NUMERO: 2419A ANNO: 2019

A P P U N T I

STUDENTE: Ricci Francesco

MATERIA: Microelectronics Systems - Prof. Mariagrazia Graziano

MICROELECTRONICS HISTORY

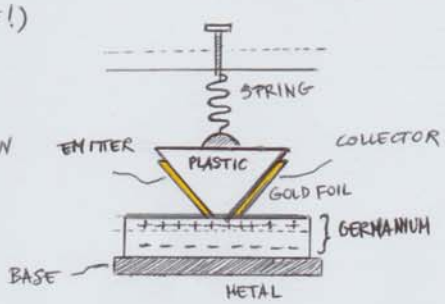
DIGITAL ELECTRONICS BORN WHEN RELAYS WERE SUBSTITUTED BY VACUUM TUBES. (1906)

1946: **ENIAC**: ELECTRICAL NUMERIC INTEGRATOR AND CALCULATOR.

COMPOSED BY: 17000 VACUUM TUBES, 7000 RESISTANCES, 10000 CAPACITORS, 6000 SWITCHES, 30 TONS, POWER CONSUMPTION OF 160 KW (HUGE!)

ABLE TO COMPUTE 5000 ADDITIONS IN 1 SECOND.

1949: BIPOLAR TRANSISTOR INVENTION: SHOCKLEY, BARDEEN, BRATTAIN (POINT-CONTACT TRANSISTOR)



1958: KILBY INTEGRATED CIRCUIT (TEXAS INSTRUMENTS)

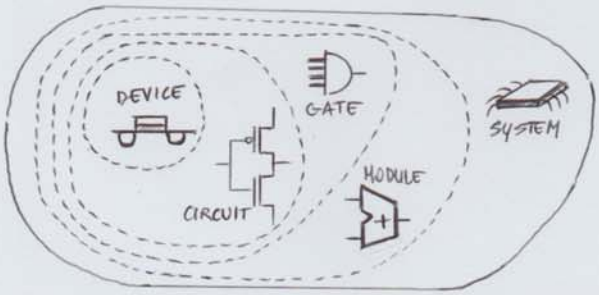
1971: 4004 INTEGRATED MICROPROCESSOR.

FIRST USAGE OF CMOS TECHNOLOGY (SILICON GATE): 10 μm NMOS LOGIC
COMPOSED BY 2300 TRANSISTOR, ONLY 3 mm × 4 mm (AREA)
2 METAL LAYERS, 4 BIT CPU, 100 KHz

i2HAC

2000: PENTIUM 4 INTEGRATED PROCESSOR
COMPOSED BY 54 MILION OF TRANSISTORS, 1.5 GHz, 0.18 μm TECH

(1965) MOORE'S LAW: INTEGRATION COMPLEXITY GROWS ESPONENTIALLY - THIS LAW SEEMS CONFIRMED UNTIL 2000....
WHY? WHAT DID ALLOW SUCH A PERFORMANCE IMPROVEMENT? KEYWORD IS **ABSTRACTION**



FIRST APPROACH: all circuit was done directly at the layout level from teams of designers which knew everything

ABSTRACTION: means **HIERARCHY**
SMALL BLOCKS ARE DEFINED CAREFULLY AT FULL CUSTOM LEVEL, AFTER CHARACTERIZED. PARAMETERS ARE DEFINED AND USED BY NEXT LEVEL TO BUILD BIGGER BLOCKS, ... AND SO ON....

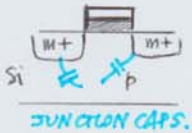
EACH LEVEL SOLVES IT'S OWN PROBLEMS!
HIGHER LEVELS JUST USE A LOWER LEVEL REPRESENTATION

TODAY THIS IS NOT ABSOLUTELY TRUE: STRINKING DOWN TRANSISTORS, NEW PROBLEMS OCCURS ACROSS MULTIPLE LEVELS... SO SOLUTIONS ON ONE LEVEL AFFECT ALSO OTHER LEVELS...

ITRS (international technology roadmap of semiconductor) make some predictions, like:

YEAR	2010	2013	2018
NODE [nm]	45	32	18
METAL LAYERS	15	16	18
MAX FREQ [MHz]	15079	22 980	53207
POWER [W]	120	138	158

SOI SILICON ON INSULATOR is a huge improvement in tech: AVOID CAPACITANCE = INCREASE SPEED WHILE REDUCING POWER
VERY FREQUENTLY USED TODAY!



JUNCTION CAPS.

NO JUNCTION CAPS

COMPLEX MICROELECTRONIC SYSTEMS

- **ASIC**: application specific integrated circuits ; composed by data memory, hierarchical finite state machine CU, arithmetic datapath with few dedicated registers.

- **ISA**: instruction set architectures ⇒ general purpose machines.
  composed by a data memory and an instruction memory.
  control unit update the program counter (PC): store the address of the next instruction that should be executed. That instruction is loaded into the IR (instruction register), used by the data path.

ISA CLASSIFICATION:
- L/S LOAD-STORE ARCH: all operands inside registers (RISC) ← REGULARITY OF EXECUTION, MANY REGISTERS
- R/M REGISTER/MEMORY ARCH: 1 operand may reside in memory (x86)
- R+M: all operands in memory + all operands in registers + all other combinations. (VAX)

RISC VS CISC: cisc processors use a variable length instructions: in order to identify the correct ~~access mode~~ addressing mode, more complex logic is necessary. Decoding multiple instructions at the same time means prediction on the starting point and ending point of each. Extra logic means extra area and an higher power consumption.

risc processors are based on two principles:
1. KEEP IT SIMPLE: a reduced instruction set is available because real programs spend 80% of the time executing 20% of possible type of instructions
2. L/S: only registers as operands means fixed length instructions and simple addressing modes.

Historically CISC give the possibility to build better processors because of
- poor silicon budget (maximum number of transistors: if use too many for registers, you don't have enought for ALU,....)
- high memory cost ⇒ because CISC allows usage of less number of instructions, total code is shorter.

Today RISC processors are clearly better, but binary compatibility stops them to spread all over the market. CISC nowadays implements a translation inside, to exploit RISC advantages, while maintaining binary compatibility.

SEQUENTIAL PROCESSOR ⇒ PIPELINED PROCESSOR ⇒ SUPERSCALAR PROCESSOR ⇒ VLIW (VERY LONG INSTRUCTION WORD) ⇒ EPIC (EXPLICITLY PARALLEL INSTRUCTION COMPUTING)

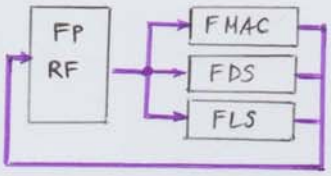
PIPELINED PROCESSOR: introduce possible data and control and structural hazards. Can be solved with:
- DATA HAZARDS: • OUT OF ORDER EXECUTION • FORWARDING • REGISTER RENAMING
- CONTROL HAZARDS: • INSTRUCTION QUEUES • BRANCH DELAY SLOTS • BRANCH PREDICTION
  - STATIC: predict branch not taken always / predict branch always taken / predict based on direction
  - DYNAMIC: predictors (1 BIT / 2 BIT), BHT branch history table, BTB branch target buffer.
- STRUCTURAL: • STALLS • MORE COMPLEX HW.

SUPERSCALAR PROCESSOR: execute more than one instruction per clock cycle. Fetch and decode handle more instructions, and perform checks on hazards @ RUN TIME. Usually perform out of order execution...

EPIC: PACK more instructions into a single instruction @ COMPILATION TIME
- VLIW: pack instructions that can be executed together.... if only one = WASTE OF SPACE = SIZE OF CODE INCREASE
- STALL in case of cache misses... PERFORMANCE LOSS
- LACK OF BINARY COMPATIBILITY: if hardware change, code should be recompiled.

SOLVED BY EPIC: use of extra field to express dependency between instruction inside the same pack and first instruction of next pack.

ARM VFP11 COPROCESSOR



- FMAC: floating point multiply and accumulate
- FDS: floating point division and square root
- FLS: floating point load and store

SPARSE TREE

Carry look ahead (CLA) adder is the basis for every tree adder:

a_i	b_i	c_i	s_i	C_{OUT}	p	g	
0	0	0	0	0	0	0	GENERATED
0	1	0	1	0	1	0	PROPAGATED
1	0	0	1	0	1	0	PROPAGATED
1	1	0	0	1	0	1	GENERATED
0	0	1	1	0	0	0	GENERATED [KILLED]
0	1	1	0	1	1	0	PROPAGATED
1	0	1	0	1	1	0	PROPAGATED
1	1	1	1	1	0	1	GENERATED

$s = a \oplus b \oplus c_{IN}$

$C_{OUT} = a \cdot b + a \cdot c + b \cdot c$

$p = a \oplus b$

$g = a \cdot b$

Let's do some math:

$s_1 = a_1 \oplus b_1 \oplus c_0 = p_1 \oplus c_0$

$c_1 = a_1 \cdot b_1 + a_1 \cdot c_0 + b_1 \cdot c_0 = a_1 \cdot b_1 + (a_1 \oplus b_1) c_0 = g_1 + p_1 \cdot c_0$

GENERATED OR PROPAGATED

NB: I can write XOR instead of OR because in this particular case the outcome is the same:

a	b	OR	XOR
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

WHEN THIS HAPPEN, THE FIRST TERM ($a_1 \cdot b_1$) IS EQUAL TO 1.
1 OR SOMETHING = 1, NO MATTER WHAT ($a_1 + b_1$) c_0 is EQUAL TO...

$s_2 = a_2 \oplus b_2 \oplus c_1 = p_2 \oplus c_1$

$c_2 = g_2 + p_2 \cdot c_1 = g_2 + p_2 g_1 + p_2 p_1 c_0$

SUBSTITUTE

$s_3 = p_3 \oplus c_2$

$c_3 = g_3 + p_3 c_2 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 c_0$

SUBSTITUTE

NOTES:

1. Virtually all carries can be calculated in function of c_0 (carry in).
2. However this require a very complex net, that is not easily scalable
3. It introduce a little congestion (not a regular structure) ⇒ HIGHER DELAY
4. Each stage drive a different amount of gates: sizing is more complex
5. Different size means different delay and a more complex design phase.
6. Different configurations are possible: maybe c_5 is written in function of c_1 instead of c_0 ...

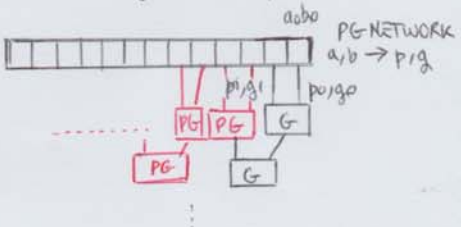
GENERAL PROPAGATE AND GENERAL GENERATE

By modifing values of i, j and k, the network can be constructed and carries can be computed.

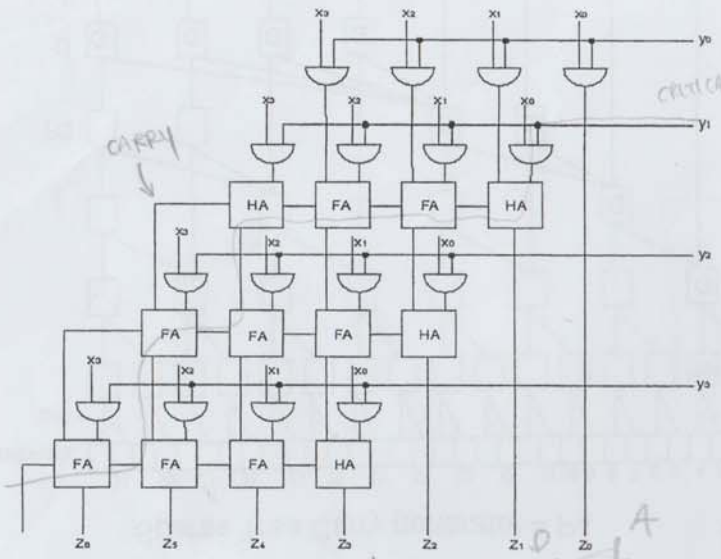
$P_{i:j} = P_{i:k} \cdot P_{k-1:j}$

$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1:j}$

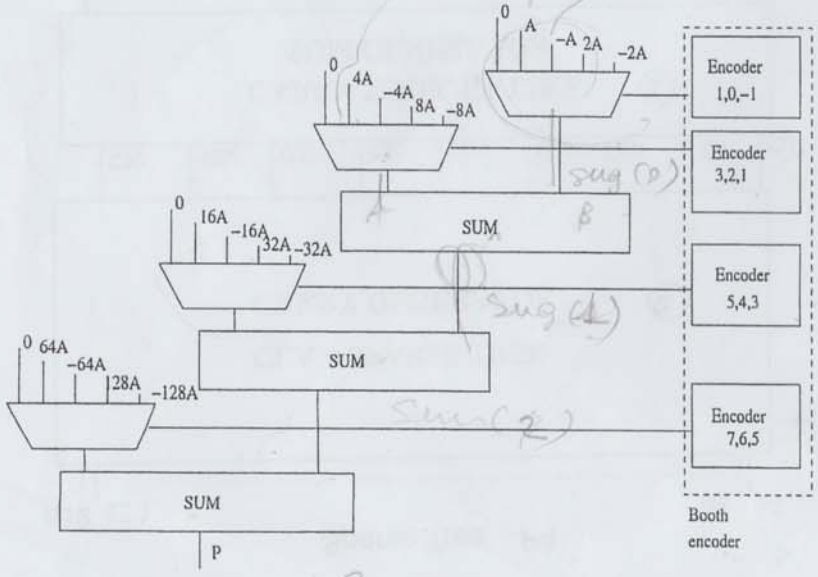
$i >= k > j$ $\quad G_{xx} = g_x$ $\quad P_{xx} = p_x$ $\quad g_0 = c_{in}$ $\quad p_0 = 0$

