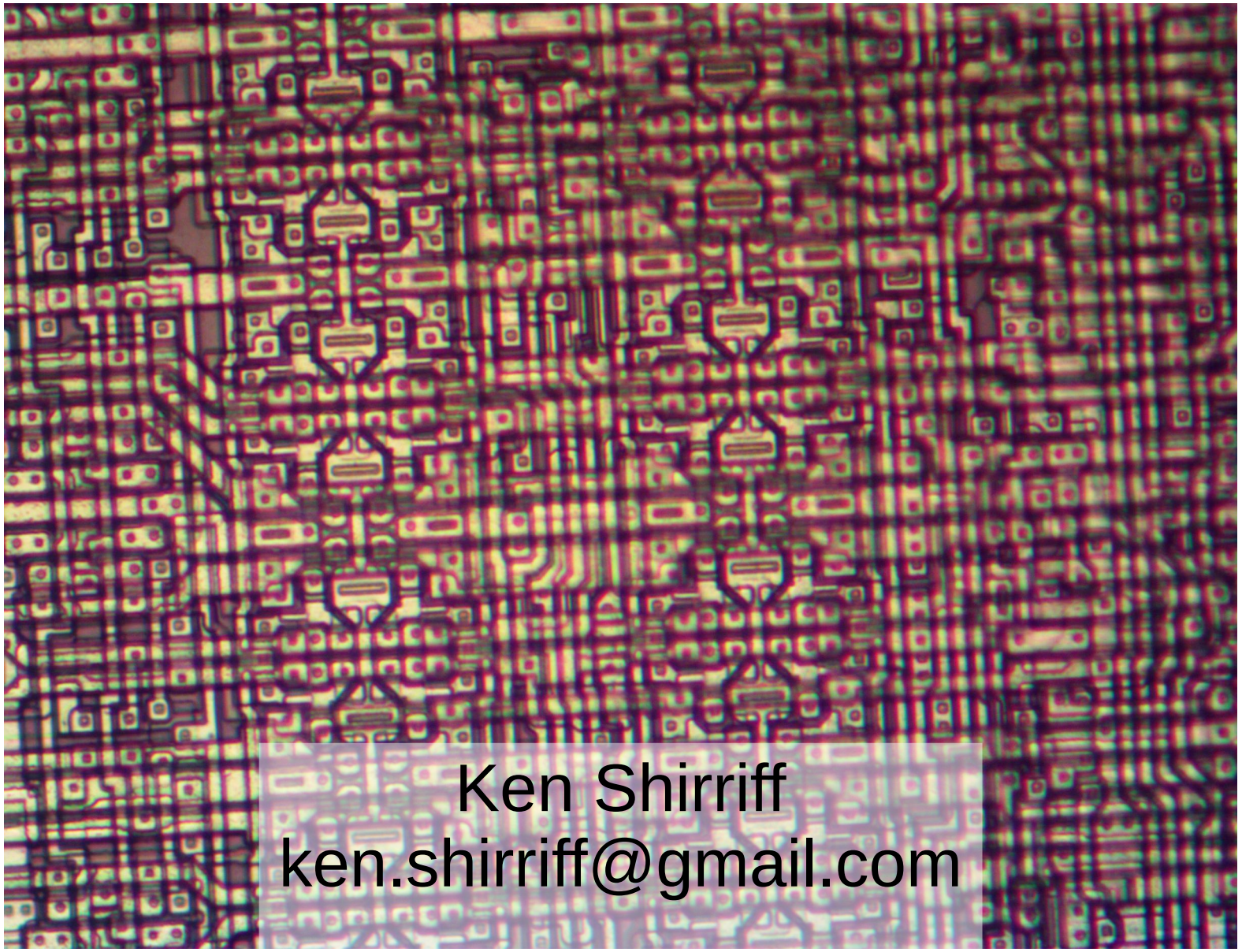
Reverse engineering the first FPGA: Inside the XC2064



Ken Shirriff
ken.shirriff@gmail.com

Field-Programmable Gate Array

- Arbitrary digital logic in a chip
- Can do anything from logic analyzer to HDMI converter to microprocessor
- Programmed in e.g. Verilog
  - Bitstream (secret format) generated and loaded into the chip
- Invented by Xilinx in 1985:
  - XC2064
    - 64 logic blocks vs millions today
  - Replace TTL glue logic chips
    - Equivalent to 1000-1500 gates

Why reverse engineer an FPGA?

- Figure out undocumented bitstream
  - Build open-source tools
  - Interpret bitstreams in products
- Curiosity: How is an FPGA is implemented?

How does an FPGA work?

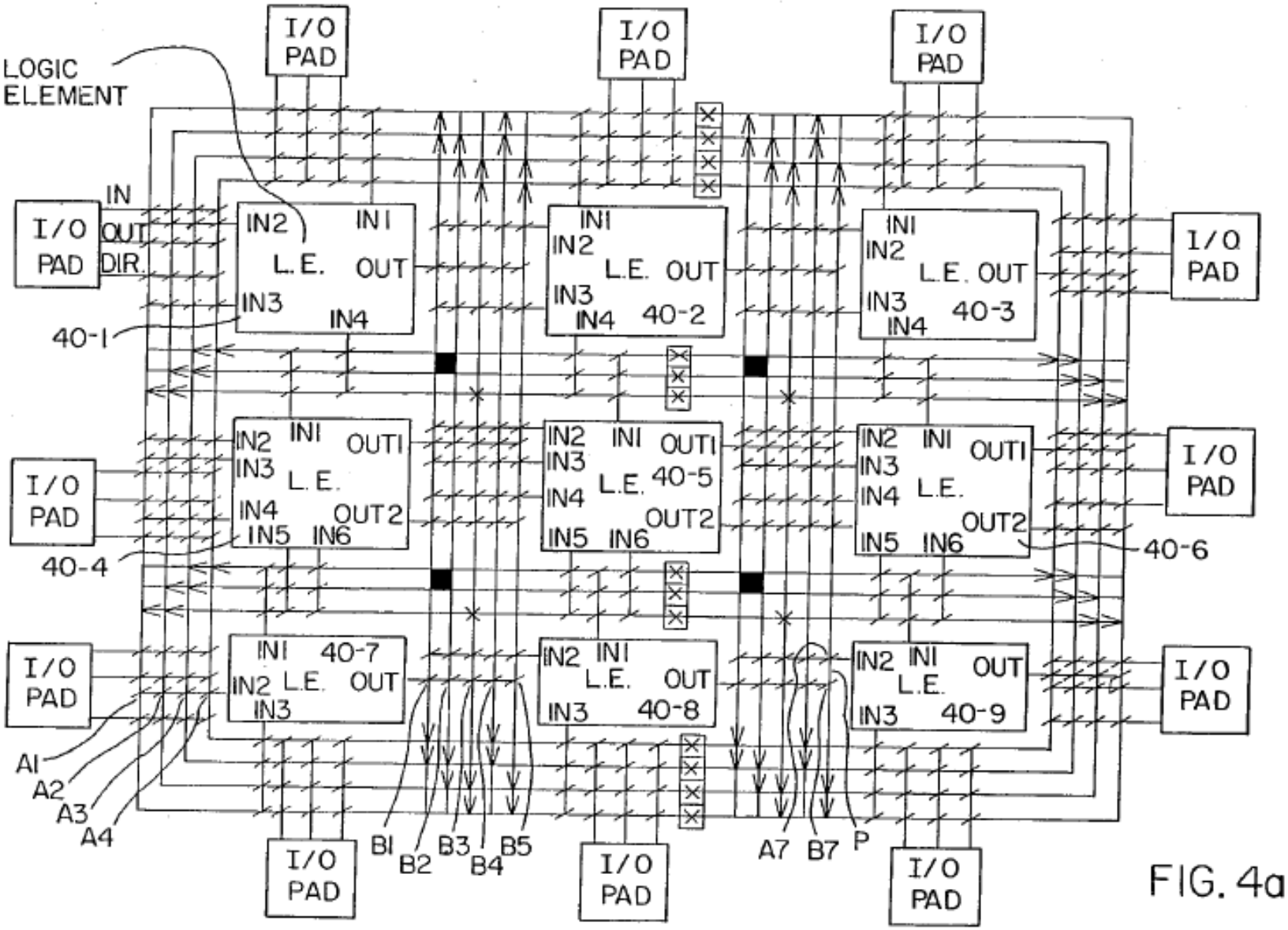
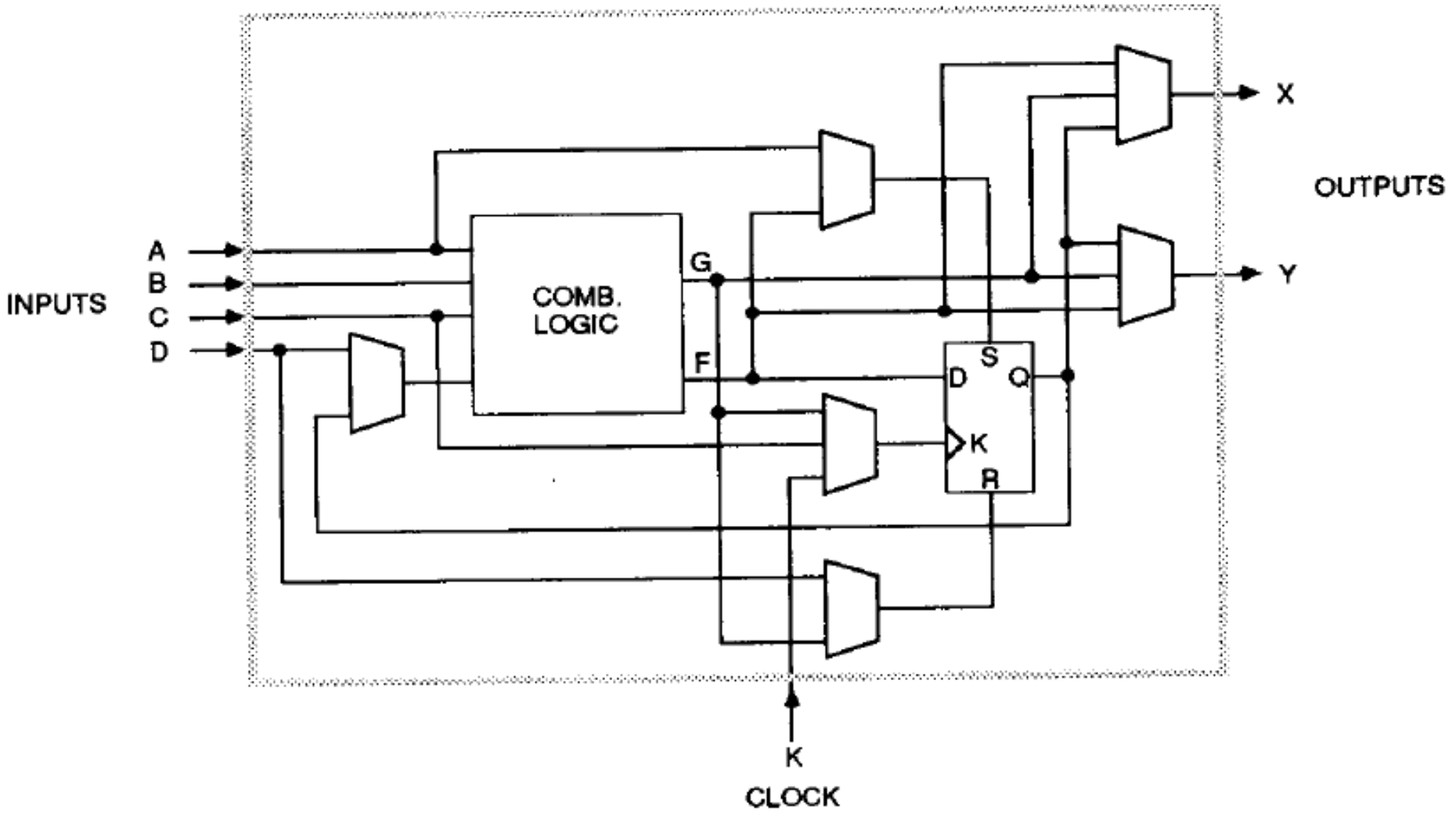


