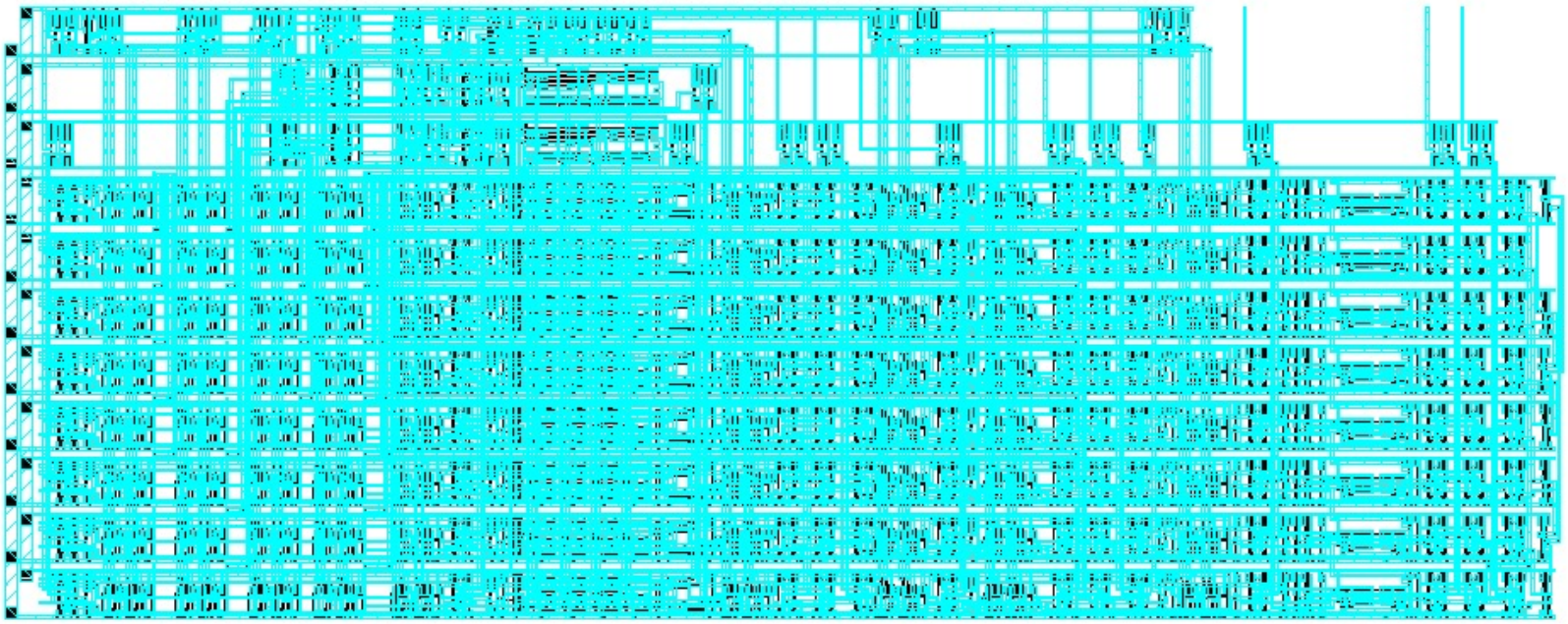
Pitch Matching

- Synthesized controller area is mostly wires
  - Design is smaller if wires run through/over cells
  - Smaller = faster, lower power as well!
- Design snap-together cells for datapaths and arrays
  - Plan wires into cells
  - Connect by abutment
    - Exploits locality
    - Takes lots of effort