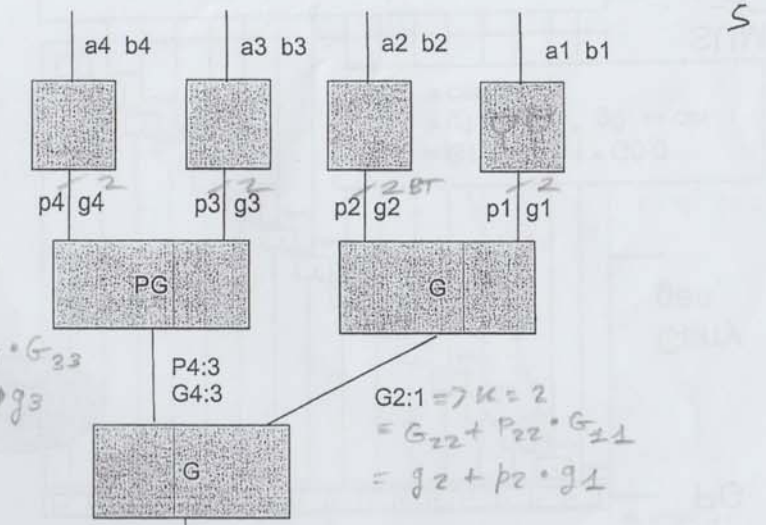


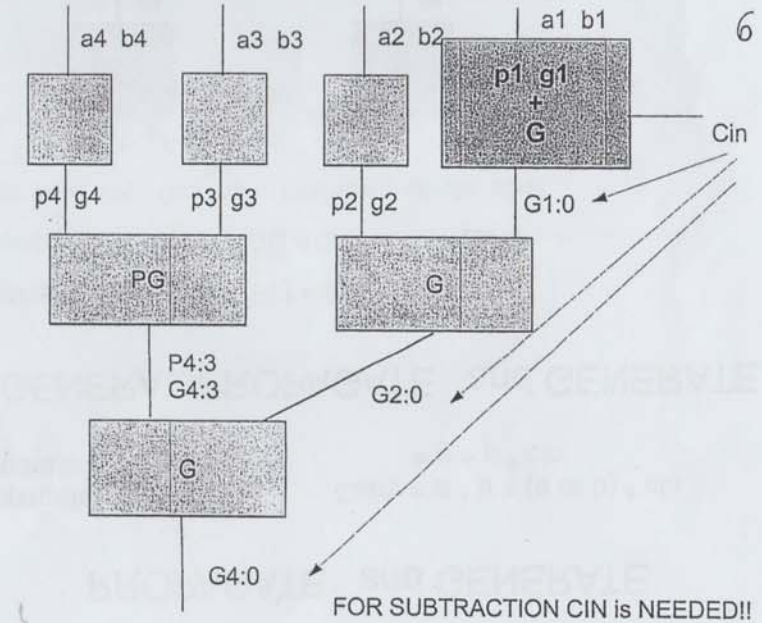
Array multiplier



Booth's multiplier







BOOTH'S ALGORITHM

Suppose M is even; by convention $B_{-1} = 0$. NB: M is the parallelism of multiplicand B

algorithm:

```
i = 0;
P = 0;
WHILE (i ≤ M-2) LOOP
    P ⇐ P + V_p(B_{i+1}, B_i, B_{i-1});
    A ⇐ 4*A;
    i ⇐ i+2;
END LOOP;
```

y_{i+1}	y_i	y_{i-1}	V_p
0	0	0	0
0	0	1	+A
0	1	0	+A
0	1	1	2A
1	0	0	-2A
1	0	1	-A
1	1	0	-A
1	1	1	0

A: MULTIPLIER

Example: $B = \overset{7\ 6\ 5\ 4\ 3\ 2\ 1\ 0\ -1}{0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0}$

(i=0) $V_p(B_1, B_0, B_{-1}) = V_p(000) = 0$

$P \Leftarrow P + 0 = 0$

$A \Leftarrow 4A = 2^2 A$

$i \Leftarrow i+2 = 2$

(i=2) $V_p(B_3, B_2, B_1) = V_p(100) = -2A$

$P \Leftarrow P + (-2A) = -2^3 A$

$A \Leftarrow A4 = 2^4 A$

$i \Leftarrow i+2 = 4$

(i=4) $V_p(B_5, B_4, B_3) = V_p(011) = 2A$

$P \Leftarrow P + (2A) = -2^3 A + 2 \cdot 2^4 A = 2^5 A - 2^3 A$

$A \Leftarrow 4A = 2^6 A$

$i \Leftarrow i+2 = 6$

(i=6) $V_p(B_7, B_6, B_5) = V_p(000) = 0$

$P \Leftarrow P + 0 = \boxed{P = 2^5 A - 2^3 A}$

RIGHT!

Key concept: any by many number can be written exploiting additions or partial shift + subtractions:

EX: $B = 00011000 = 2^3 + 2^4 = 48$

$= 2^5 - 2^3 = 48$

EX: $B = 01011000 = 2^7 - 2^6 + 2^5 - 2^3 = 176$

WHAT HAPPENS IF $B = 1001100$ (1 IN MSB...)?

In this case only last iteration changes:

(i=6) $V_p(B_7, B_6, B_5) = V_p(100) = -2A$

$P \Leftarrow P + (-2A) = -2^7 A + 2^5 A - 2^3 A = -104A$

$= -104$

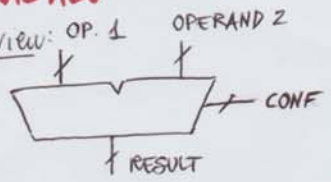
MULTIPLICATION STILL WORKS, BUT ONLY IF YOU CONSIDER NEGATIVE (2'COMPL.) THE MULTIPLICAND.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 114

27 MARZO
PT2
REC CELL

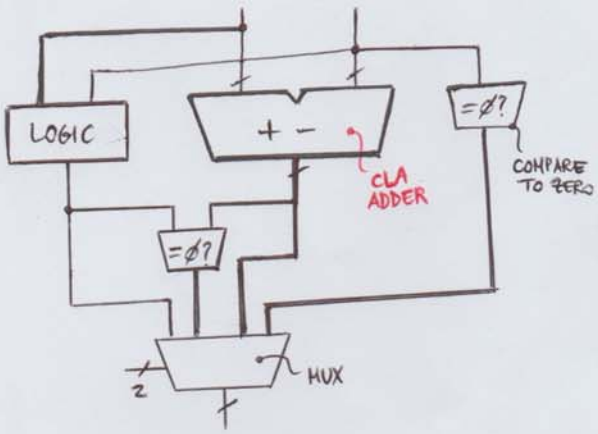
GENERAL ALU

High level view:



- ADD, SUB
- MUL, DIV
- LOGIC
- SHIFT, ROT
- COMPARISON (= != >> << <=)
- MODULOS, SQUARE, ROOT

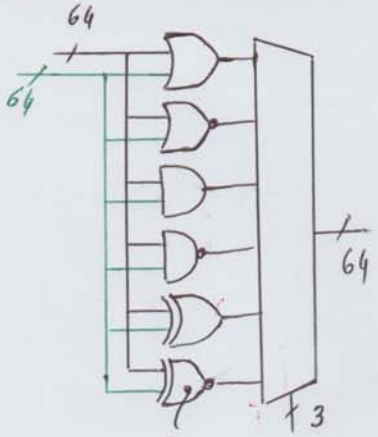
ALU OF T2



In T2 processor, the alu is separeted from the shifter unit

Logic unit : it is the last piece missing from the above circuit.

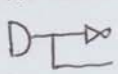
Obvious implementation:



they are ~~tree~~ of gates, or an array: in the second case I use 64x6 gates + a huge mux

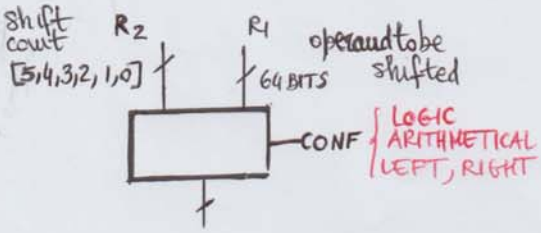
This is a very bad solution!

1) Mux has a tone of logic! ≤ AREA DELAY
2) Every gate has a different delay, for this reason I need to consider the slowest port (NAND is the fastest)
3) (XOR is slowest)

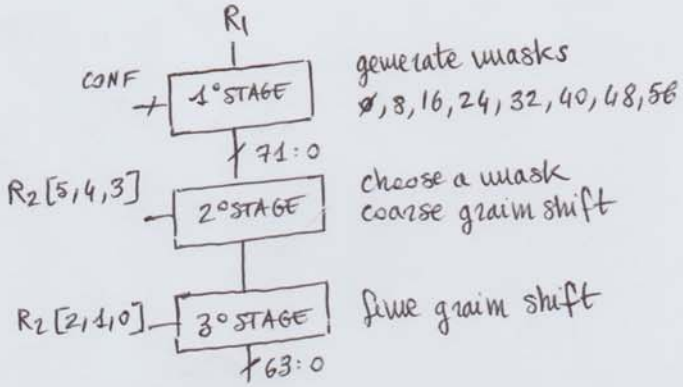
Why reducing the number of transistor using  is a bad solution? Because the and gate has to drive a lot (twice) the electronics! And for this reason It must be ~~longer~~!

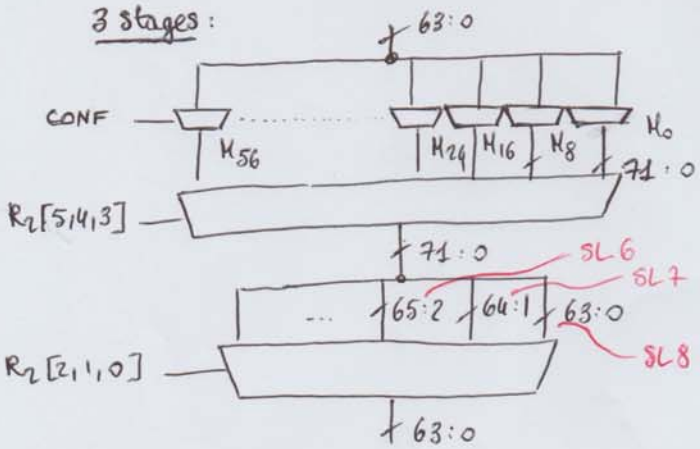
Oral question: implement this solution and discuss why it is bad.

T2 SHIFTER.



Composed of 3 stages:



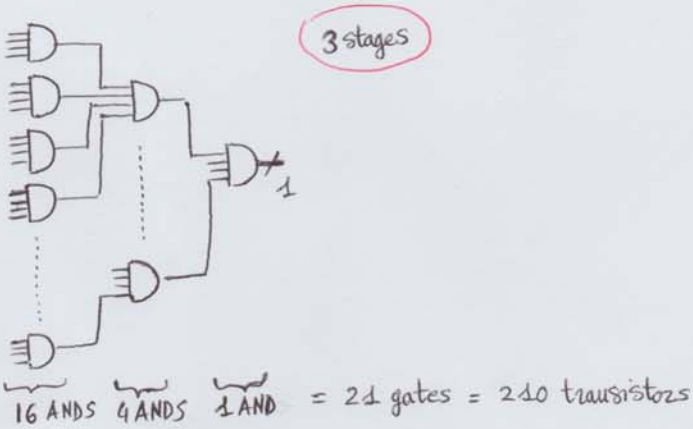


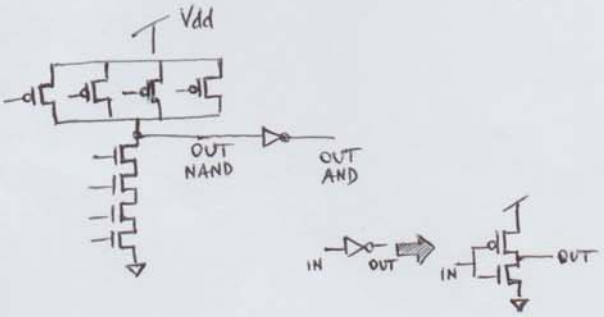
Correct implementation:

A good implementation is a multilevel (tree) one.

High level view:

using an AND gate with 4 inputs

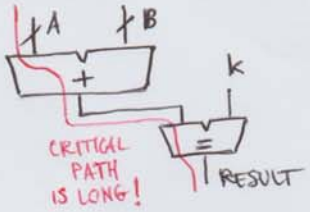




EVEN MORE COMPLEX COMPARATOR

Now I want to perform $A+B=k$ operation. Result = 1 if $A+B=k$

Obvious implementation



A big disadvantage is the delay due to carry propagation inside the adder.

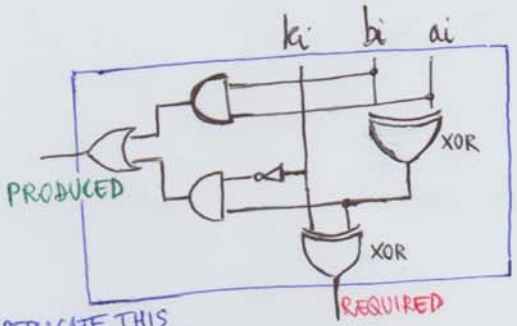
Better implementation

a_i	b_i	k_i	REQUIRED CARRY IN C_{i-1}	PRODUCED CARRY OUT C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

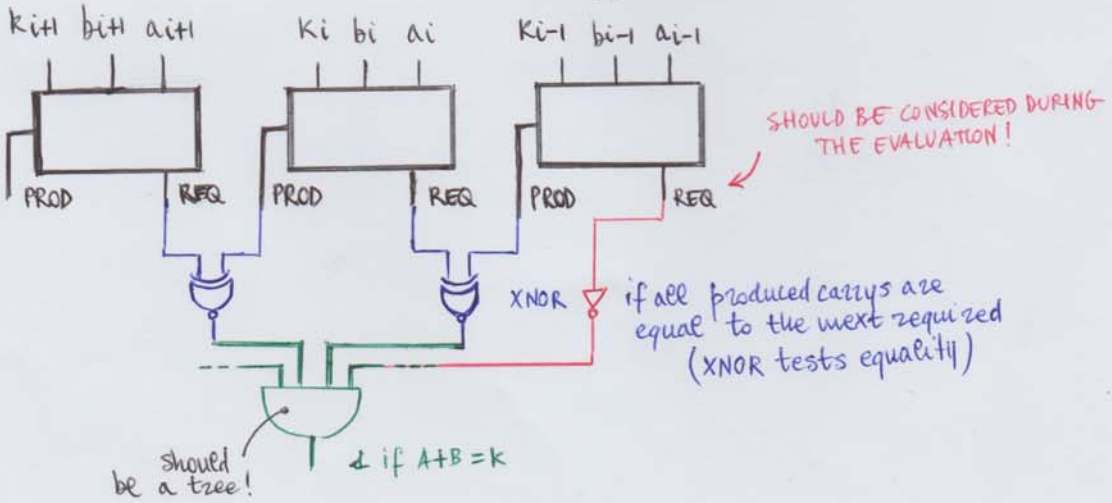
We don't demostrate it, but we can write:

$C_{i-1} = a_i \oplus b_i \oplus k_i$ REQUIRED

$C_i = (a_i \oplus b_i) \cdot \bar{k_i} + a_i \cdot b_i$ PRODUCED



REPLICATE THIS BLOCK FOR EVERY BIT OF THE TWO OPERANDS

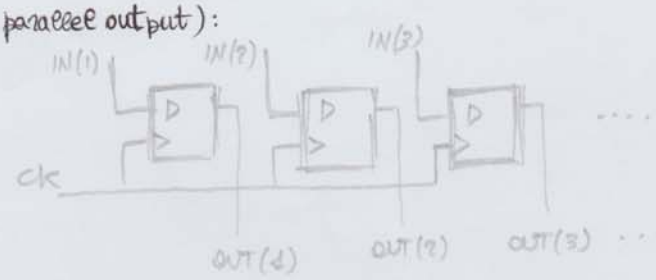


Advantage: I process every bit in parallel. A ripple carry adder process each bit in a sequential way.