FIG. 4a

Xilinx patent 4,642,487

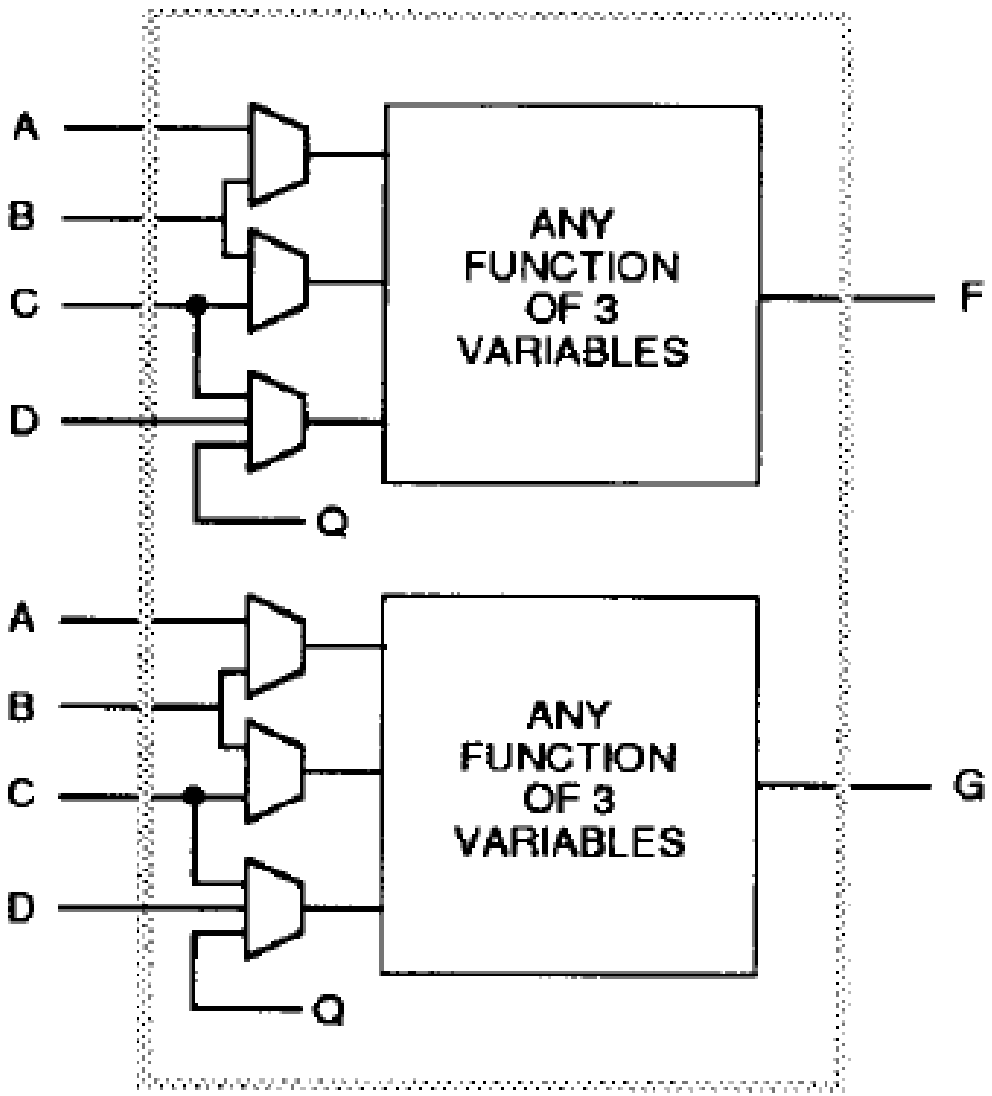
CLB: Configurable Logic Block



http://www.itisravenna.gov.it/sheet/XC2000FM.PDF

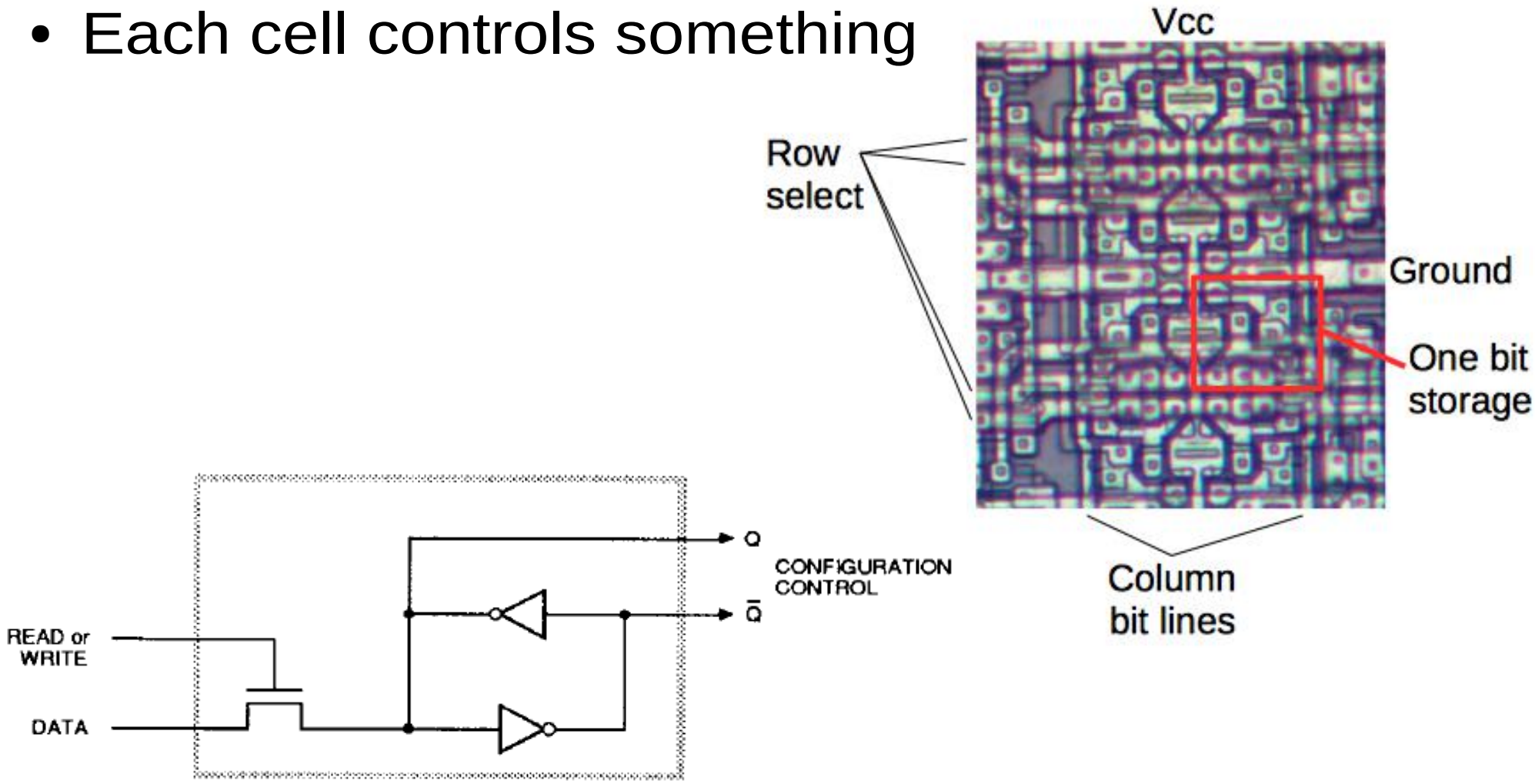
LUT: Lookup Table

- Arbitrary logic function
- 8 bit truth table
  - Stored in SRAM

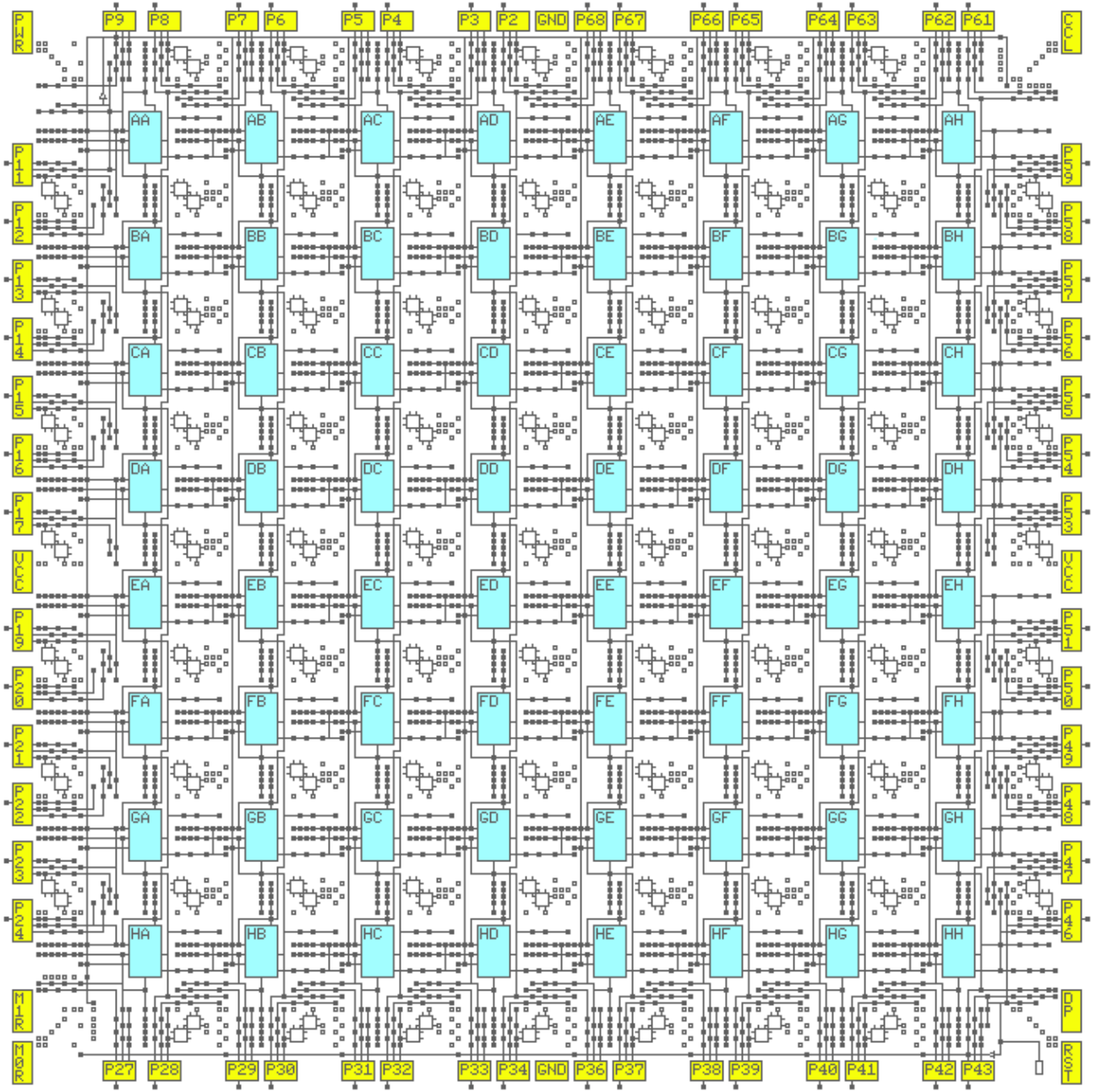


Programming an FPGA

- Static RAM cells
- Each cell controls something