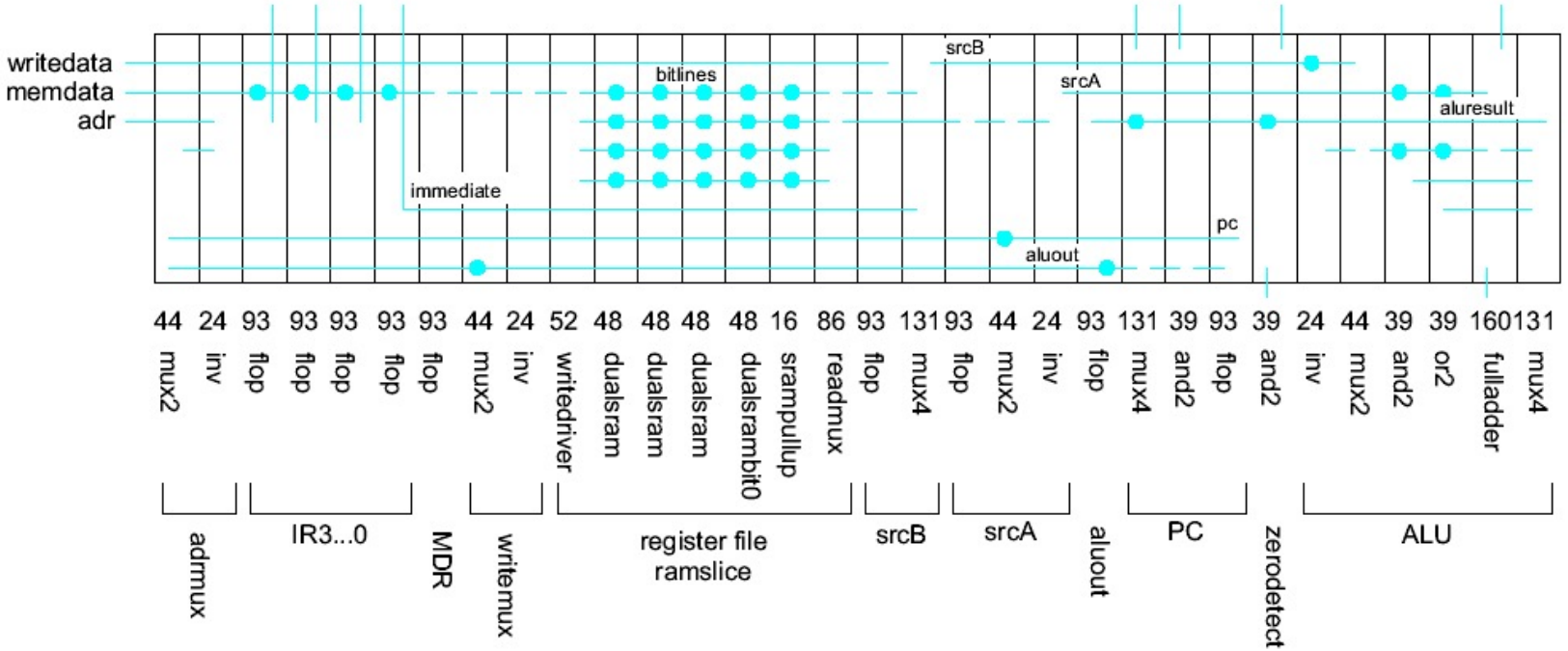
MIPS Datapath

- 8-bit datapath built from 8 bitslices (regularity)
- Zipper at top drives control signals to datapath