REGISTERS

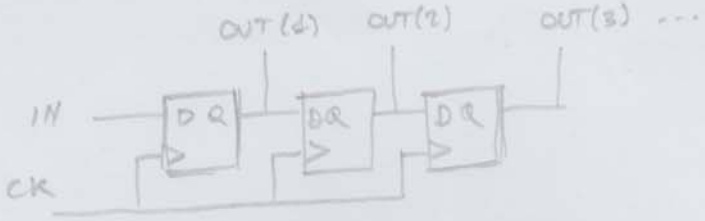
Based on how input and outputs are connected, we can classify registers as 4 groups:

- PIPO (parallel input, parallel output):

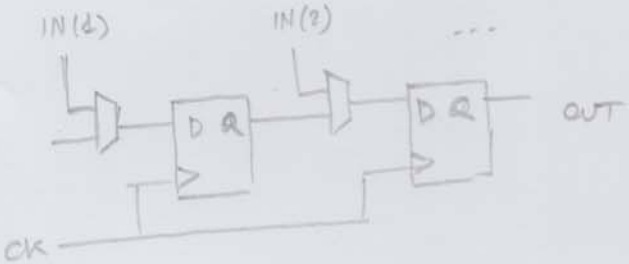


- SISO (serial input, serial output):

- SIPO:



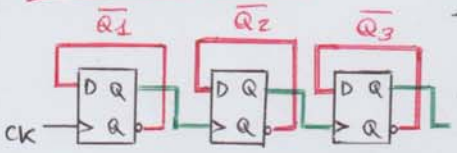
- PISO:



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 114

COUNTERS → ASYNCHRONOUS, SYNCRONOUS, JOHNSON, HA COUNTER

ASYNCHRONOUS COUNTER

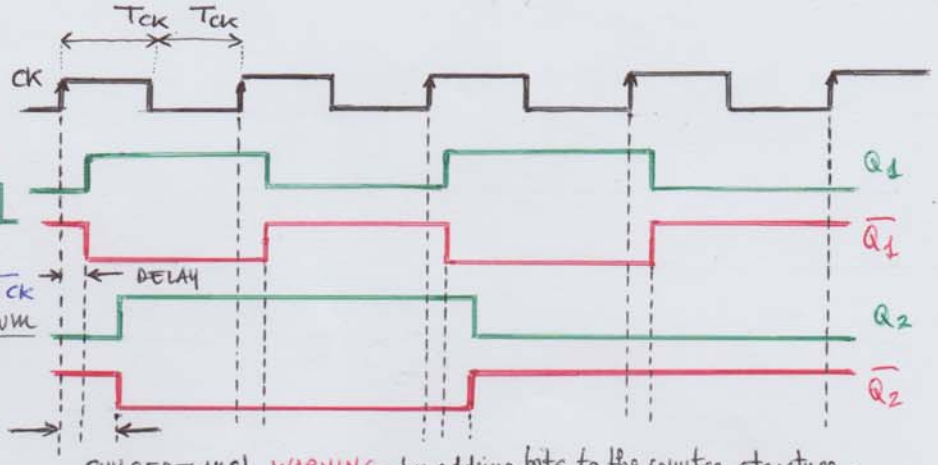


if use in parallel $Q_1 Q_2 Q_3$ $\uparrow$CK you obtain an asyncronous down counter!

BEWARE OF DELAYS!

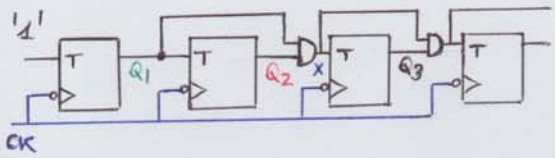
$T_{Q1} = 2T_{CK}$

$T_{Q2} = 2T_{Q1} = 4T_{CK}$



SUM OF DELAYS! WARNING: by adding bits to the counter structure, the delay sum at the end will increase. Maybe, if the clock period is too short, there will be a setup time violation! ⇒ NOT EASY TO SCALE ⇒ EASY TO BUILD

SYNCRONOUS COUNTER



SEARCH THE CRITICAL PATH: a path should always start from the out of FF to the IN of another FF (or the same one, if there are paths).

Supposing $t_{AND} = 3ms$, $t^{FF}_{CK \to Q} = 5ms$, paths are:

$t_{p1} = 5ms$

$t_{p2} = 5ms + 3ms = 8ms$

$t_{p3} = 5ms + 3ms = 8ms$

$t_{p4} = 5ms + 3ms = 8ms$

$t_{p5} = 5ms + 3ms + 3ms = 11ms$ CRITICAL PATHS!

$t_{p6} = 5ms + 3ms + 3ms = 11ms$

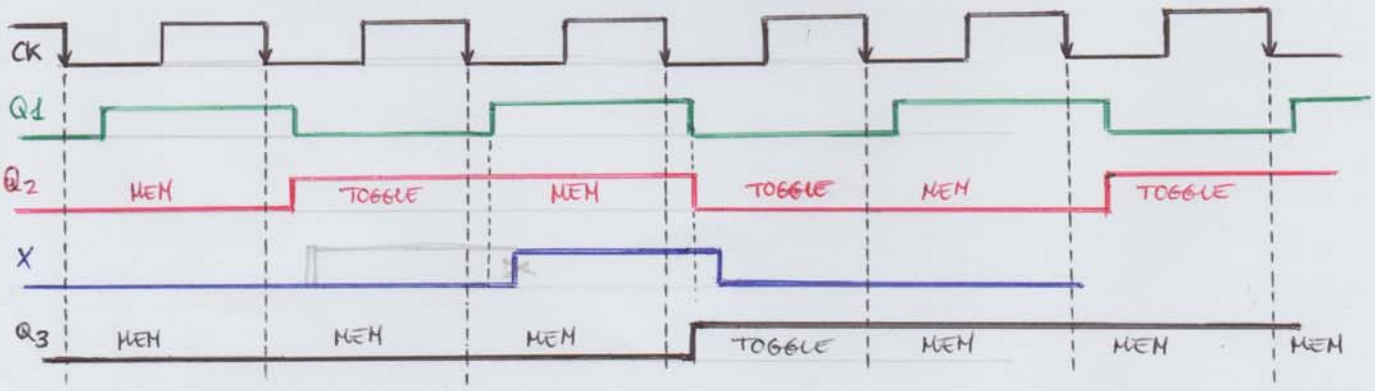
In general:

p_1 = delay logic 1

$p_2 = t_{CK \to Q}$ + delay logic 2

$p_3 = t_{CK \to Q}$ + delay logic 3

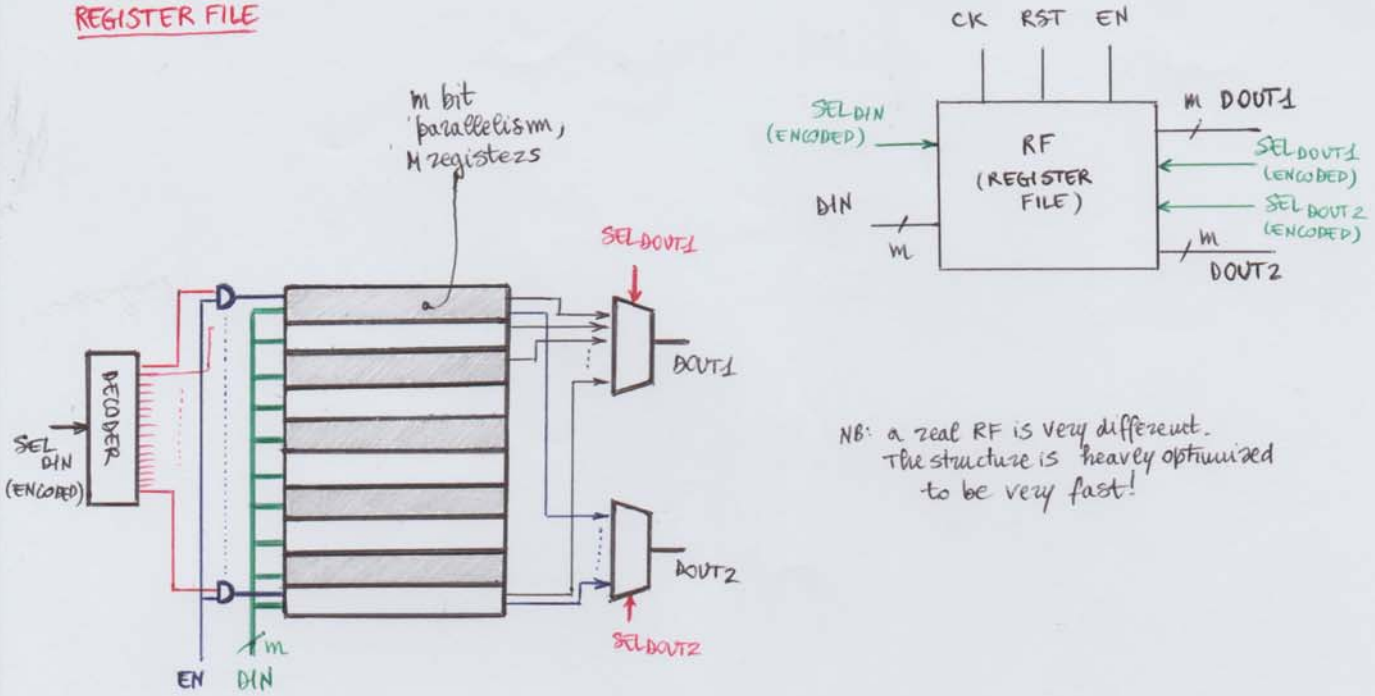
$$MAX\_delay = MAX\{p_1, p_2, p_3\} \Rightarrow MIN\_freq = \frac{1}{MAX\_delay + SETUPTIME}$$



000 ..... 001 ..... 010 ..... 011 ..... 100 ..... 101 110

Q_3 Q_2 Q_1

REGISTER FILE



NB: a real RF is very different. The structure is heavily optimized to be very fast!

NOTES:

1. Conflict between input and output path! When is requested to read and write the same register, which is the order to follow? Is important to define a policy! ⇒ WAR or RAW
   Another solution could be a double data rate ⇒ half clock cycle read, half clock cycle write.

2. Decoder recall: it is a combinational logic circuit that converts binary information from n coded inputs to a maximum of 2^n unique outputs.

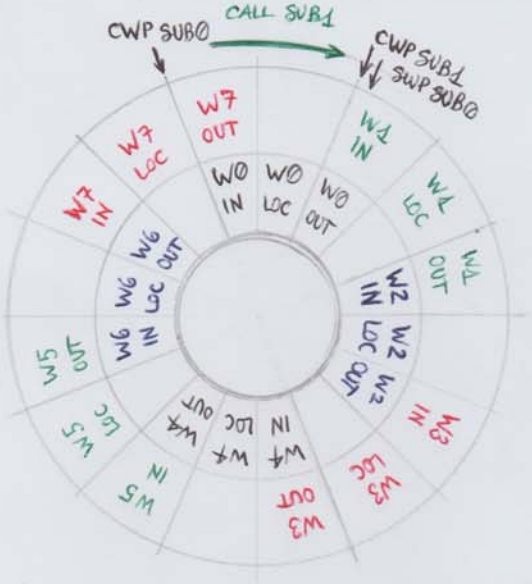
3. Problem of context switching: if a procedure is called (by using assembler instruction CALL/RET) two actions should be performed:
   a) move all registers from RF to the stack ⇒ CALL or move all registers from stack to the RF ⇒ RET
   b) save in the stack the value of flags ⇒ CALL or restore from stack the value of flags ⇒ RET

NB: context switching creates a long "bubble" in the pipeline. Often a procedure requires few registers to accomplish the task, but every time CALL/RET are used, the entire RF should be copied to the stack. A solution is windowing.

WINDOWING → FIXED: size of windows are fixed
WINDOWING → FLEXIBLE: size of windows variable

Instead of moving the entire RF content every time, I just move a small portion of it (a window).

REGISTER FILE FULL: Im case the RF is full (all windows are occupied) and a CALL arrive, a **SPILL** should be performed. A spill moves the oldest window im the stack, im order to free space for the new procedure.
When a RET arrive, if one or more SPILL were performed, a **FILL** should be done, im order to restore the replaced window from the stack.



Im order to understand when perform a SPILL / FILL we use 2 pointers:

- CWP: current window pointer
- SWP: saved window pointer.

SUB0:
operations
⋮
CALL SUB1 → SHIFT CWP, SHIFT SWP IS NOT NEEDED! ONLY AFTER A SPILL!

SUB1
operations
⋮
CALL SUB2 → SHIFT CWP, SHIFT SWP IS NOT NEEDED!