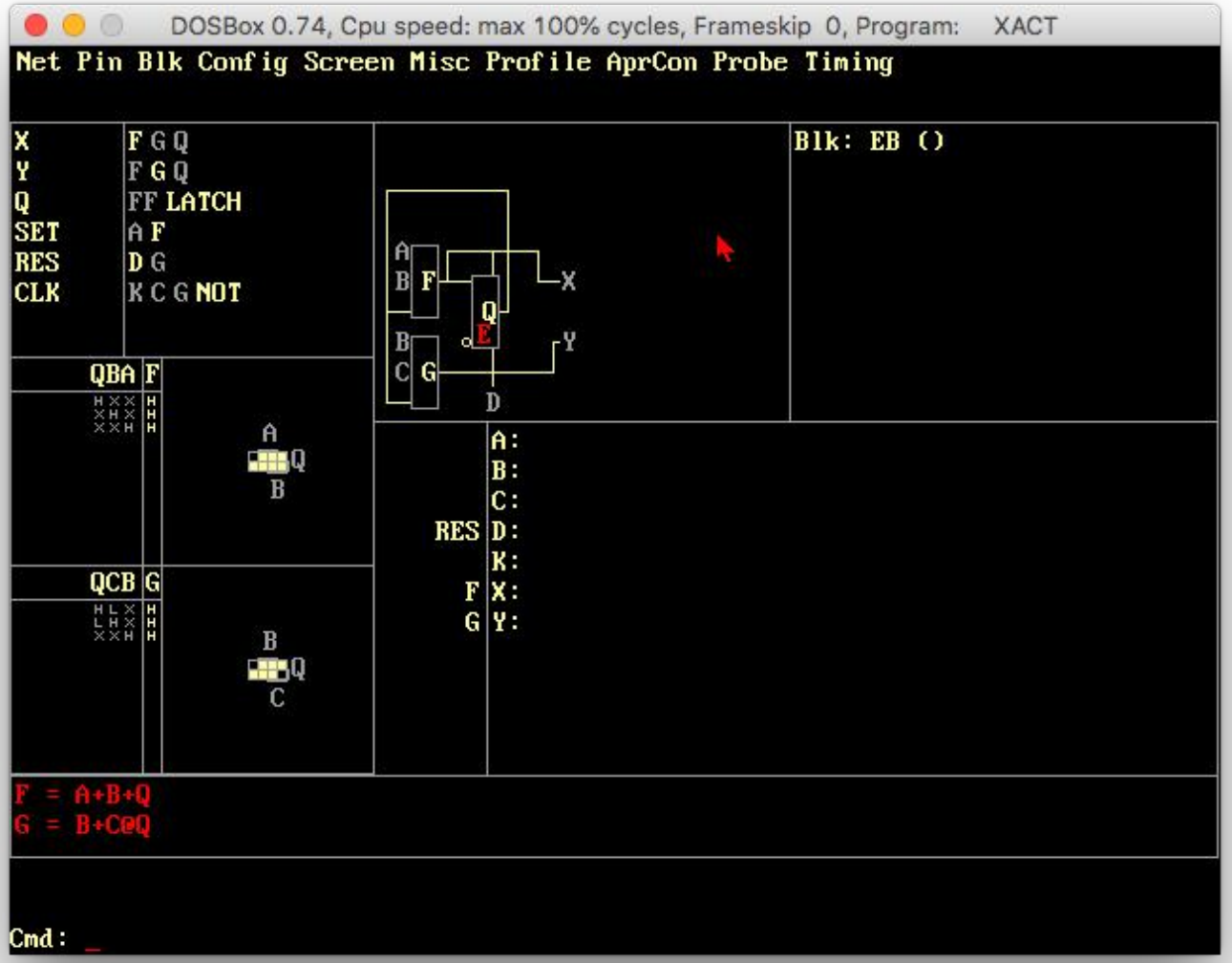


XC2064: 8x8 grid of CLBs



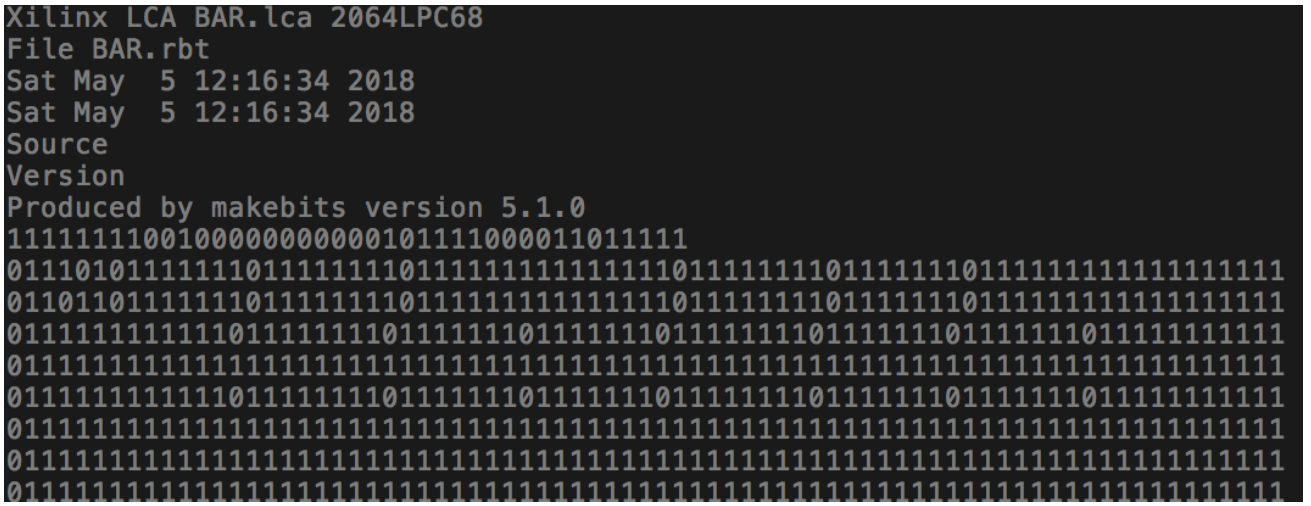
XACT

- DOS-based FPGA design tool, $12 000



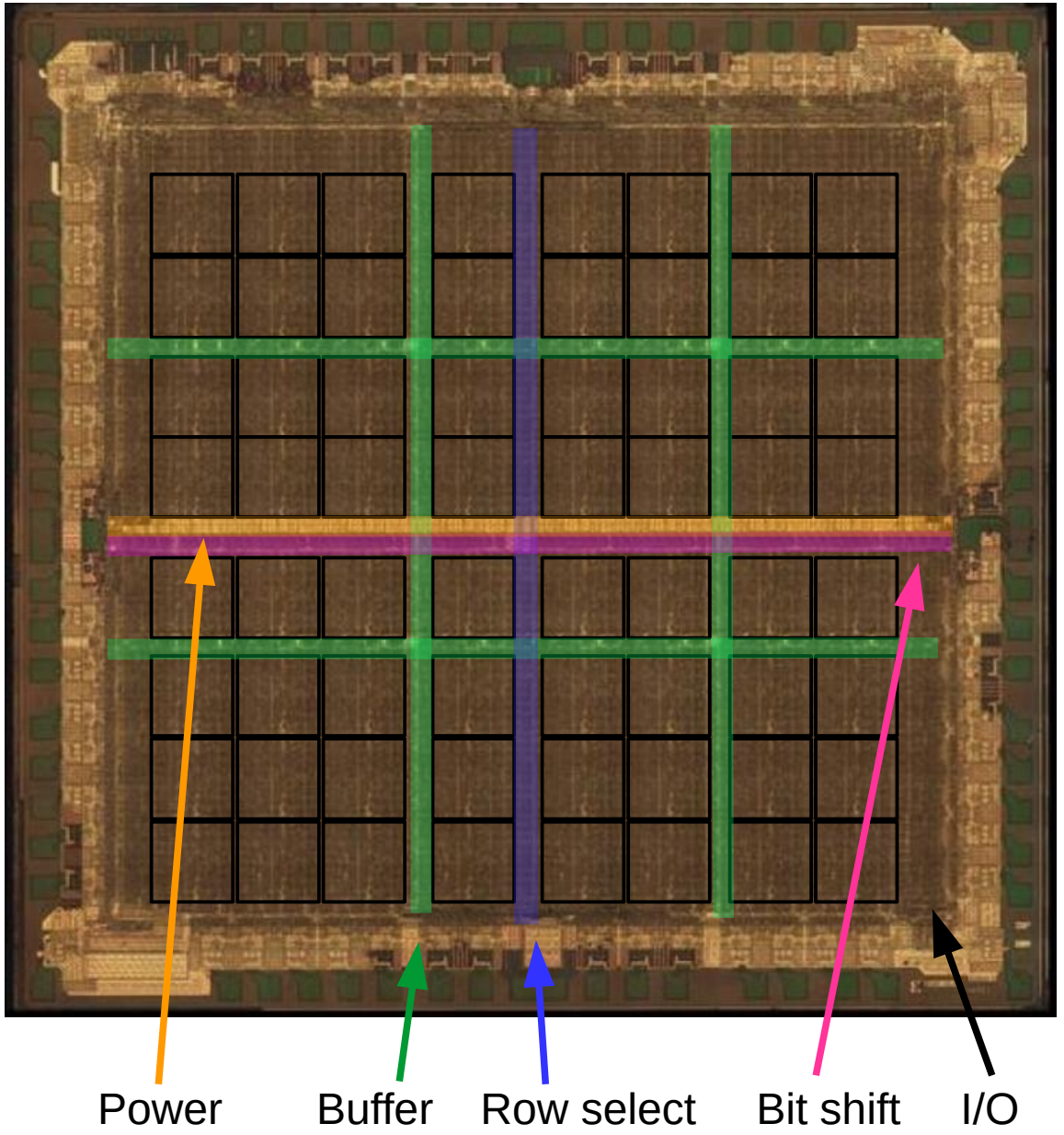
Reverse engineering with XACT

- Generate configs, changing features incrementally
- Examine bitstream to find meanings of bits



- Not regular. Bits scattered in file. Strange encoding.
- 71x160 bits, 8x18 blocks for CLBs