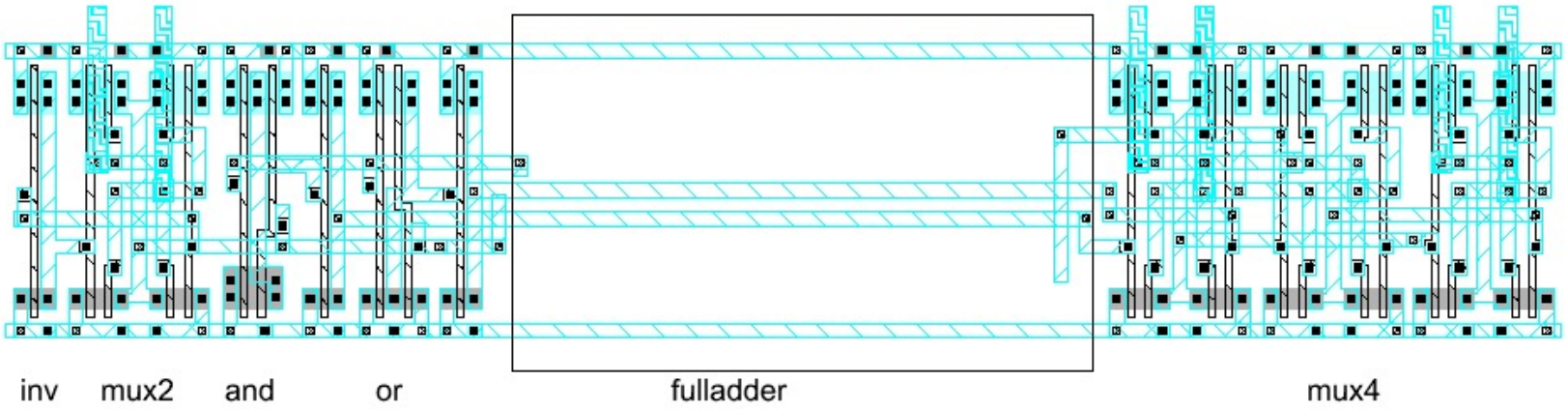
Slice Plans

- Slice plan for bitslice
  - Cell ordering, dimensions, wiring tracks
  - Arrange cells for wiring locality



MIPS ALU

- Arithmetic / Logic Unit is part of bitslice



Area Estimation

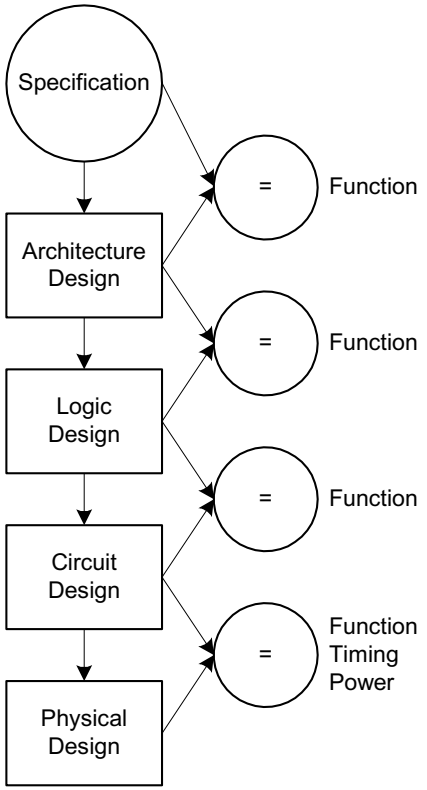
- Need area estimates to make floorplan
  - Compare to another block you already designed
  - Or estimate from transistor counts
  - Budget room for large wiring tracks
  - Your mileage may vary!

Table 1.10 Typical layout densities

Element	Area
random logic (2-level metal process)	1000 – 1500 λ^2 / transistor
datapath	250 – 750 λ^2 / transistor or 6 WL + 360 λ^2 / transistor
SRAM	1000 λ^2 / bit
DRAM (in a DRAM process)	100 λ^2 / bit
ROM	100 λ^2 / bit

Design Verification

- Fabrication is slow & expensive
  - MOSIS : $??K, 3 months
  - State of art: $??M, 1 month
- Debugging chips is very hard
  - Limited visibility into operation
- Prove design is right before building!
  - Logic simulation
  - Ckt. simulation / formal verification
  - Layout vs. schematic comparison
  - Design & electrical rule checks
- Verification is > 50% of effort on most chips!



Fabrication & Packaging

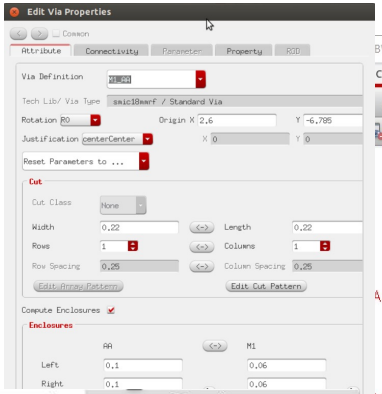
- Tapeout final layout
- Fabrication
  - 6, 8, 12” wafers
  - Optimized for throughput, not latency (10 weeks!)
  - Cut into individual dice
- Packaging
  - Bond gold wires from die I/O pads to package