⋮

SUB6
operations
⋮
CALL SUB7 ⇐ FREE SPACE IS NOT ENOUGH! NEED A SPILL

SPILL W0 IN and W0 LOC im STACK (or MEMORY)

SHIFT CWP TO SUB7 ⇒ imo RF use W7 IN, W7 LOC, W7 OUT

SHIFT SWP TO REMEMBER THAT W0 IN and W0 LOC ARE IN MEMORY

SUB7
operations
⋮
CALL SUB8 ⇦ NO FREE SPACE! SPILL REQUESTED

SPILL W1 IN and W1 LOC IN MEMORY

SHIFT CWP TO SUB8 (OLD SUB0)

SHIFT SWP TO SWP1

SUB8:
operations
⋮
RET FROM SUB8 (TO SUB7)

SHIFT BACK CWP, DON'T TOUCH SWP!

SUB7
⋮
RET

SUB6
⋮

SUB3
⋮
RET FROM SUB3

SHIFT BACK CWP.

CHECK: SWP = CWP? YES, BECAUSE CWP2 ≡ SWP1
↳ IF YES NOTHING TO DO NOW.

SUB2
⋮
RET FROM SUB2

⚠ SWP1 ≫ CWP1 ⇐ perform a FILL!

FILL FROM STACK W1 IN and W1 LOC

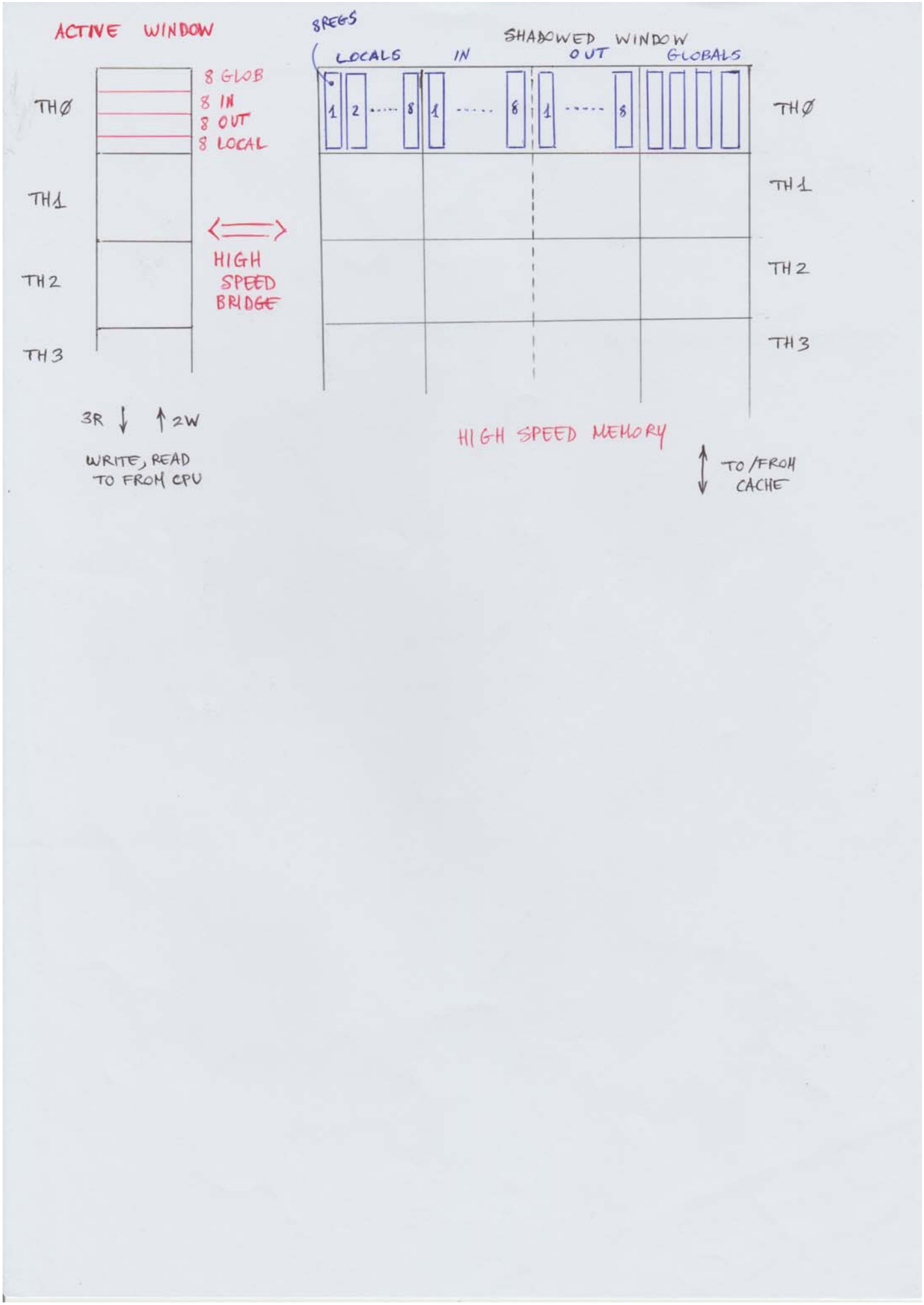
SHIFT BACK CWP TO CWP1

SHIFT BACK SWP TO SWP0

⋮

SUMMARY: I need those operations:

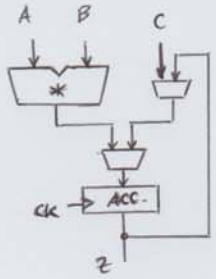
1) SHIFT BACK AND FORTH CWP & SWP
2) COMPARISON BETWEEN CWP & SWP
3) SPILL / FILL ⇒ COMUNICATE WITH A **MMU** (memory management unit)!



ARM VFP11 COPROCESSOR

As already said before, it is composed by:
- LOAD AND STORE UNIT
- F.P. MULTIPLY AND ACCUMULATE
- F.P. DIVIDE AND SQUARE ROOT

MULTIPLY AND ACCUMULATE: working principle:



$$z_0 = A_0 \times B_0$$
$$z_1 = z_0 + A_1 \times B_1$$
$$\vdots$$
$$z_{i+1} = \sum_{i=0}^{m} A_i \times B_i$$

DIVIDER: ARM COPROCESSOR IMPLEMENTS A RADIX4 SRT ALGORITHM.
HOWEVER LET'S TAKE SOME STEPS TO UNDERSTAND THAT ALGORITHM:

A) BASIC PRINCIPLE: ITERATIVE (THEORY, NO HW IMPL. POSSIBLE)
B) RESTORING ALGO
C) NON RESTORING + REDUNDANT ALGO
D) SRT
E) RADIX-4
F) ARM IMPLEMENTATION...

(A) BASIC: **ITERATIVE**

DIVIDEND $z = 14_{10} = 1110_2$
DIVISOR $D = 3_{10} = 11_2$

BASE 10

```
14 | 3
 0 | 04
---
14
12
---
 2
```

PARTIAL REMINDER — QUOTIENT — REMINDER

$$\frac{z}{D} = 4 + \frac{2}{D}10^0$$

BASE 2

```
1110 | 11
  11 | 100
-----
 001
  00
-----
  10
  00
-----
  10
```

P.R. — QUOTIENT — P.R. — REMINDER

$$\frac{z_2}{D_2} = 100 + \frac{10}{D_2}2^0$$

ITERATIVE ALGO: GUESS Q.DIGIT (REALLY HARD IN HW), MULTIPLY, FIND P.R., GO BACK TO POINT 1. (EASY IN HW)

FORMALLY THE ALGORITHM IS: $R^{j+1} = rR^j - q_{j+1} \cdot D$

$$\boxed{R^{j+1} = 2 \cdot R^j - q_{j+1} \cdot D}$$

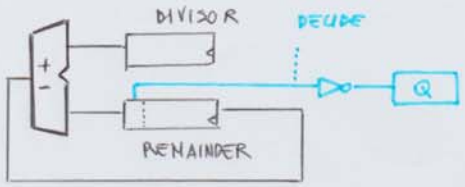
BINARY ALGORITHM.

WHERE
- R^j PARTIAL REMAINDER ($R^0 = z$ = DIVIDEND)
- D DIVISOR
- q_j QUOTIENT DIGIT
- r # DIGIT OF THE ALPHABETH (BINARY $r = 2$)

TO SIMPLIFY ASSUME:

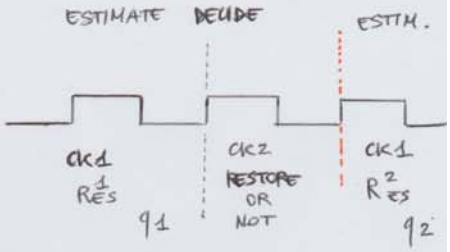
z, D POSITIVE AND NORMALIZED AND $z < D$ (OBTAIN SO $0.\ldots$)

$z = R^0 \rightarrow .r_1^0 r_2^0 r_3^0 r_4^0 \ldots r_m^0$
$D \rightarrow .d_1^0 d_2^0 d_3^0 \ldots d_m^0$
$Q \rightarrow .q_1 q_2 \ldots q_m$



if SIGN 0 → $q_i = 1$
MEANS $R_{ES} > 0$

if SIGN 1 → $q_i = 0$
MEANS $R_{ES} < 0$
⇓
RESTORE BY SUMMING D!



Let's verify: A : 0.01010110 = R^0 = z
B : 0.1100 = D ⇒ $\overline{D}$ 1.0011 + 1 = 1.0100

You have to subtract D, that in electronics means sum the 2' complement $\overline{D}$

$2R^0$ = 0.1010110 +
−D 1.0100
R^1_{ES} 1.1110110

DECIDE → IT'S NEGATIVE! → RESTORE → CORRECT q

ASSUME $q_1 = 1$

$R^1 = 2R^0$ AND NOT $2R^0 - D$...

$q_1 = 0$

$2R^1$ = 1.0101100 +
−D 1.0100
R^2_{ES} 0.1001100

ASSUME $q_2 = 1$

DECIDE → IT'S POSITIVE! → NO RESTORE → q IS OK

$R^2 = R^2_{ES} = 2R^1 - D$

$q_2 = 1$

$2R^2$ = 1.0011100 +
−D 1.0100
R^3_{ES} 0.0111100

ASSUME $q_3 = 1$

DECIDE → IT'S POSITIVE! → NO RESTORE → q IS OK

$R^3 = R^3_{ES} = 2R^2 - D$

$q_3 = 1$

$2R^3$ = 0.1111000 +
−D 1.0100
R^4_{ES} 0.0001000

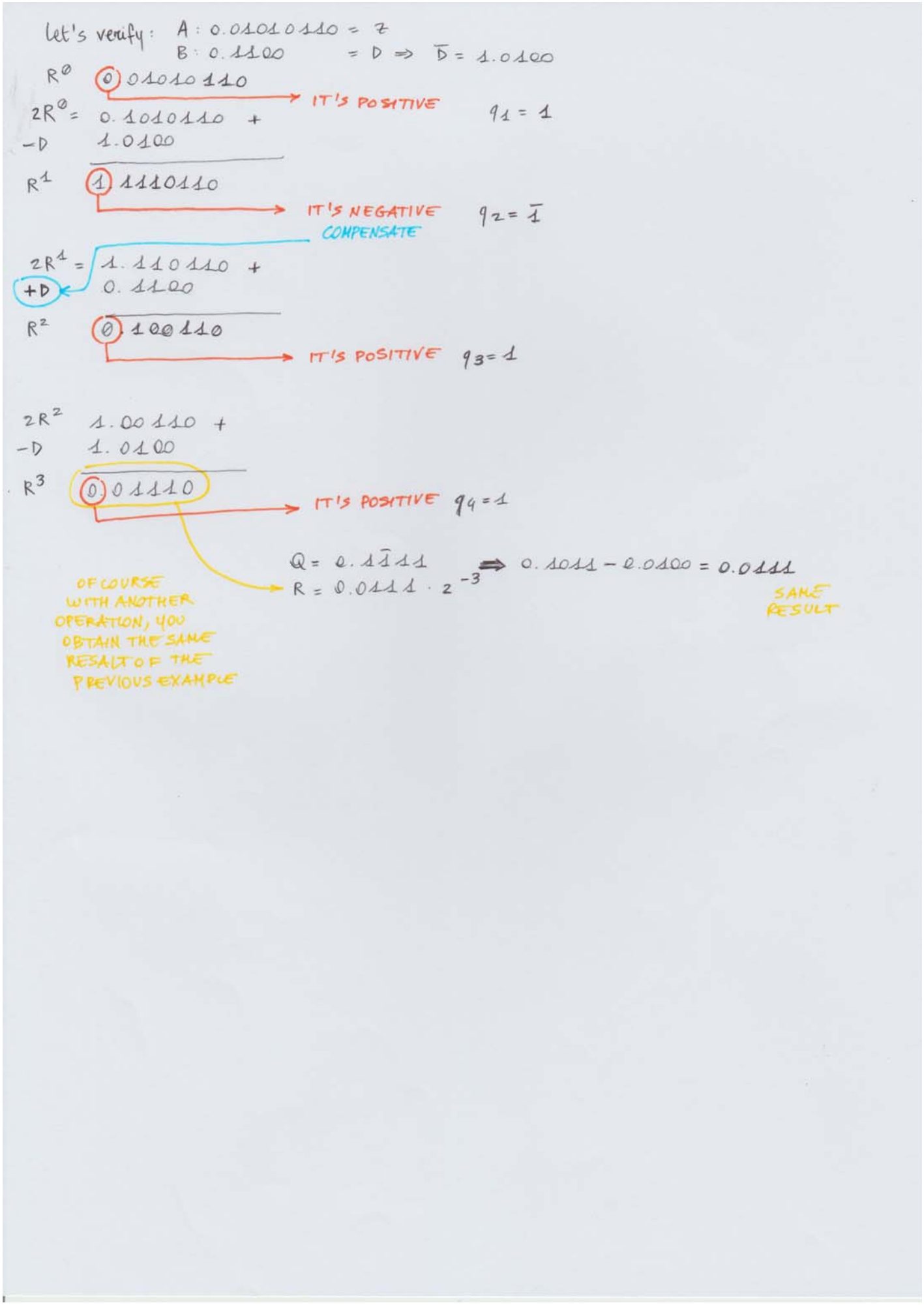
ASSUME $q_4 = 1$

DECIDE → IT'S POSITIVE! → NO RESTORE → q IS OK

$R^4 = R^4_{ES}$

$q_4 = 1$

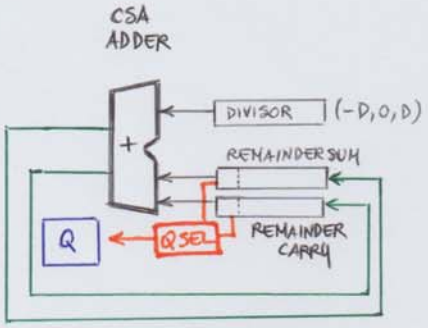
Q = 0.0111
R = 0.0010 · 2^{-4} (4) # ITERATIONS



Basic CSA algorithm:

A: 0110
B: 0111 → A+B = 0110 + 0111 = 1101

LET'S CALL IT CARRY VECTOR. THE PRINCIPLE IS SAVE CARRY BEFORE RUN TIME, KEEPING THE FINAL SUM THAT REQUIRES CARRY PROPAGATION ONLY AT THE END.