XC2064 chip layout

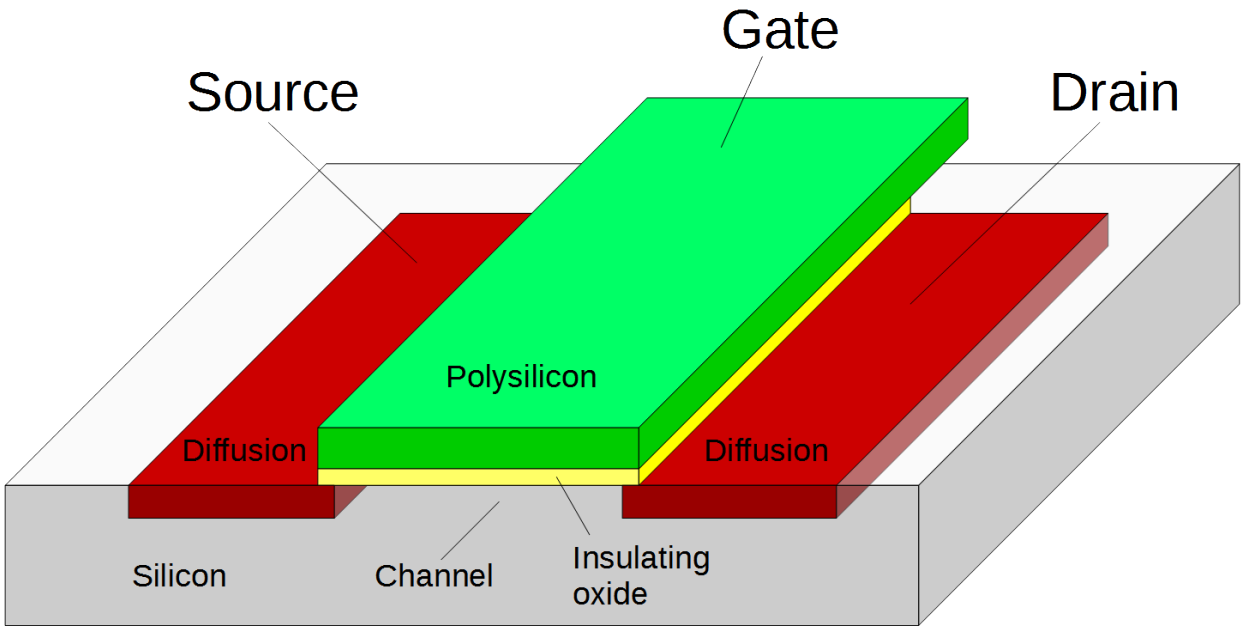


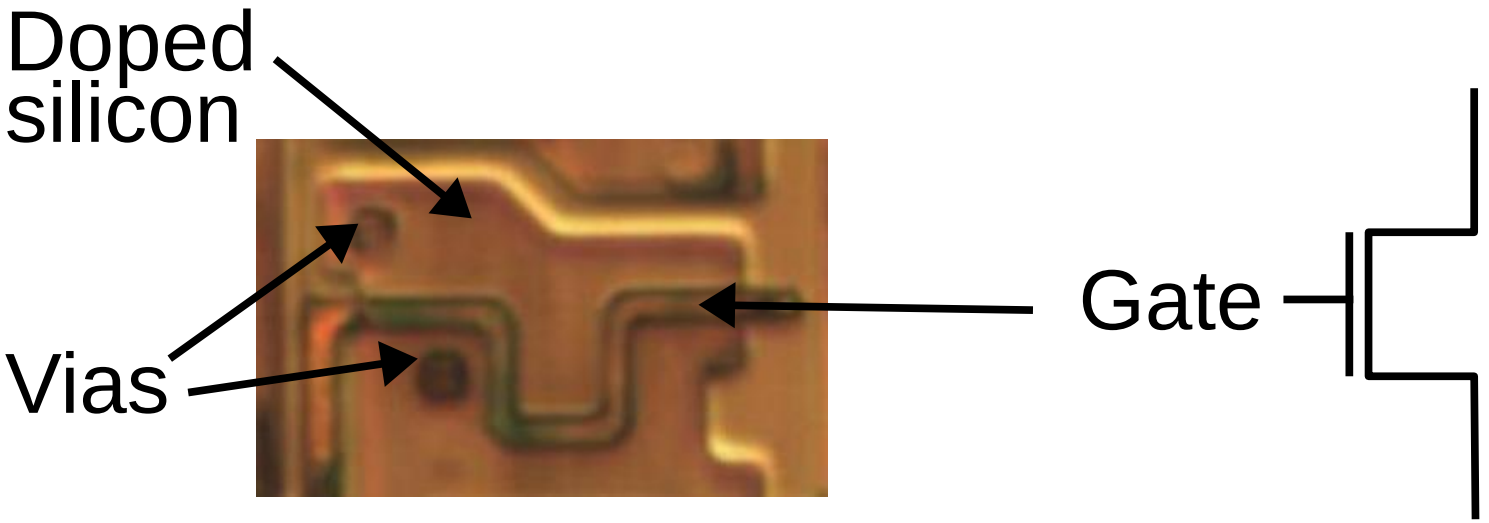
- CLB and routing combined
- Layout == bitstream
- Loaded row at a time
- Buffers: extra bits

http://siliconpr0n.org/map/xilinx/xc2064

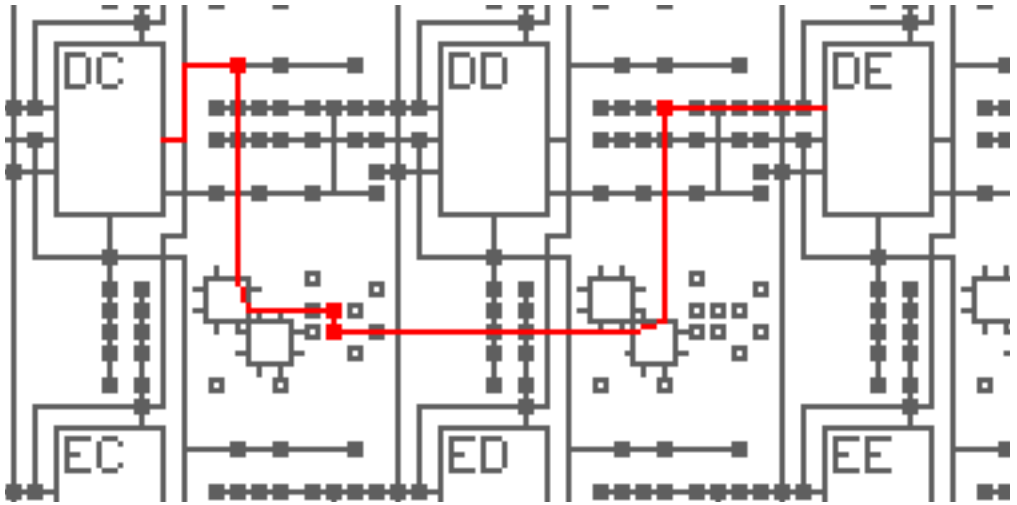
NMOS transistors

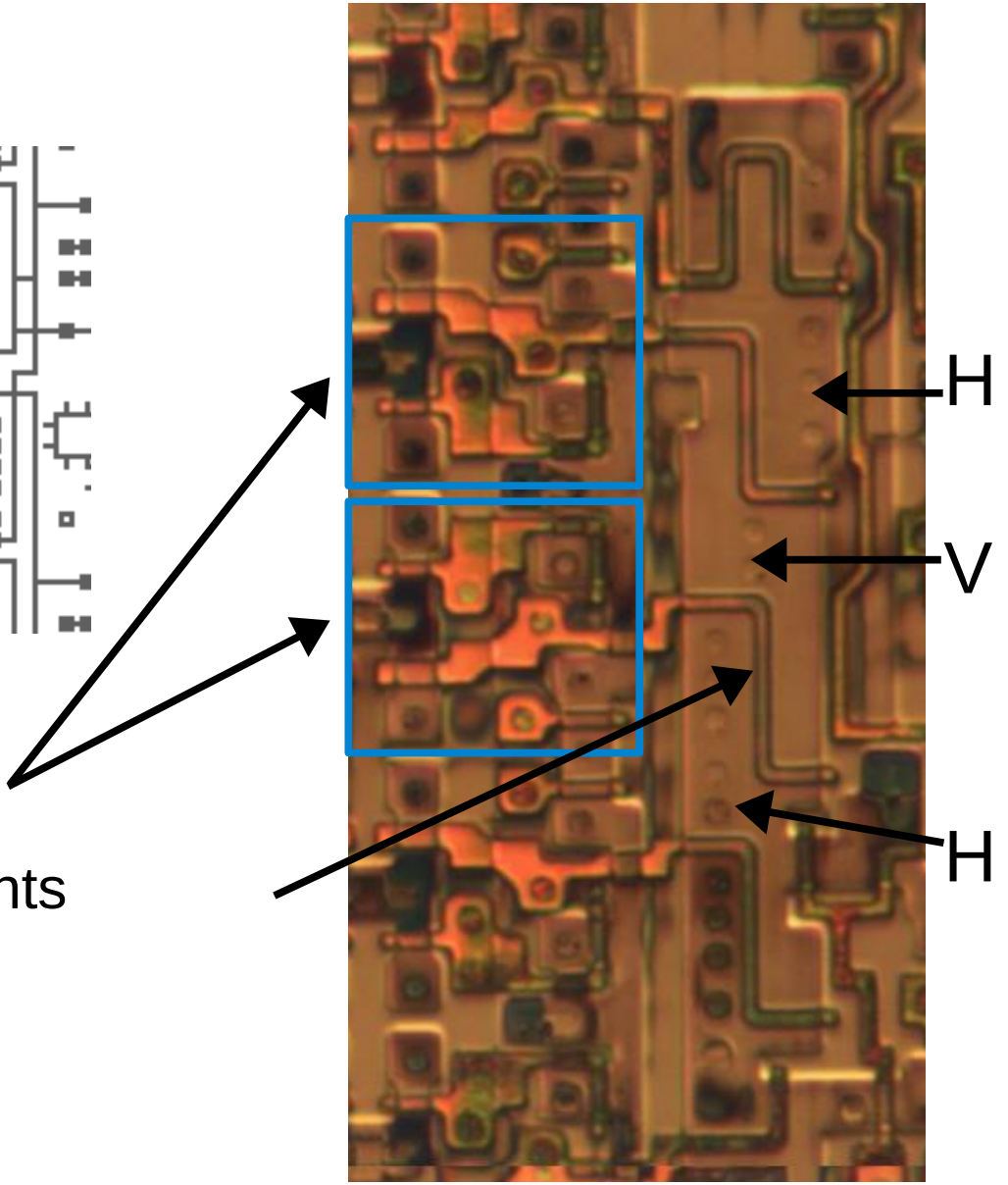
- Switch
  - Closed by gate





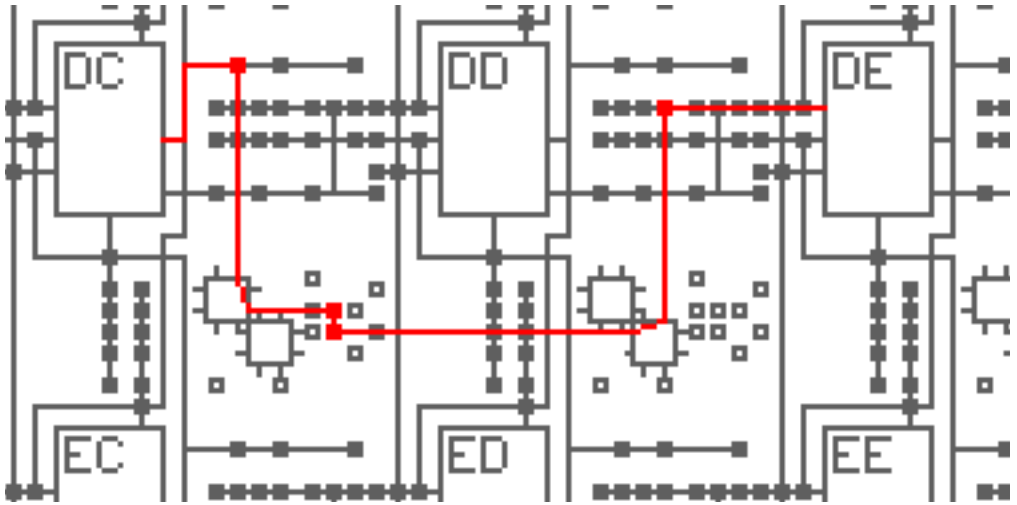
Routing points

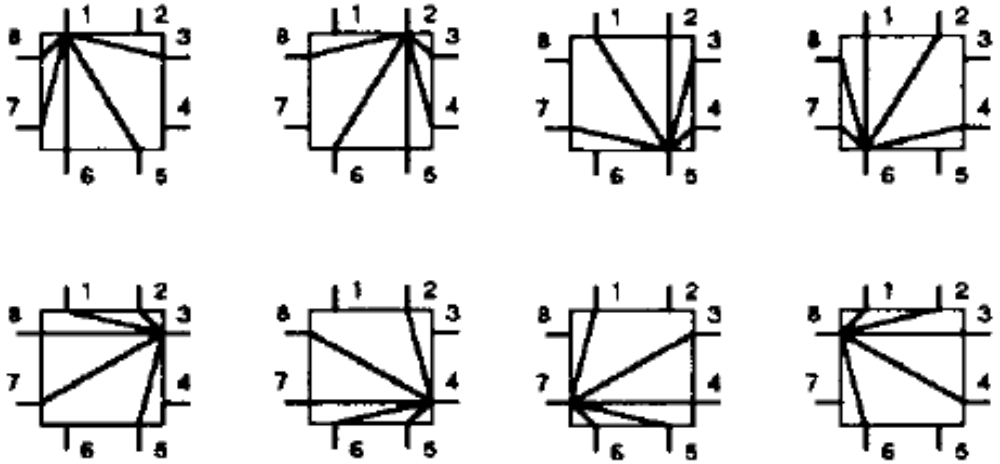


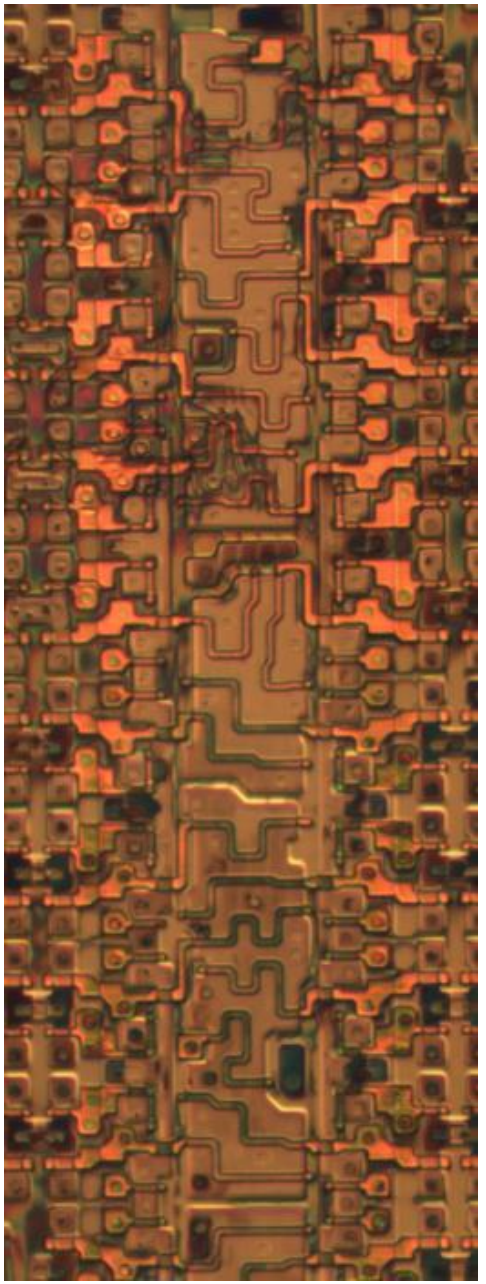


- Controlled by SRAM
- Pass transistors for switch points
- Can map bits to switches
- Overlapping transistors