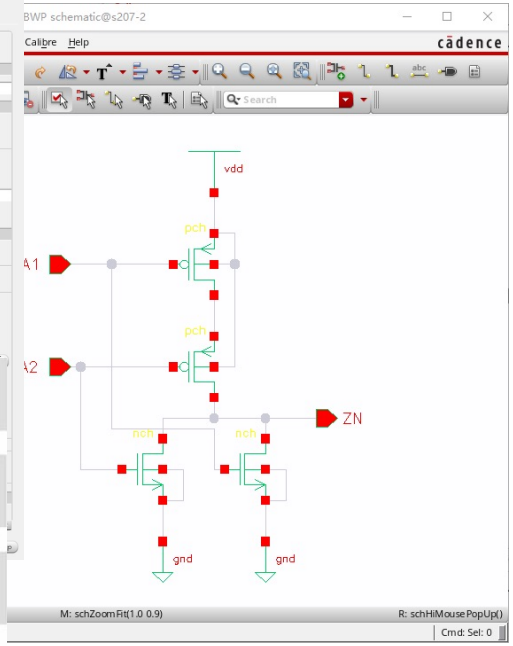
Testing

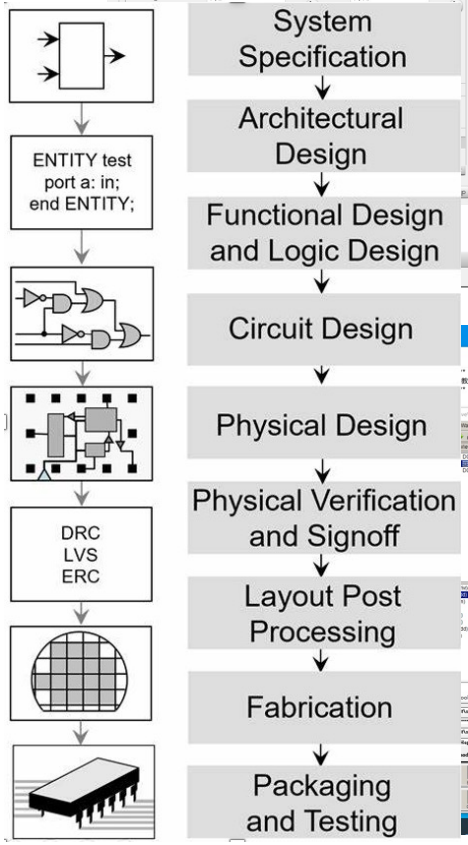
- Test that chip operates
  - Design errors
  - Manufacturing errors
- A single dust particle or wafer defect kills a die
  - Yields from 90% to < 10%
  - Depends on die size, maturity of process
  - Test each part before shipping to customer

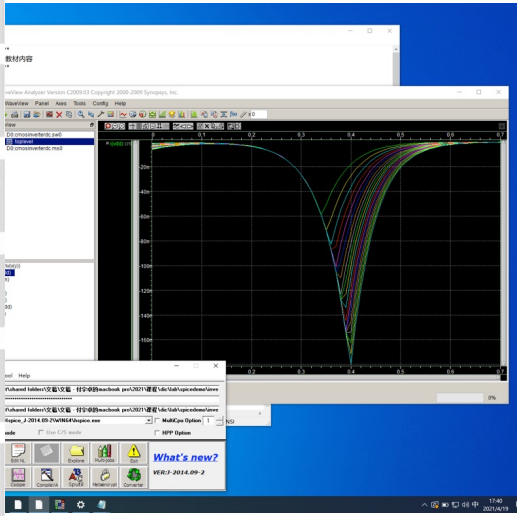
DIC Target

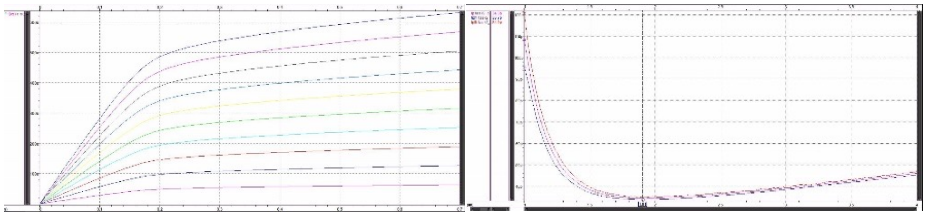
- ✓ Transistor level Optimizing
  - Inverter
  - Combinational logic
  - Sequential logic
  - Datapath design
- ✓ Modeling
  - Clocking issues
  - Wire issues
- ✓ Tooling
  - HSPICE programming
  - Cadence flow











Why digital circuits ?

- Design and implement the module libraries
- Critical timing path still need circuit design
- Some technologies tend to be pushed to its limit need circuit design
- New design issues and constraints tend to emerge over time
- Model not always show the truth

We are the pioneers of the NEW IC technology!