(E) RADIX 4

Operands on m bits

$r = 2^b$ (RADIX) $\quad K = \frac{m}{b}$ WHERE Q comprises (?)
K is radix
r are digits

GENERAL NAME: RADIX-r ALGORITHM, NEEDS K ITERATION TO COMPLETE

IN OUR CASE RADIX-4: $r = 2^2$, $K = \frac{m}{2}$

RADIX-8: faster but much more complex

C.U. IMPLEMENTATION

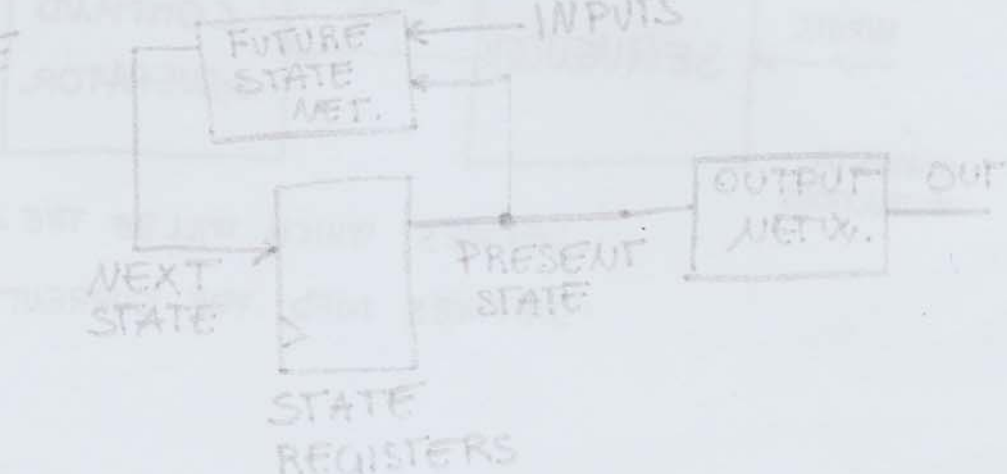
1) CLASSIC F.S.M. (ENCODED STATES)

2) HARDWIRED — A FSM (NOW ENCODED STATES) WHERE THE SEQUENCER IS A COUNTER

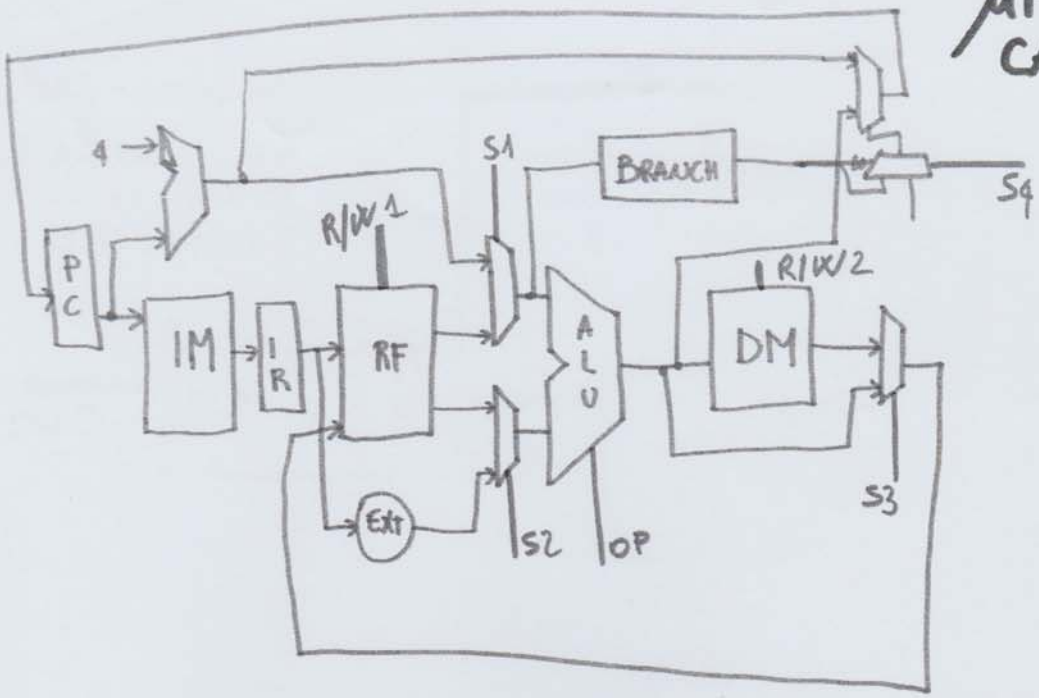
3) MICROPROGRAMMED

A FLASH BACK TO F.S.M.

2 TYPES: { MOORE, MEALY }



µP CASE

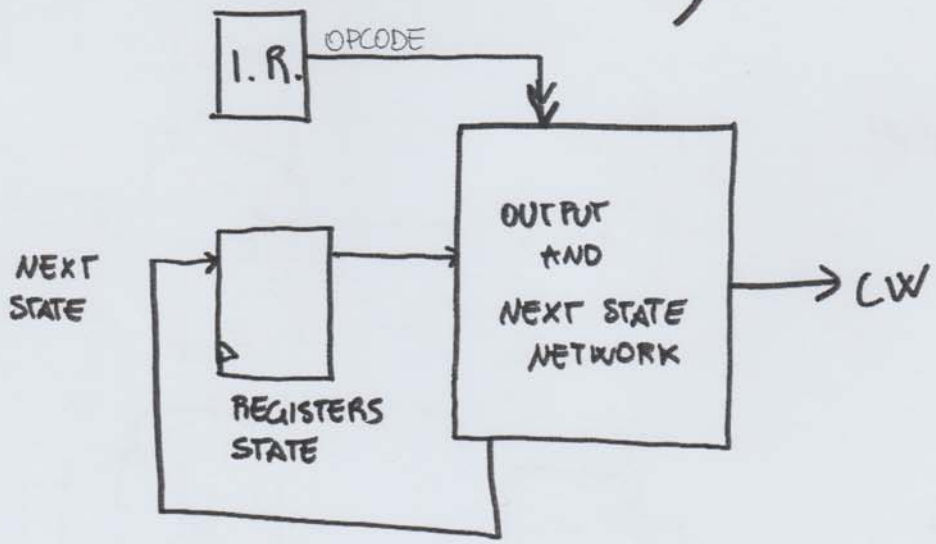


DLX CONTROL SIGNALS → CW
CONTROL WORD

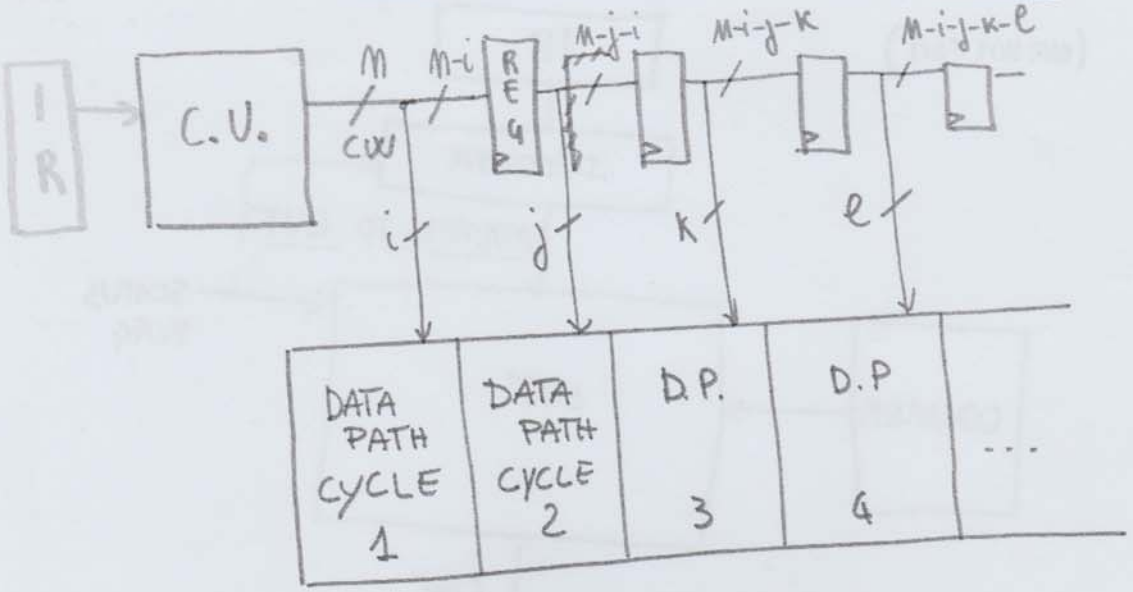
CW 01100101100.....
OP RW2
S1 S2 RW1 S3 S4

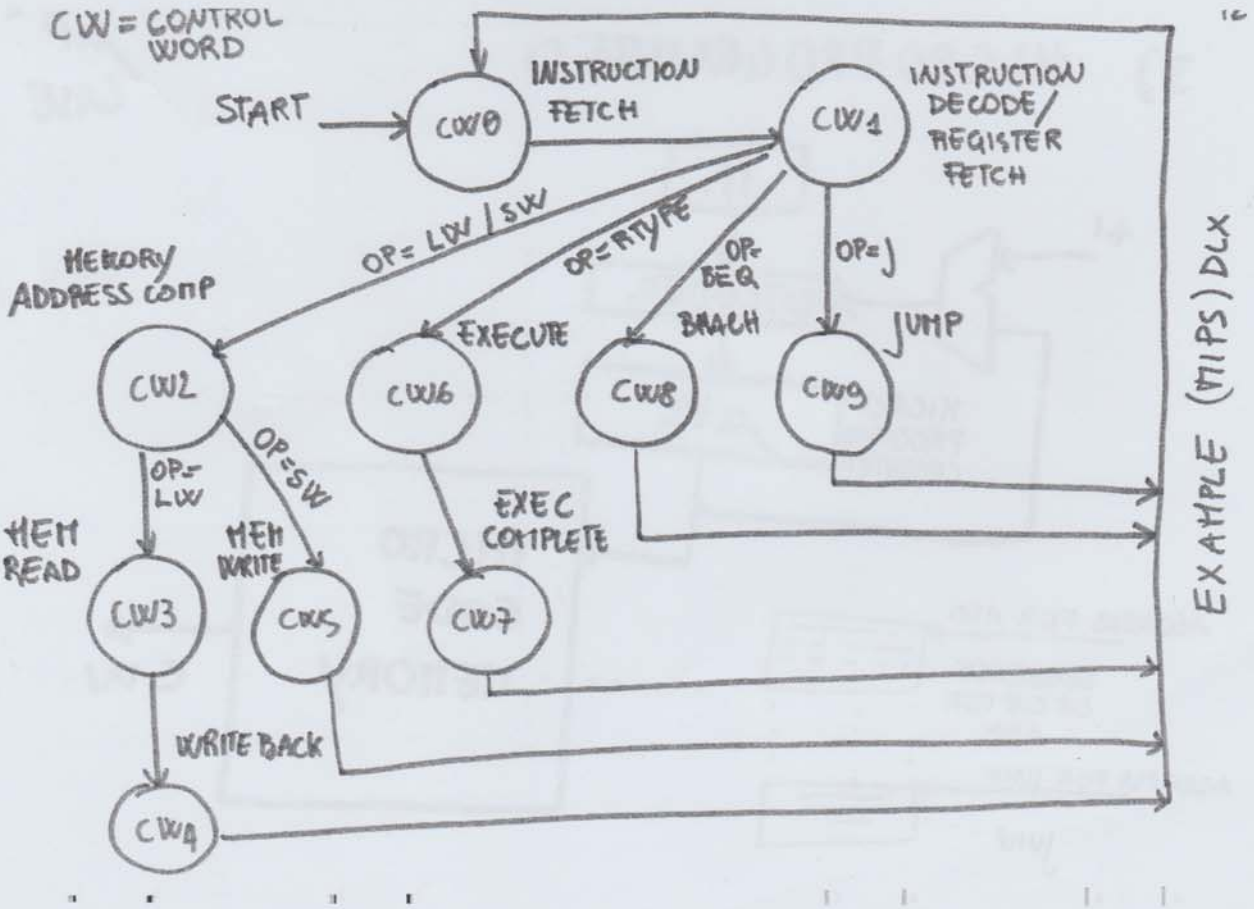
µP CASE

1) FSM



WHAT IF PIPELINED?



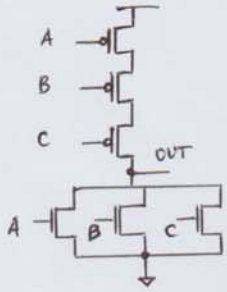


Homework: using CMOS logic family, implement the following logic functions:

① $OUT = \overline{A+B+C}$ ② $OUT = (A+B)(C+D)$ ③ $OUT = A+(B+\overline{C}D)$

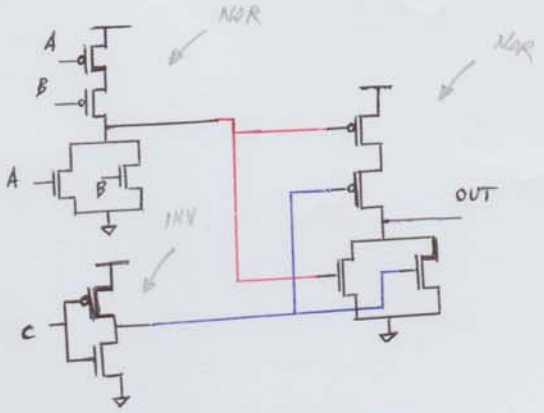
by using a single stage and then a cascade of stages (each one with 2 inputs) at most.