20-point matrix switch

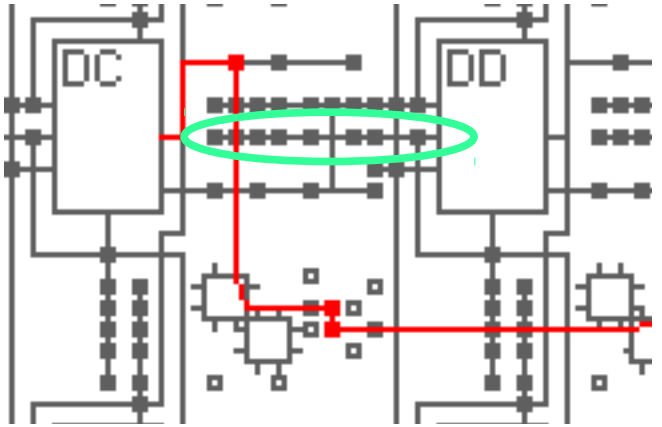


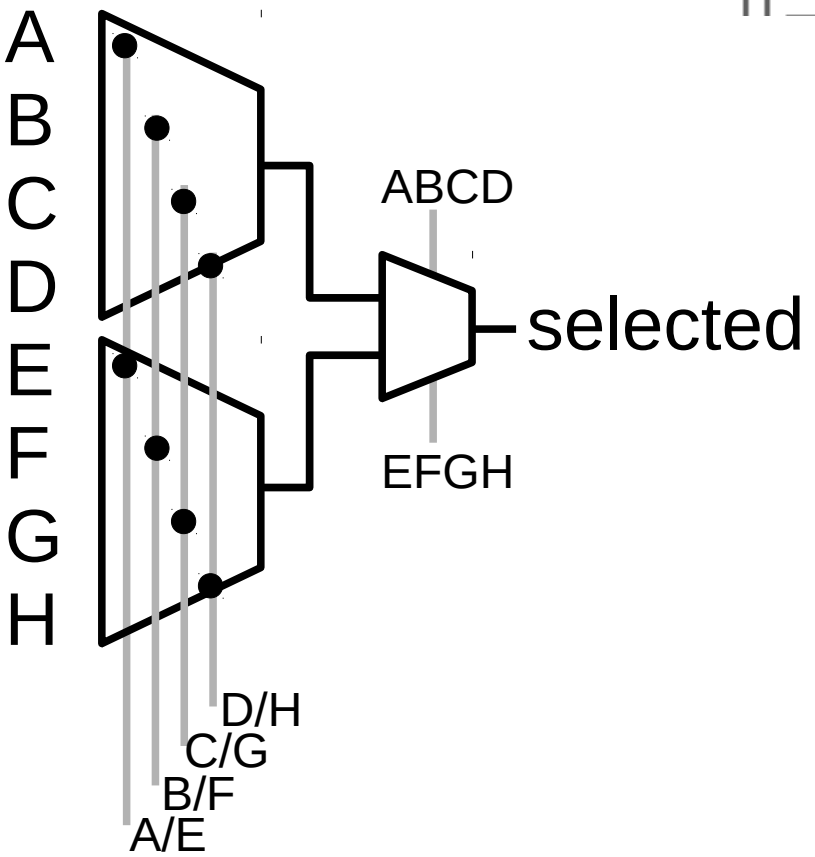




Input routing

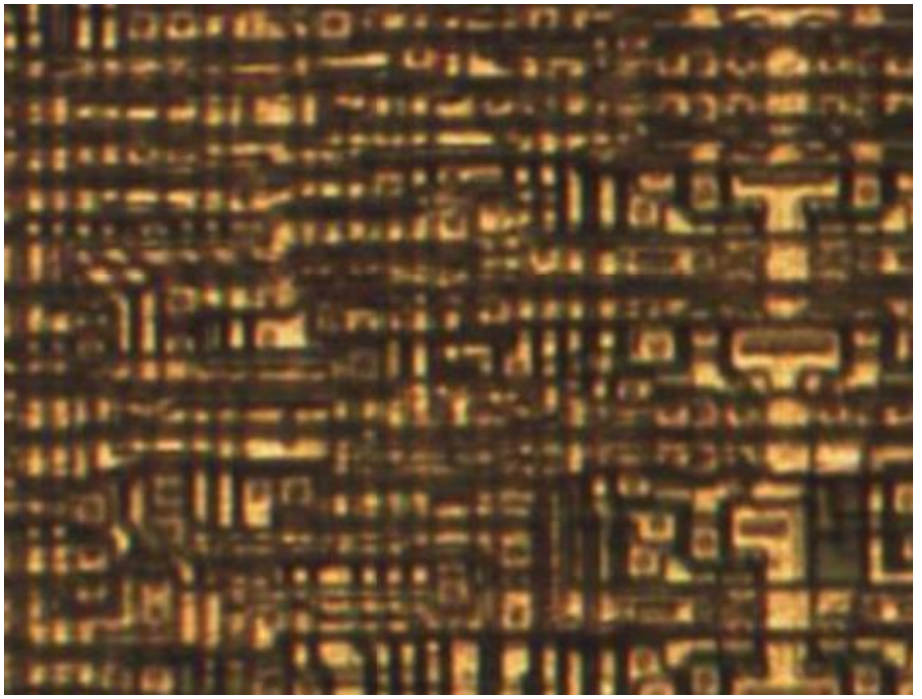
- Use 5 bits instead of 8
- Two multiplexer levels
- Mux is just pass transistors





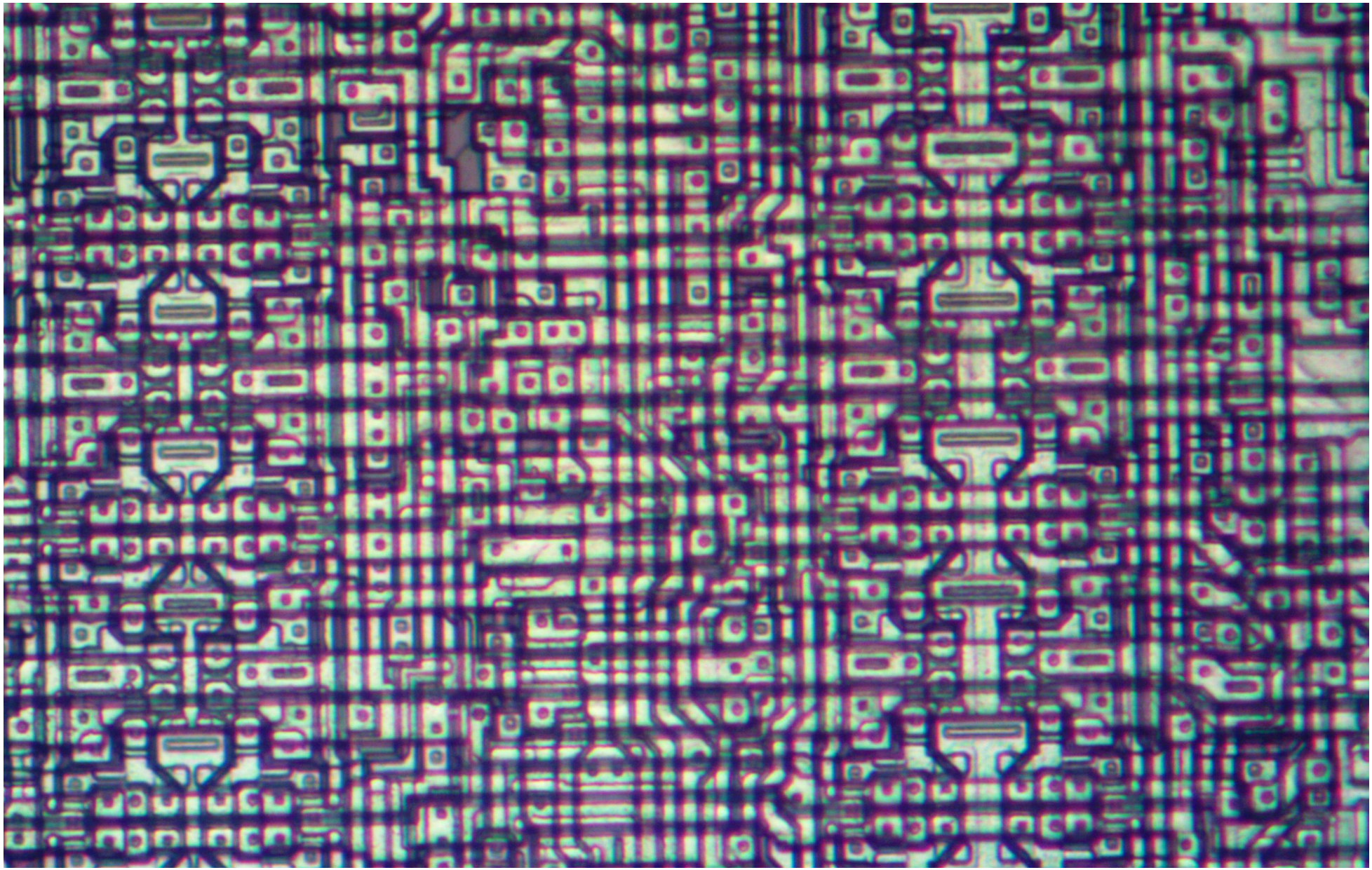
Trouble with the metal layers

- Two layers of metal
  - Top: mostly horizontal
  - Bottom: vertical
- Layers blur together in die photos