① $OUT = \overline{A+B+C}$ it is a 3input NOR



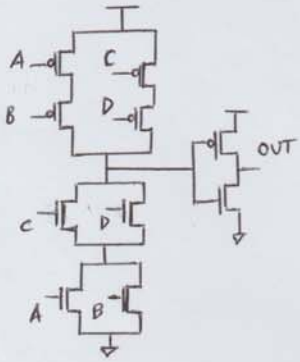
Using de Morgan I can manipulate the expression:

$$\overline{A+B+C} = \overline{A}\cdot\overline{B}\cdot\overline{C} = \overline{A+B}\cdot\overline{C} = \overline{\overline{A+B}+\overline{C}}$$

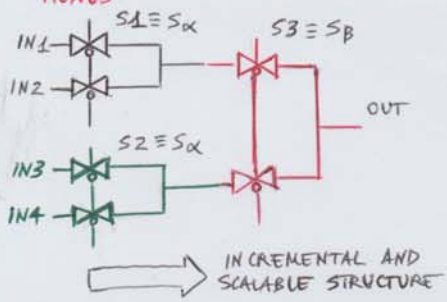


② $OUT = (A+B)\cdot(C+D)$ using deMorgan: $\overline{\overline{A+B}}\cdot\overline{\overline{C+D}} = \overline{\overline{A+B}+\overline{C+D}}$

Since CMOS is inverting, I need an inverter at the end!



MUXES:



OPTIMIZATION: we can reduce the number of selection signals:

$S1 = S2 = S_\alpha$

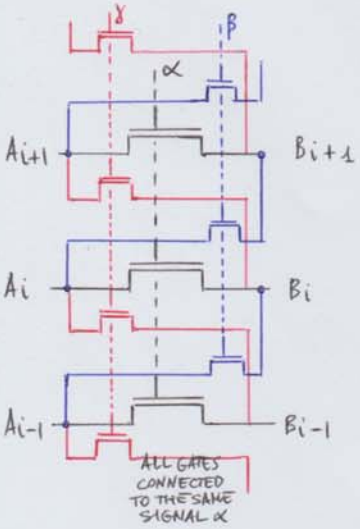
$S3 = S_\beta$

IMPORTANT: S_α and S_β should be inverted and connected the the other side of TG. The inversion is characterized by a small delay! Here isn't so important, but in LATCHS AND FFs based on TG it matters a lot!

SHIFTERS: three different implementations will be proposed:

- BINARY SHIFTER
- BARREL SHIFTER
- LOGARITHMIC SHIFTER

BINARY SHIFTER:



NB: to symplify the drawing, TG are reported as  .

IF $\alpha = 1$ AND $\beta = 0$ AND $\gamma = 0$ $\Rightarrow B_i = A_i \Rightarrow$ NOP

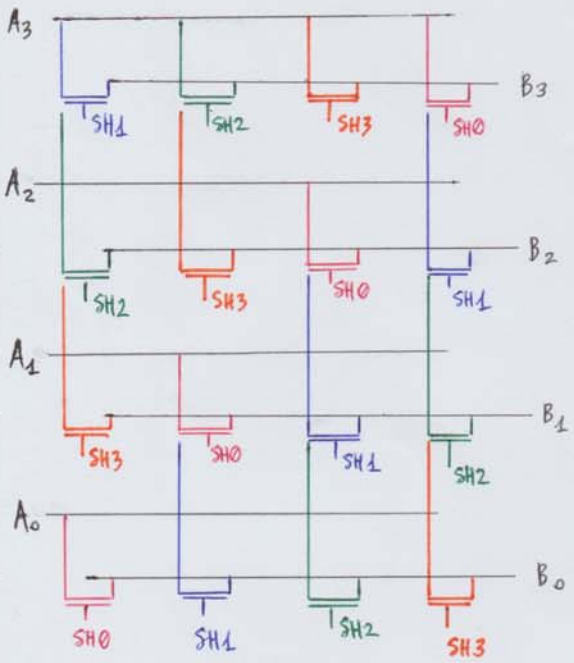
IF $\alpha = 0$ AND $\beta = 1$ AND $\gamma = 0$ $\Rightarrow B_i = A_{i-1} \Rightarrow$ SHL

IF $\alpha = 0$ AND $\beta = 0$ AND $\gamma = 1$ $\Rightarrow B_i = A_{i+1} \Rightarrow$ SHR

COMMENTS: this is a very simple and compact shifter. It is possible to add more connections in order to perform shifts of more than one position in a single pass.
More connections = more control signals.
IMPORTANT: always only once signal should be = 1

Usually is required a more configurable/programmable shifter...

BARREL SHIFTER:



NB: to simplify the drawing, TG are reported as 

IF SH0 = 1 AND all other signals = 0 $\Rightarrow$ SH0 = NOP

IF SH1 = 1 ——"—— $\Rightarrow$ SH1

IF SH2 = 1 ——"—— $\Rightarrow$ SH2

IF SH3 = 1 ——"—— $\Rightarrow$ SH3

COMMENTS: Regularity helps a lot in area compactness. Here in particular, is easy to build a transistor matrix, and then connect to input and output.

Suppose a SHIFT REGISTER OF 8 BITS, you'll need 4x transistors and 4x area
To enable SHR, you'll need 2x transistors...

NOTES:

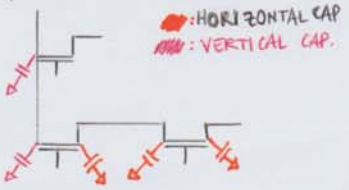
1) Long lines: a line has a parasitic capacitance! Longer the line, bigger the capacitance. Transistors connected to long lines should be correctly sized (BIGGER = FASTER).

2) parasitic capacitance of transistors: also those capacitance are summed with the line capacitance on the line

NB: SH0, SH1, SH2, ... SHOULD DRIVE A LOT OF GATES ⇒ LOAD = SUM OF GATE CAPS.

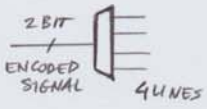
1 DROP V CASE ⇒ 1 PARASITIC RESIST.

PARASITIC CAPACITANCES
- HORIZONTAL LINES
- VERTICAL LINES
- DRAIN / SOURCE / GATE OF TRANSISTORS



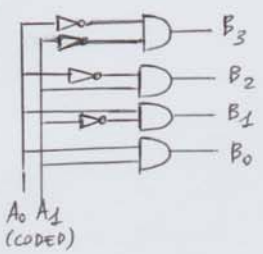
DECODERS

RECALL:



Combinational logic that converts binary information from the m coded inputs to a maximum of 2^m unique outputs.

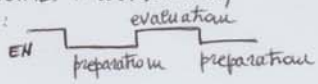
CMOS IMPLEMENTATION:

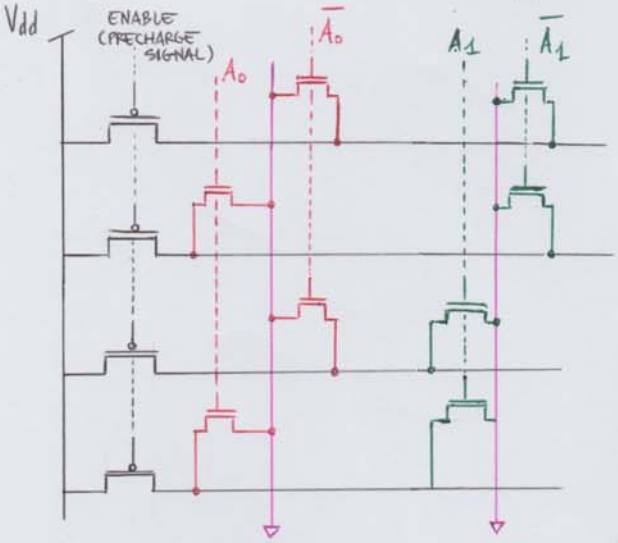


4 ANDS = 24 TRANS +
4 INV = 8 TRANS =
32 TRANS

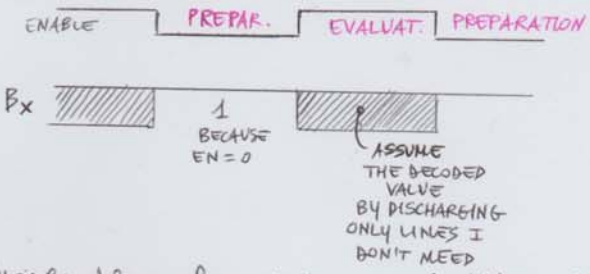
PASS TRANSISTORS IMPLEMENTATION:

NB: STATIC TOPOLOGY ≡ COMBINATIONAL
DYNAMIC TOPOLOGY ≡ COMBINATIONAL, BUT CHARACTERIZED BY A CONTROL SIGNAL. IT WORK USUALLY IN 2 PHASES:





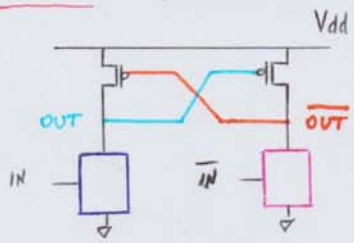
A_0	$\overline{A_0}$	A_1	$\overline{A_1}$	B_3	B_2	B_1	B_0
0	1	0	1	0	0	0	1
1	0	0	1	0	0	1	0
0	1	1	0	0	1	0	0
1	0	1	0	1	0	0	0



DRIVING CAPABILITIES: every signal should drive 2 transistors. Parasitic capacitance = caps of line + caps of transistor

Typical problem of TG (out not perfect Vdd or GND) is avoided due to direct connection between Vdd or GND to output

DC VSL : differential CMOS...

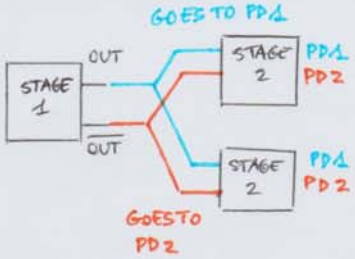


N NETWORK (PULL DOWN 1)

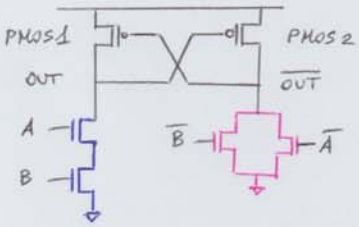
N NETWORK (PULLDOWN 2) COMPLEMENTARY W.R.T PULLDOWN 1 (SERIES → // AND VICEVERSA)

NOTES: 1) TOPOLOGIES OF PD1 AND PD2 ARE COMPLEMENTARY

2) INPUTS IN AND $\overline{IN}$ ARE INDIPENDENT AND GENERATE FROM PREVIOUS STAGES: NO NEED FOR AN INVERTER!



Example:



$OUT = \overline{A \cdot B}$, $\overline{OUT} = \overline{B + \overline{A}}$

Suppose $A = B = 0$ at the beginning. PD2 is a short circuit to ground, so the gate of PMOS1 is connected to ground. It means PMOS1 is ON, am because PD1 is am open circuit, the gate o PMOS2 see 1, and so it is OFF.

BECAUSE PMOS2 START TO GO ON, $\overline{OUT}$ STARTS TO BECAME 1, SWITCHING OFF PMOS1 ... AT THE BEGINNING IT IS SLOW, BUT THE SPEED INCREASE OVER TIME

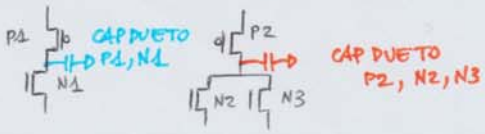
IT BEHAVE LIKE A POSITIVE REACTION!

ADVANTAGES:

1. POSITIVE REACTION SPEEDS-UP THE SWITCHING
2. SHORT CIRCUIT POWER (VDD-GND DIRECT CONNECTION DURING TRANSITION) IS REDUCED
3. AS IN NMOS-LIKE, GATE CAPACITANCE OF THE LOAD IS REDUCED; EACH OUT DRIVES ONLY 1 NMOS W.R.T CMOS → 1 NONOS, 1 PMOS).
4. USED TO INCREASE S.N.R (RESILIENCE TO NOISE). IN FACT IT IS NOT USED A STANDALONE SIGNAL, BUT THE DIFFERENCE BETWEEN TWO SIGNALS.
5. INTRINSICALLY GENERATES OUT AND $\overline{OUT}$, NO INVERTER NEEDED

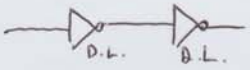
DISADVANTAGES:

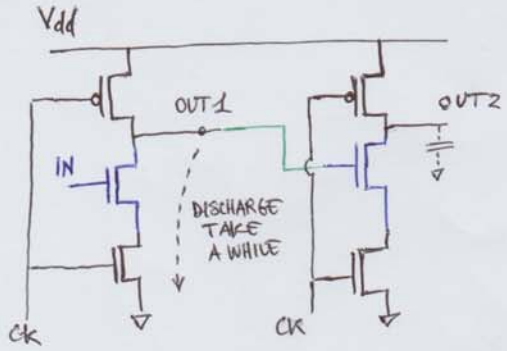
1. AREA: 2 PULLDOWN NETWORKS + 2 PMOS ARE USED (HOWEVER SAME NUMBER OF TRANSISTOR OF A CMOS IMPLEMENTATION + INV)
2. OPTIMAL SIZING IS A CRITICAL DECISION: CAPACITANCE OVER POSITIVE REACTION NETWORK SHOULD BE THE SAME

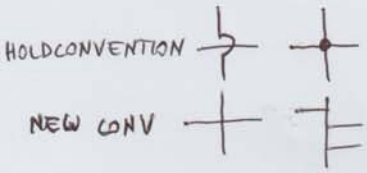


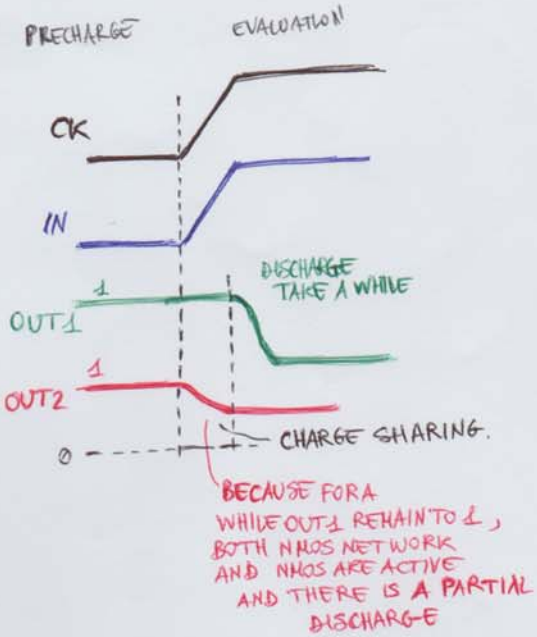
CHARGE SHARING (IN DOMINO LOGIC)

8/5/2018 PT1
REG REG
μELECTR

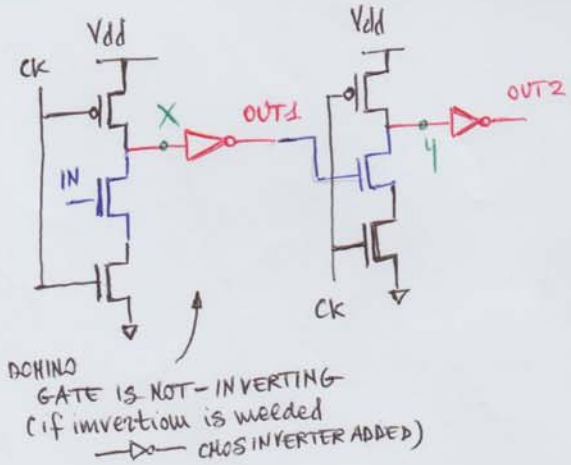


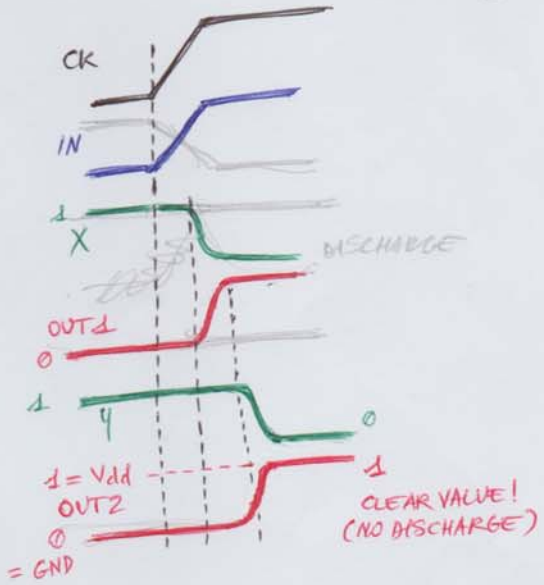




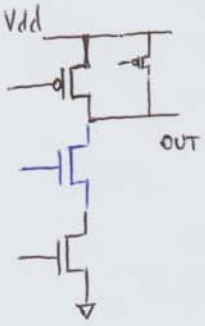


SOLUTION

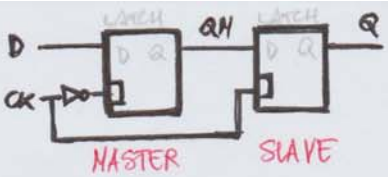




SOLUTION 2: KEEPER

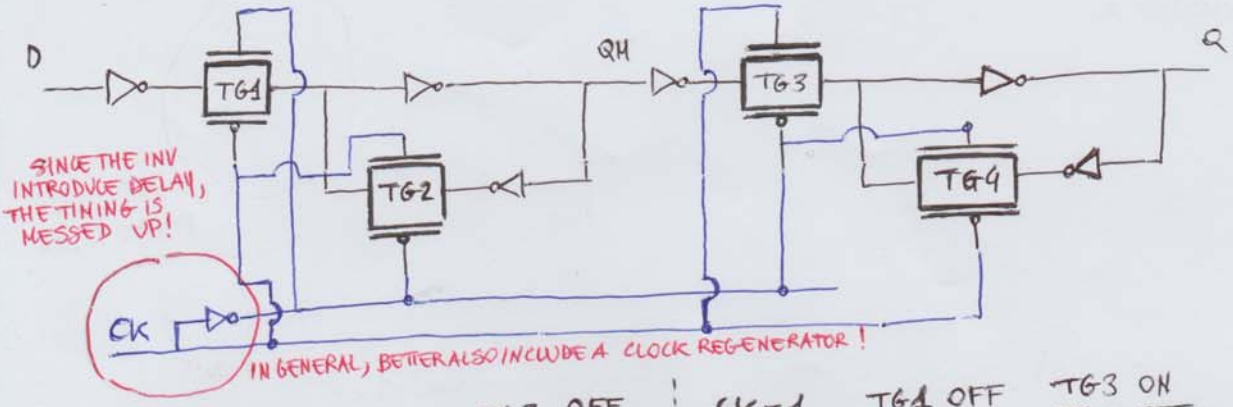


FLIP-FLOP



MASTER AND SLAVE WORK IN OPPOSITE PHASE (SEE CK)

8/05/18 PT2
~~REG REG~~ ELECTR.

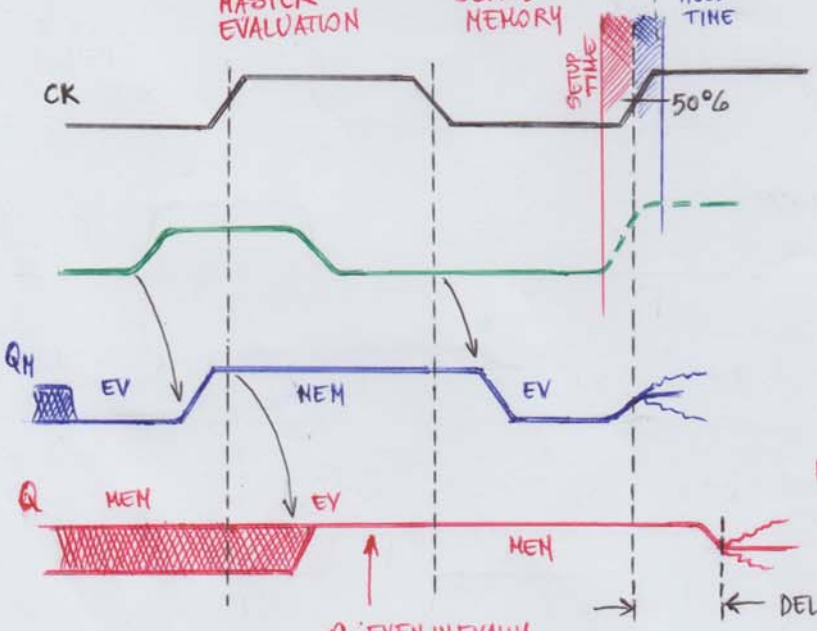


CK = 0 — TG1 ON, TG2 OFF — MASTER EVALUATION

TG3 OFF, TG4 ON — SLAVE MEMORY

CK = 1 — TG1 OFF, TG2 ON — MASTER MEMORY

TG3 ON, TG4 OFF — SLAVE EVALUATION



possible SETUP time violation of MASTER: for sure the problem is trasfered to the slave (during slave evaluation)

METASTABILITY! It depends on many parameters

DELAY $t_{CK \to Q} = t_{50\%Q} - t_{50\%CK}$

IT DEPENDS ON
- SWITCH DELAY OF CK
- SWITCH DELAY OF 2 INV
- SWITCH DELAY OF TG3

Q, EVEN IN EVALUATION, WILL REMAIN STABLE (AFTER THE EVALUATION) ⇒ NO RISK OF ~~SETUP~~ TIME VIOLATION

⇒ EDGE TRIGGERED : Q CHANGE AFTER THE EDGE OF CLOCK (transaction of CK) BECOUSE OF DELAYS!

MITH BUSTING: a flipflop isn't a magic box that sample at the clock edge and then is totally blind. Instead, inside it, a lot of things happen, without any stop!

⇒ POWER CONSUMPTION

⇒ NOISE

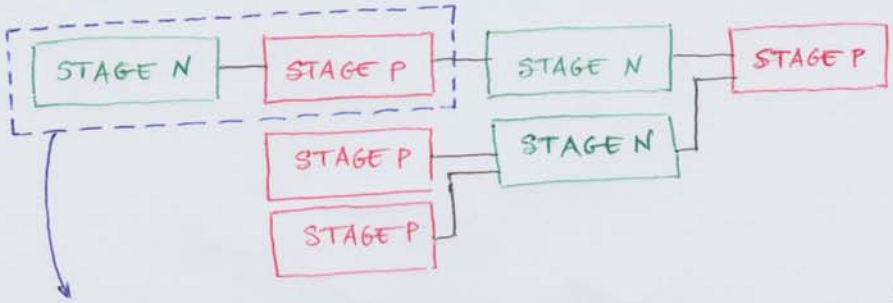
CLOCK REGENERATOR:

The system given clock should pilot a lot of logic inside our system. Should also travel long distances to reach our flipflop. A typical clock is a sine wave (∿∿). This circuit introduce a little latency, but increase the clock slope!

9/05/2018
REG REG

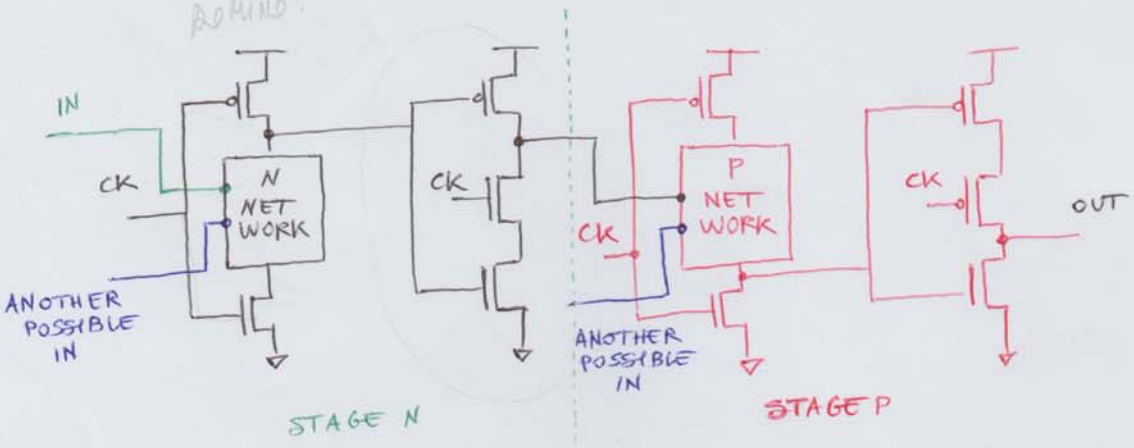
DYNAMIC SEQUENTIAL LOGIC:

TRUE SINGLE PHASE CLOCKING ⇒ TSPC

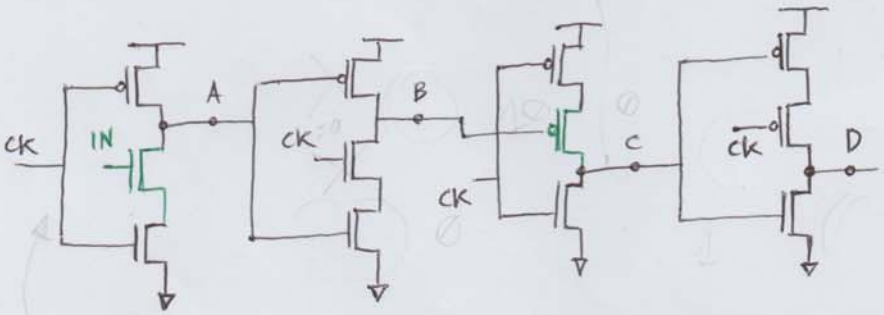


fino a MIN 6

BOUND!



Example: LOGIC FUNCTION 1 TRANSISTOR → INVERTER



CK=0 PRECHARGE
CK=1 EVALUATION

MEMORY
EVALUATION

in precharge A=1 so PMOS is OFF. the NMOS connected to CK is OFF. It means neither the pull down or pull up is driving B, that maintain the old value!

EVALUATION
PRE DISCHARGE

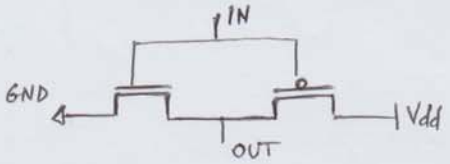
depending on the value of IN I can activate the pull up or maintain both disabled, maintaining zero

EVALUATION
MEMORY

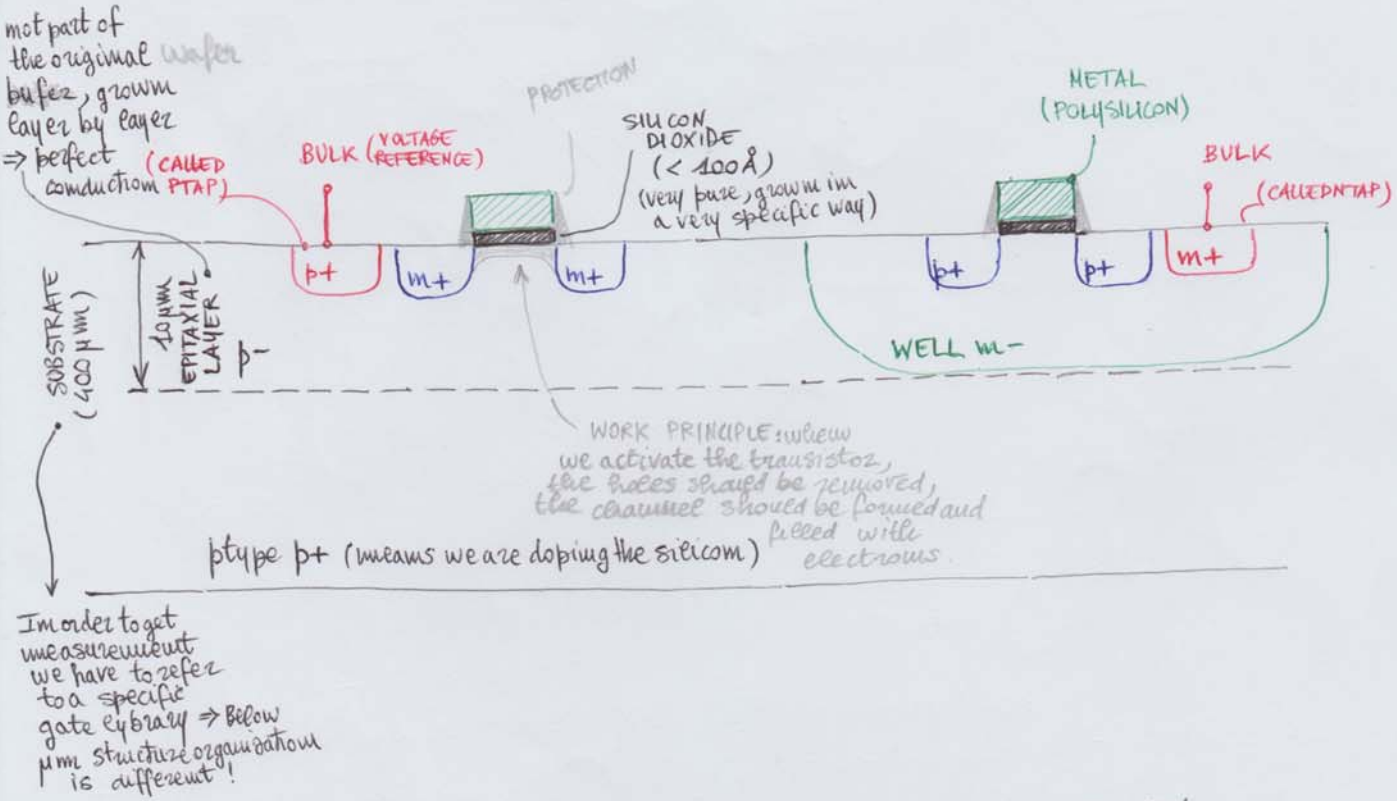
MIN 21-24

This logic allow to embedd a logic function inside a latch. It's like computing on a double data rate, because we perform EXE+MEM in the same clock cycle.