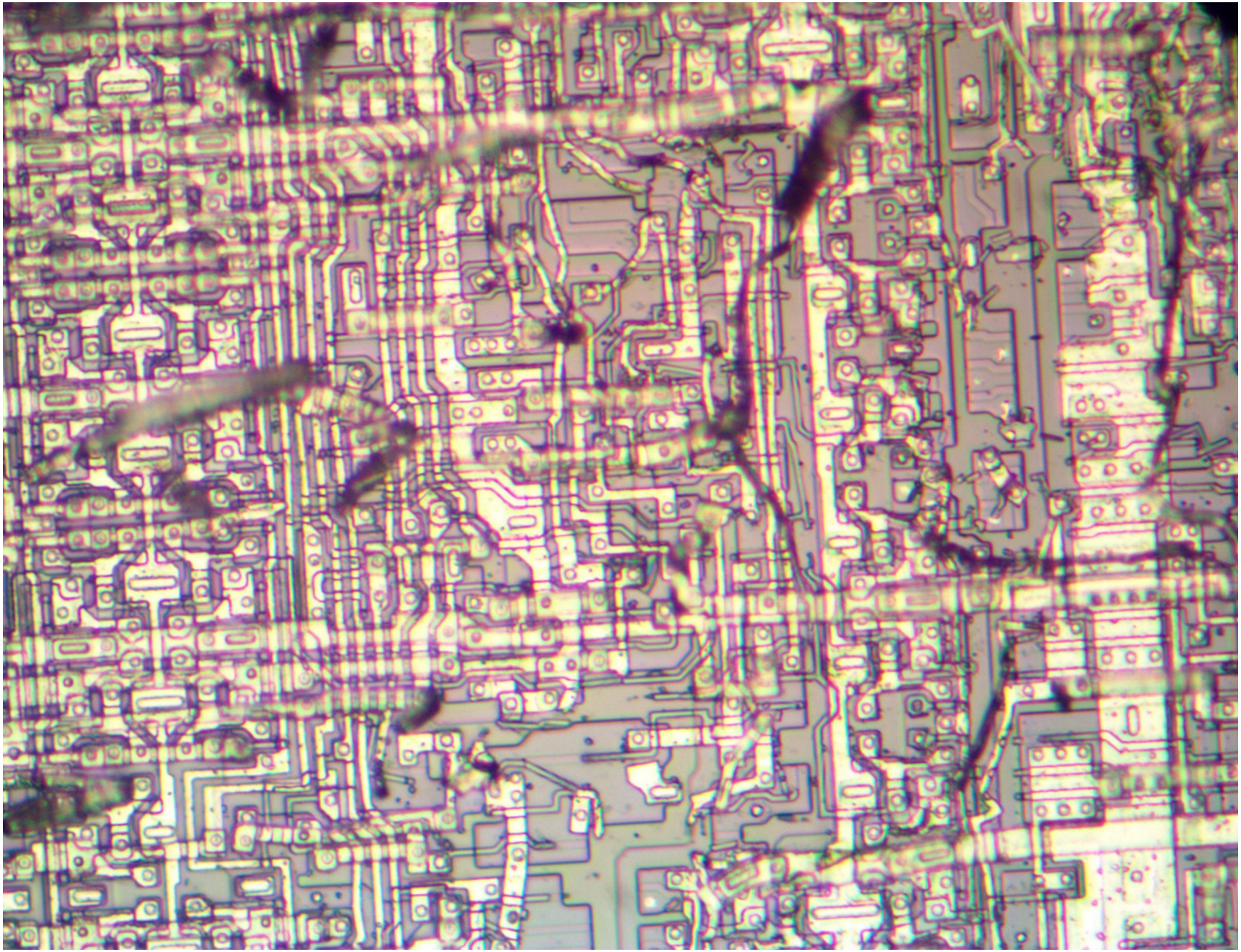


http://siliconpr0n.org/map/xilinx/xc2064

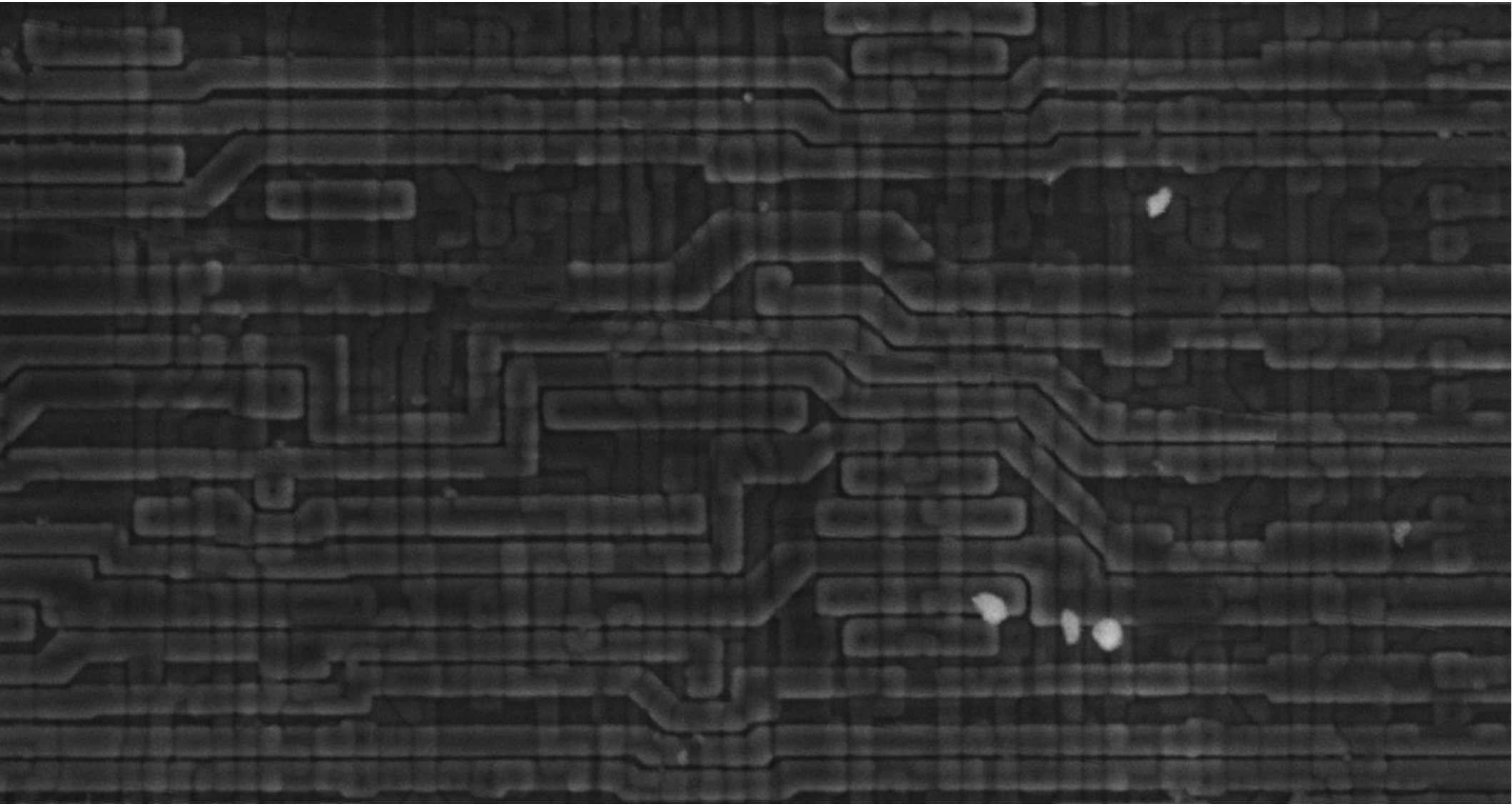
Sharper die photos



Etch oxide and metal layers

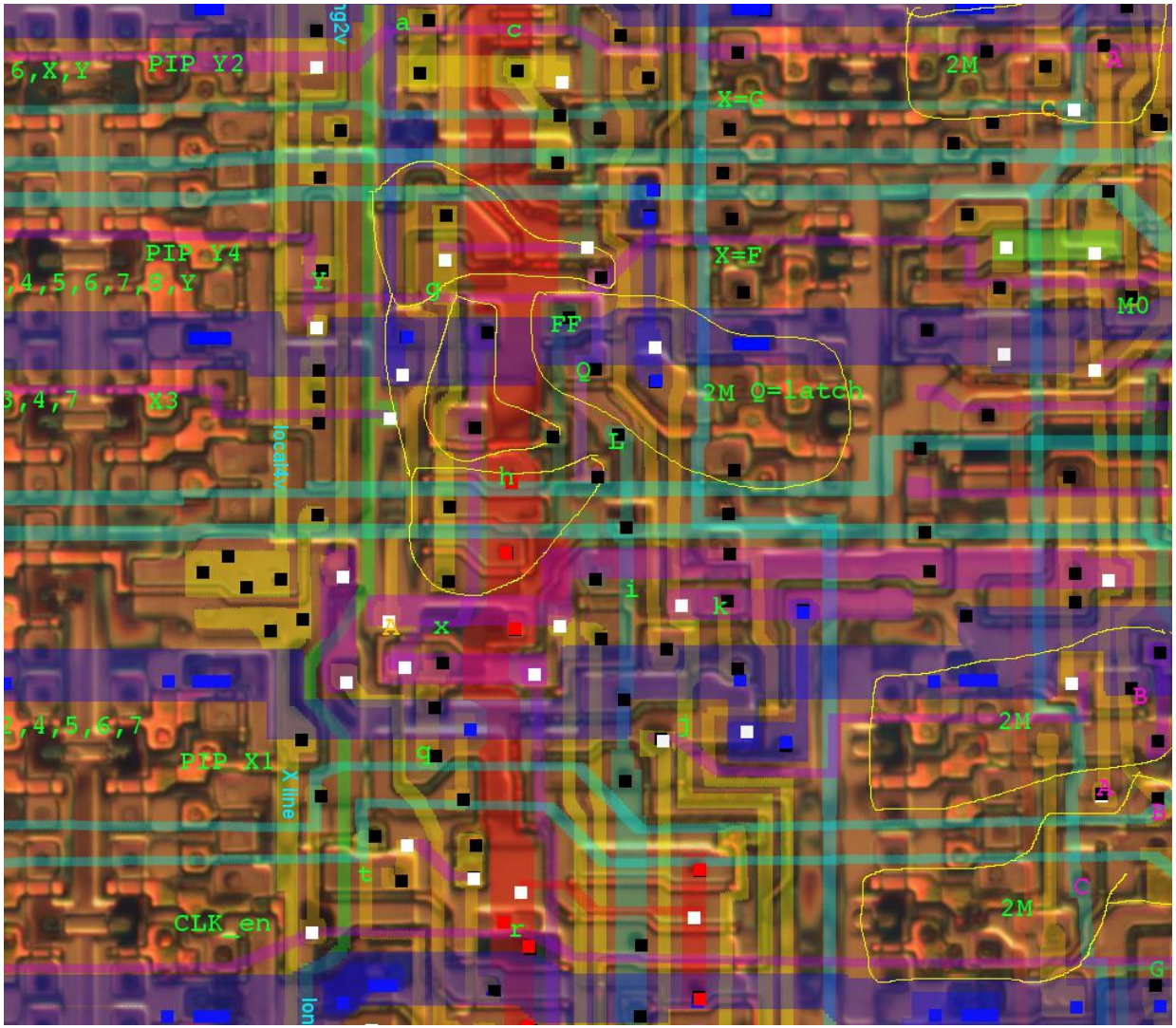


Electron microscope



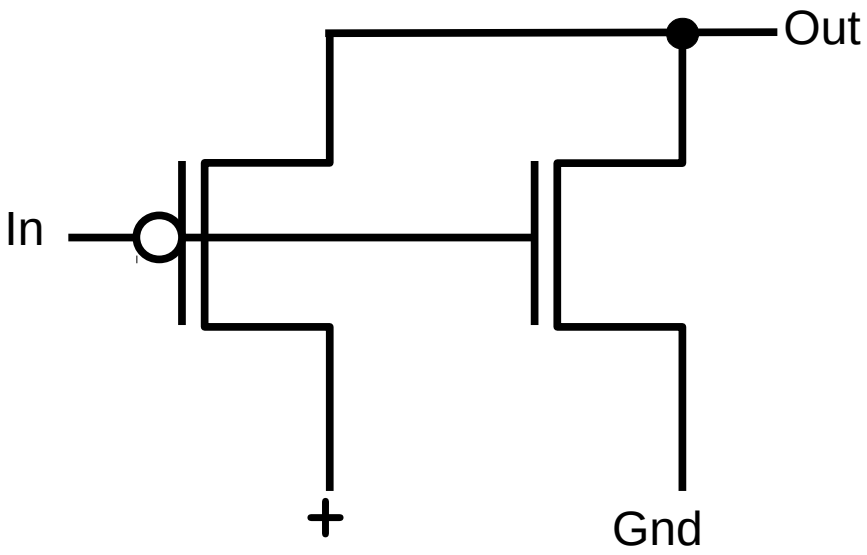
Tools: GIMP

- Trace out connections
  - Trial-and-error
  - Not Polychip
- Lots of layers
- Need better system



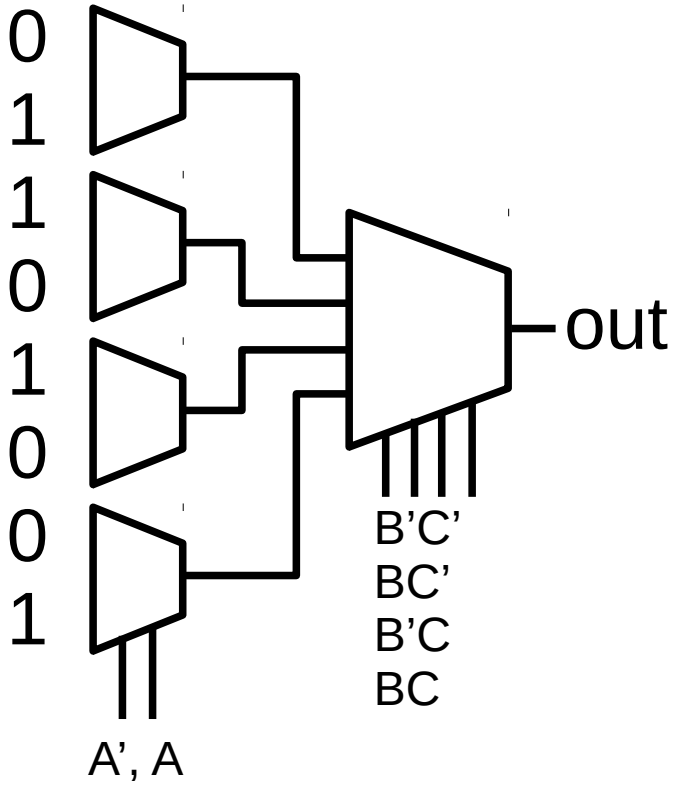
CMOS inverter

- 0 turns PMOS on: pulls output high
- 1 turns NMOS on: pulls output low

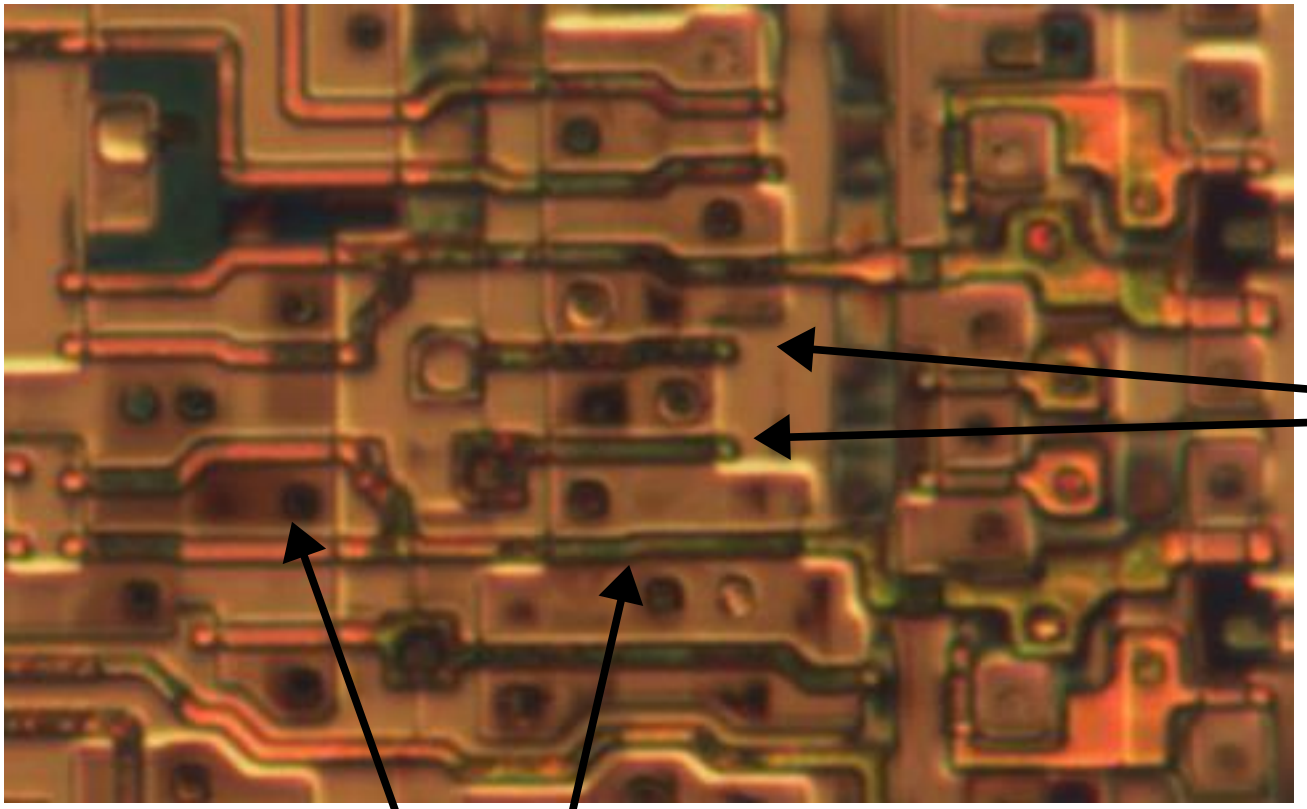


LUT multiplexers

- Program truth table into SRAM
- E.g. A xor B xor C

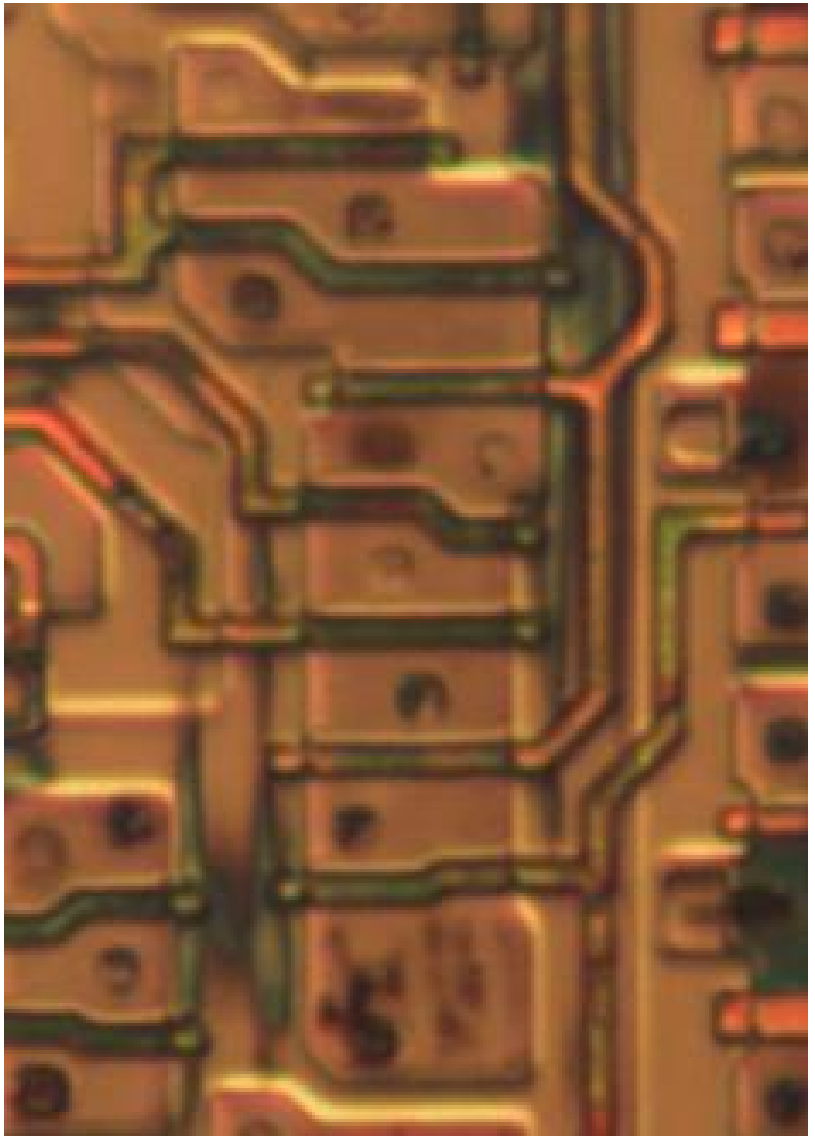


LUT implementation



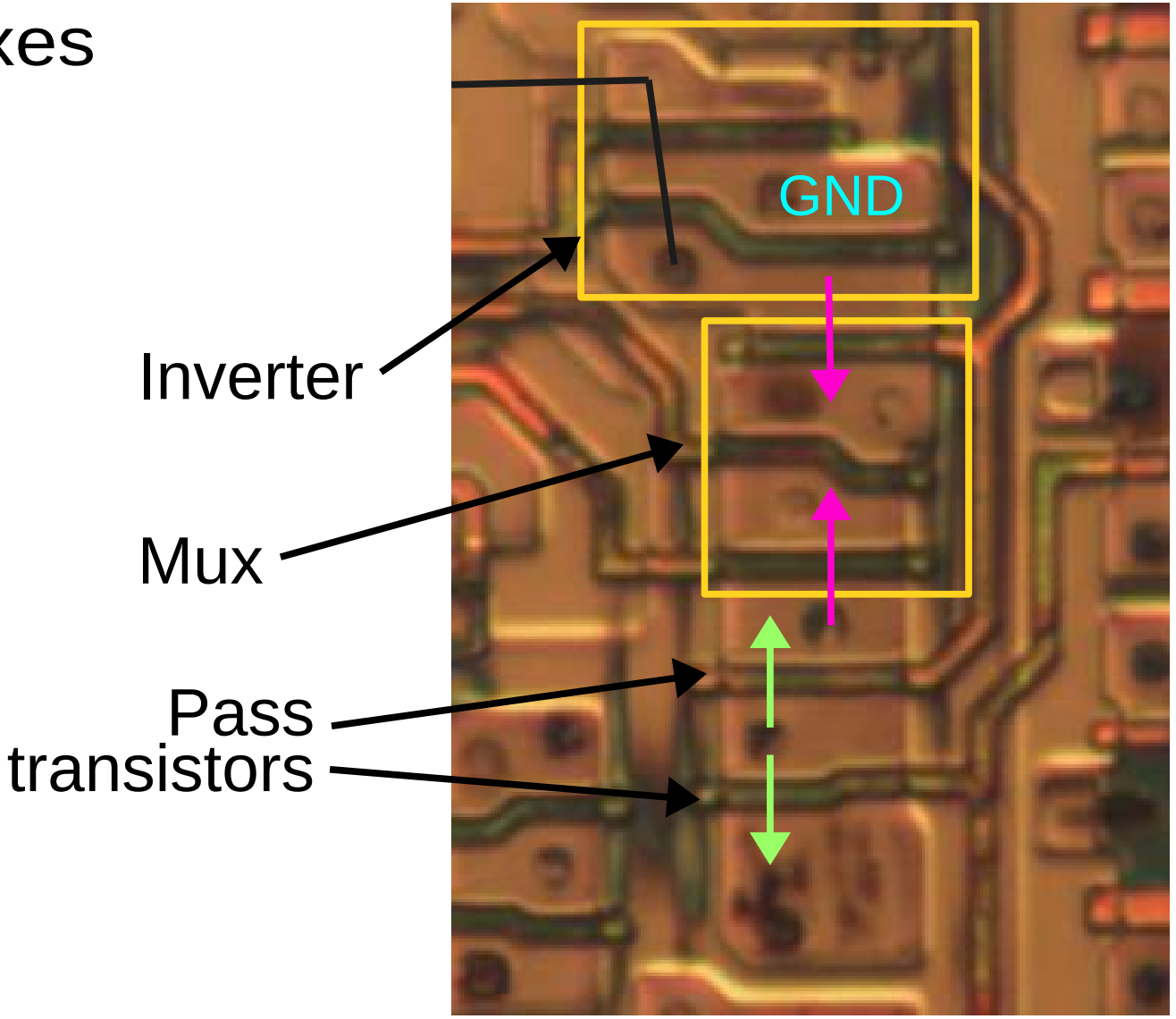
Complex gates

- Part of input routing
- 7-input NAND gate?

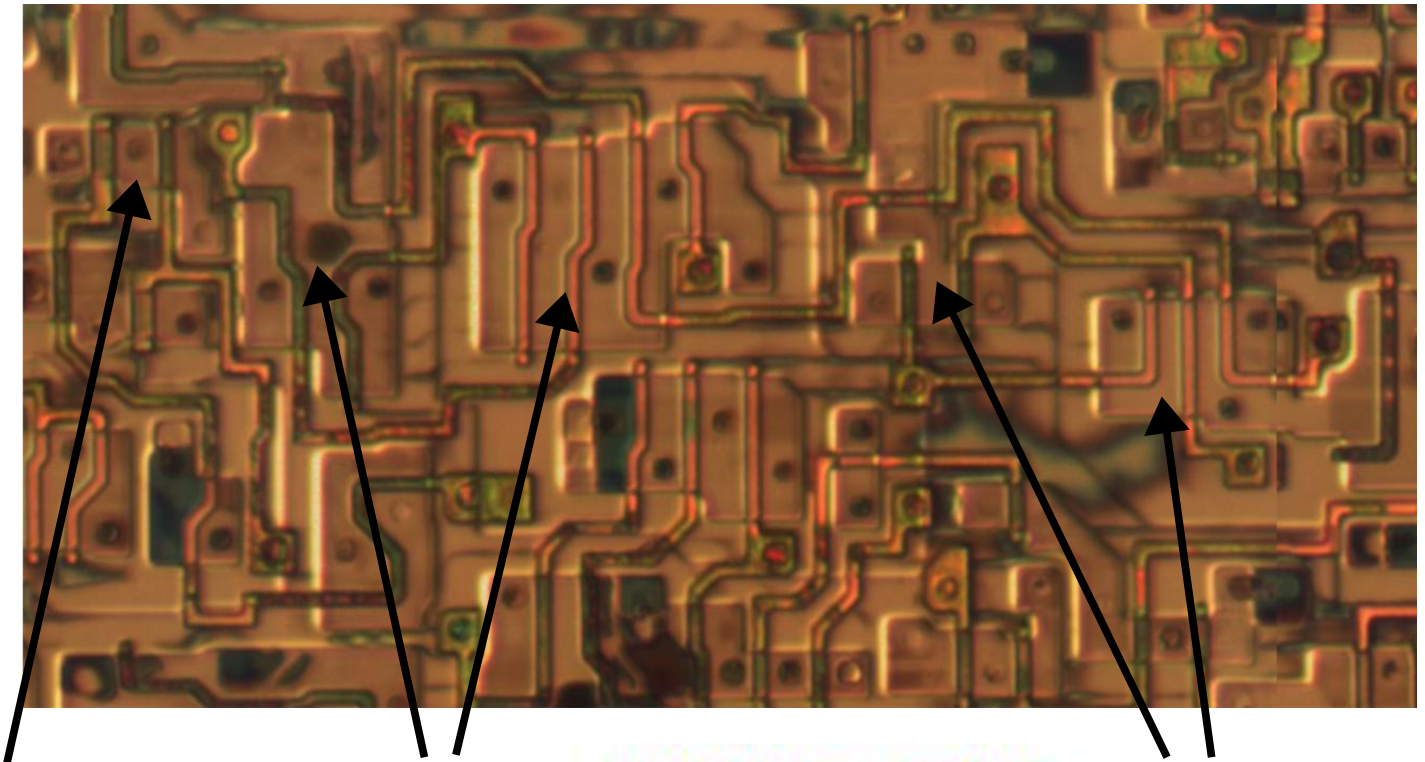


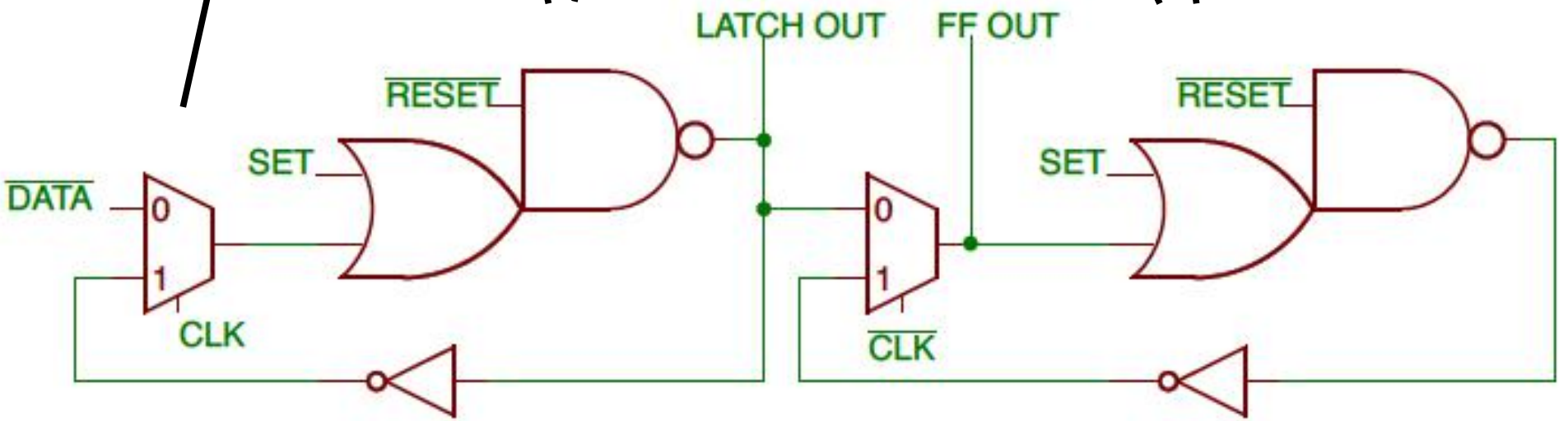
Complex gates

- Inverter + muxes

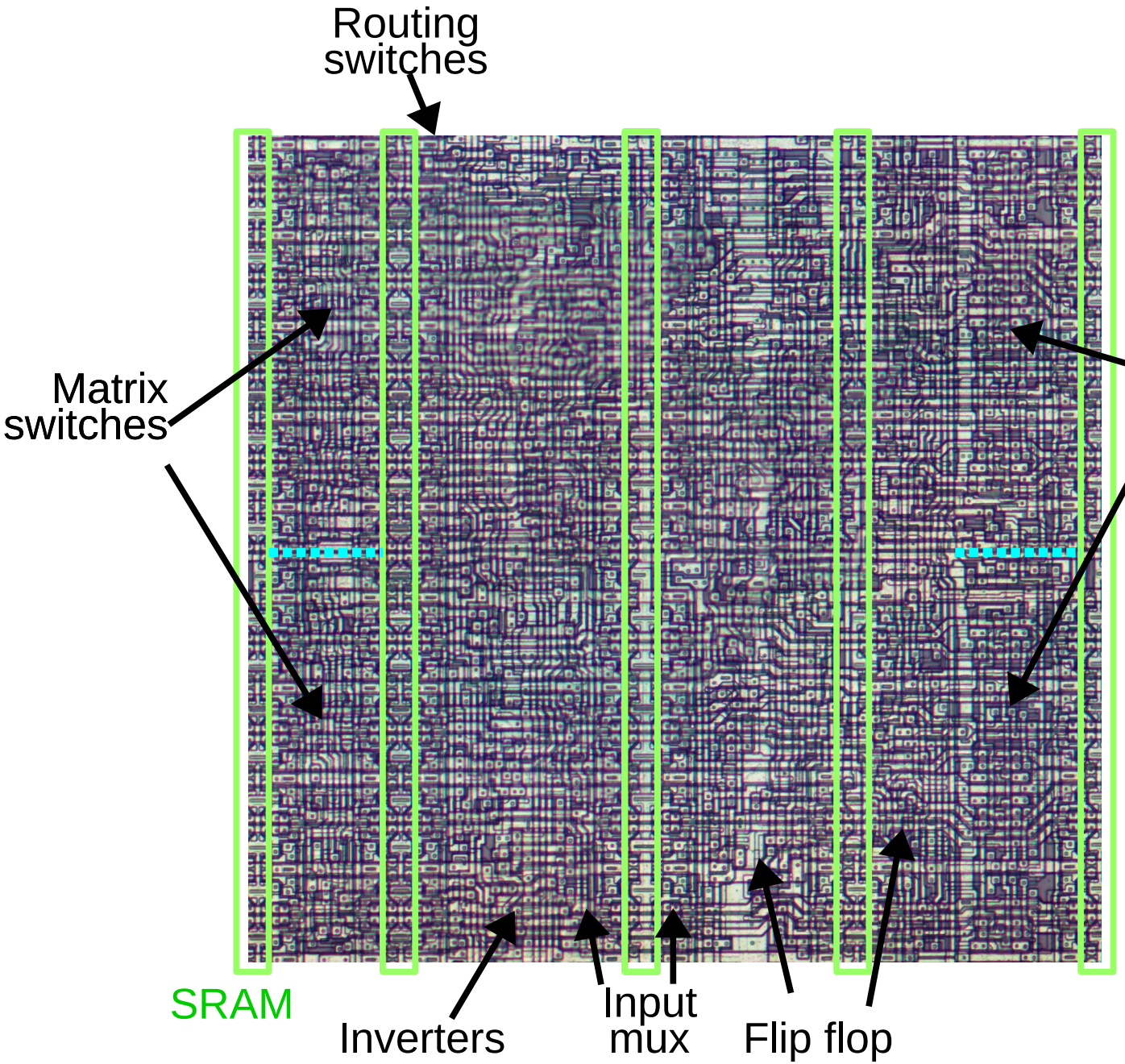


Latch / Flip Flop





CLB cell



Vertical:
interconnect
power
SRAM data

Horizontal:
interconnect
ground
SRAM row select

Conclusion

- FPGA built from pass transistors, muxes, SRAM, inverters, a few gates
- Bitstream makes sense looking at die layout
  - Highly-optimized, dense layout
- Made spreadsheet defining CLB bits

A	B	C	D	E	F	G	H
#2: 1-2	#2: 2-6	#2: 2-4	---	PIP A4, A3, A1, Afrom X = not A2,A5	---	Gin_3 = C	G = 1' 2' 3'
#2: 3-7	#2: 3-6	PIP D5, D4, D3	---	PIP A5, A4, A3	---	---	G = 1' 2 3'
#2: 2-7	#2: 2-8	ND 11	---	PIP A1, A4	---	---	G = 1 2 3'
#2: 1-5	#2: 3-5	PIP A3, AfromX	---	PIP D1, D4	Y=F	---	G = 1 2' 3
#2: 4-8	#2: 5-8	ND 10	---	PIP D3, DfromX	Y=G	Gin_2 = B	G = 1' 2' 3
#2: 7-8	#2: 6-8	ND 9	PIP B5, B2, B6,BfromX, BfromY	PIP Y2	X=G	Gin_1 = A	G = 1' 2 3
#2: 5-6	#2: 5-7	ND 8	PIP B1, B5, B2, B4, B6, B7, B8, BfromY (not B3,BX)	PIP Y4	X=F	---	G = 1 2 3
#2: 4-6	#2: 1-4	#2: 1-7	PIP C1, C3, C4, C7	PIP X3	Q = LATCH	---	Base FG (separate
#1: 3-5	#1: 5-8	#1: 2-8	PIP X2	---	---	---	---
#1: 3-4	#1: 2-4	ND 7	PIP C1, C2, C4, C5, C6, C7 (not C3,CX)	PIP X1	---	Fin_1 = A	F = ! 1 2 3
#1: 1-2	#1: 1-3	ND 6	PIP B6, B7	CLK = enabled	---	Fin_2 = B	F = 1' 2 3
#1: 1-5	#1: 1-4	ND 5	PIP C6, C7	CLK = inverted (FF), noninverted (LATCH)	---	---	F = 1' 2' 3
#1: 4-8	#1: 4-6	ND 4	PIP C4, C5	CLK = C	---	---	F = 1 2' 3
#1: 2-7	#1: 1-7	ND 3	PIP B4, B5	PIP K1	SET = F	---	F = 1 2 3'
#1: 2-6	#1: 3-6	ND 2	PIP B2, B8	PIP K2	SET = none	---	F = 1' 2 3'
#1: 7-8	#1: 3-7	ND 1	PIP C1, C2	PIP Y3	RES = D or G	Fin_3 = C	F = 1' 2' 3'
#1: 6-8	#1: 5-6	#1: 5-7	PIP B1, BfromY	PIP Y1	RES = G	Fin_3 = D	F = 1 2' 3'

Status

- Program to generate CLB from bitstream

```
Editblk BE
Base FG
Config X:Q Y:G F: G: Q:FF SET:F RES:D CLK:C:NOT
Endblk
Editblk AB
Base FG
Config X: Y: F:B:C:D G:A:C:Q Q: SET: RES: CLK:
Equate F = B+(C*D)
Equate G = A+(C@Q)
Endblk
Editblk HA
```

  - Annoying file format and special cases
  - Almost handles I/O pins (irregularities, more routing)
  - Need graph algorithm to process routing
- github.com/shirriff/xc2064
- Thanks to John!
  - siliconpr0n.org/map/xilinx/xc2064
  - github.com/JohnDMcMaster/project2064