FROM SWITCH LEVEL TO TRANSISTOR LEVEL (TOWARDS GATE LIBRARY DESIGN & CHARACTERIZETION)



POLYSILICON: it is a silicon based conductor - Because we use silicon both for metal, oxide and semiconductor, the mobility, conduction and in general the compatibility is higher! (interface)



In order to get measurement we have to refer to a specific gate library ⇒ Below μm structure organization is different!

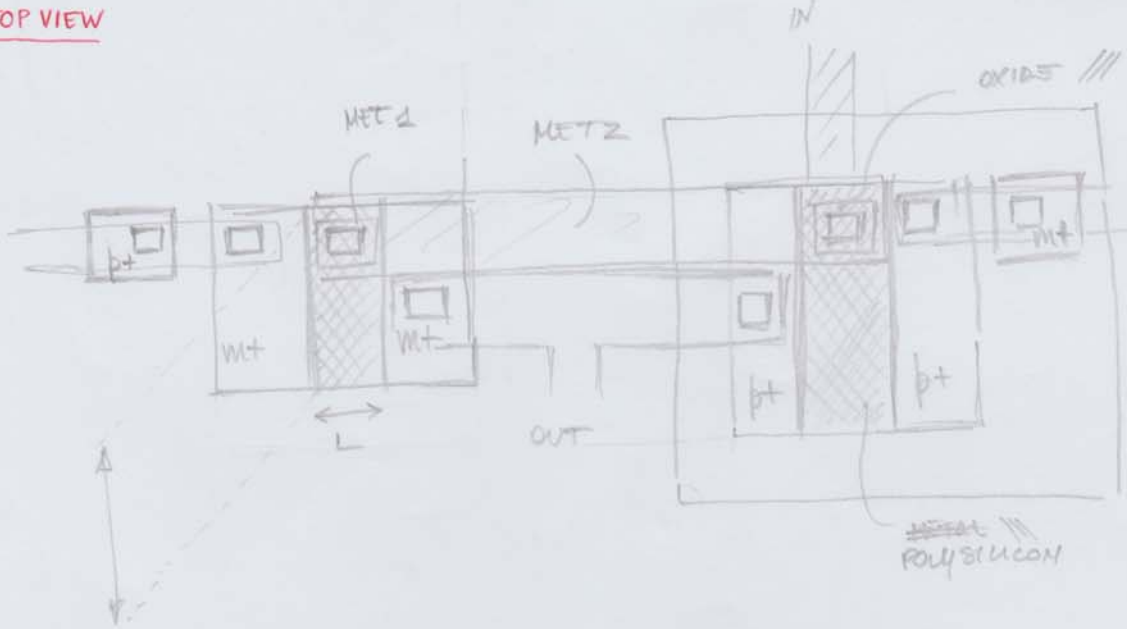
Why silicon is so widely used in the electronic industry, so that often is called "silicon industry"?

1. Easy to find on planet earth: for sure is true, and for sure it helped at the beginning of the electronic study and development, but is not the main reason.

2. Silicon dioxide: in nmos (metal oxide semiconductor) is really important the oxide layer.
   Since silicon dioxide is a natural oxide of silicon: by exposing silicone to ~~a certain~~ oxigen (at a certain temperature and pressure) it will form naturally a even layer. ⇒ Masks
   → The layer between metal and semiconductor should be placed / buried in a different way, because it should be very defined. REMENBER: between metal, oxide and semiconductor the channel should form if the transistor is active.
   → the thinner the oxide layer is, the better is for the overall transistor quality / behaviour:
      1. Less gate capacitance
      2. Threshold voltage
      3. ...
      4. ...

2 plates of a capacitor

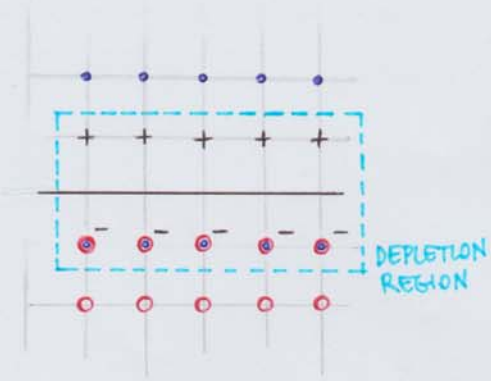
TOP VIEW



TRANSISTOR PMOS ARE BIGGER BECAUSE MOBILITY IS LOWER AND I WANT THE SAME DELAY OF NMOS

MASK VIEW

2

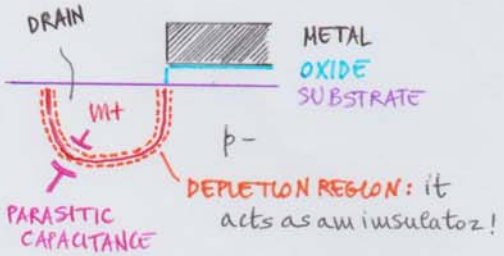


After the jump, the **border** of n-doped side is characterized by a negative **charge**, and the border of p-doped side is **characterized by a positive charge.**

This cause two things:

1. Electric field (called built-in electric field) is generated
2. A voltage difference is formed all across the border: a wall that is hard to pass by electrons. (it is around 0.6 V = V_γ of a diode)

NIN 17-19



Another parasitic capacitance! Semiconductor (p-), insulator (depletion region), semiconductor (n+) = parallel plate capacitor

JUNCTION CAPACITANCE:

$$C_j = \frac{C_{j0}}{\sqrt{1-\frac{V_D}{\phi_0}}} = A_D f(N_D, N_A, \dots)$$

(OTHER PARAMETERS DEPENDS ON TECHNOLOGY)

$A_D = W \cdot X$ — DESIGN PARAMETER

BUILT IN JUNCTION POTENTIAL

$$\phi_0 = \phi_T \ln\left(\frac{N_A N_D}{n_i^2}\right)$$

n_i = CONCENTRATION OF INTRINSIC SILICON

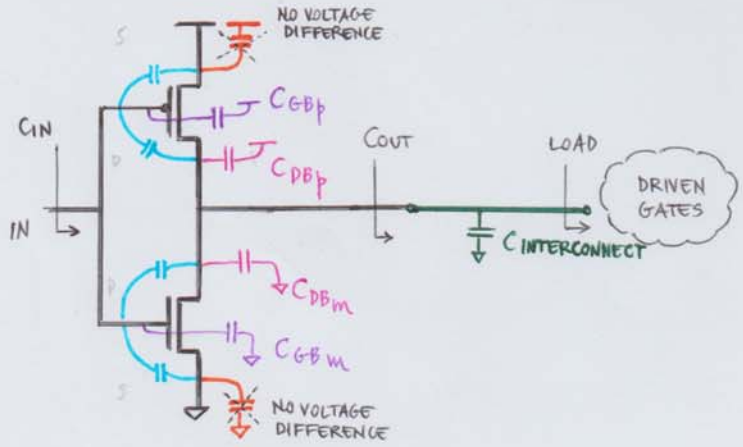
V_D of diode that depends on V_T (threshold voltage) + current

$I_d = I_S e^{V_D/mV_T}$

$V_D = mV_T \ln\left(\frac{I_D}{I_S}\right)$

Recap example: inverter

pmos: BULK connected to Vdd
nmos: BULK connected to GND



FEED THROUGH!

$$C_{IN} = C_{GBn} + C_{GBp} + C_{GDn} + C_{GDp} + C_{GSn} + C_{GSp}$$

$$C_{OUT} = C_{LOAD} + C_{INTERCONNECT} = C_{LOAD} + C_{DBn} + C_{DBp}$$

BODY EFFECT

Even if we are able to build the exact same transistor multiple times (and it is not possible), every transistor can be characterised by a different threshold voltage V_T, resulting in a more difficult way to drive correctly the gate.

Example:

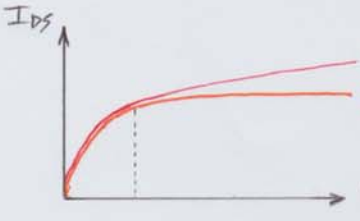
single transistor:

$V_T = V_{T0}$

simple gate:

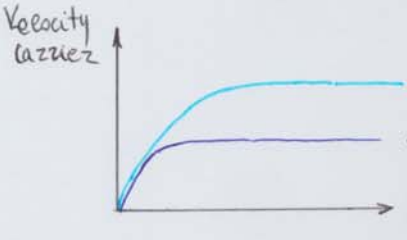
$V_T = V_{T0} + \gamma(V_{SB}, \dots)$

$V_T = V_{T0}$



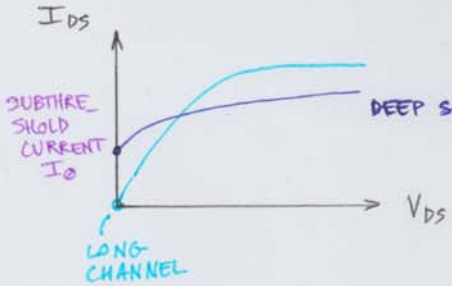
$I_{D\,REAL} = I_D(1 + \lambda V_{DS})$ CHANNEL LENGHT MODULATION

$I_{D\,IDEAL}$



LONG CHANNEL

SHORT CHANNEL ⇒ smaller mobility ($\mu \downarrow$)



DEEP SUB MICRON (= SHORT CHANNEL)

HIGH V_T transistors have low I_0, but transition is slower (higher time to reach a bigger V_T)

Long channel: $V_{DS} = 0 \Rightarrow I_{DS} = 0$ no leackage

short channel: $V_{DS} = 0 \Rightarrow I_{DS} = I_0$ leakage!

I_0: DEPENDS ON TEMPERATURE QUADRATICLY; DEPENDS ON V_T

(A) $V_{IN} = 0$

Supposing a past history of transitions, in this condition $V_{OUT} = V_{dd} = 1$
(already powered on, not our first switch ...)

NMOS OFF
- $V_{GS_N} = V_{IN} - GND = V_{IN} - 0 = 0$ — NMOS OFF (in OFF region) because of NO CHANNEL
- $V_{DS_N} = V_{OUT} - GND = V_{OUT} - 0 = V_{dd}$ — VERY HIGH: if V_{DS_N} is high, potentially we can move a lot of charges ... however, because $V_{GS_N} = 0$ there are no charges (because there is no channel ...)
- NO CHANNEL

PMOS OFF
- $V_{GS_P} = V_{in} - V_{dd} = 0 - V_{dd} = -V_{dd}$ — VERY HIGH, BUT NEGATIVE: there are a lot of holes in the channel
- $V_{DS_P} = V_{OUT} - V_{dd} = 0$ — there is not electric field in the channel: the charges are in the channel but are not moving ⇒ POFF
- THERE IS CHANNEL, BUT NOT ELECTRIC FIELD

(B) $V_{in} \uparrow$ (Vin is increasing)

$V_{GS_N} \uparrow$

$|V_{GS_P}| \downarrow$ but still is NEGATIVE

Are we going above the threeshold or not? There is an empiric rule (however it depends on technology): $V_T \approx V_{dd}/10$ ⇒ the threeshold voltage value is around $\frac{1}{10} V_{dd}$ ...

Is very likely, because V_T is so small, that V_{GS_N} is around the threeshold voltage ⇒ $V_{GS_N} \approx V_{T_N}$
V_{GS_P} remains above the threeshold voltage ⇒ $|V_{GS_P}| \gg |V_{T_P}|$

We are not putting numbers: however for sure the PMOS has (still) the channel formed and suppose that those conditions on Vin value are such that also NMOS is ON (formed a channel)

⇒ BOTH NMOS and PMOS HAVE CHANNEL FORMED
- NMOS HAS FEW CHARGES (e−) BELOW THE CHANNEL
- PMOS HAS A LOT OF CHARGES (HOLES) BELOW THE CHANNEL

$V_{DS_N} \downarrow$ remains high, remains a strong electric field across the channel ⇒ THERE IS A SMALL CHANNEL + BIG ELECTRIC FIELD ⇒ NMOS IS ON!

$V_{OUT} \downarrow$

By looking at the NMOS, probably it is in a SATURATION REGION because $V_{DS_N} > V_{GS_N} - V_T$ $(\approx 1 \gg \approx 0)$
PMOS SATURATION (?) $V_{DS_P} > V_{GS_P} - V_T$ $(0 \gg \text{NEG})$

$|V_{DS_P}|$ starting to increase, however it assume a NEGATIVE VALUE. ⇒ a small electric field is formed ⇒ PMOS transistor is ON!

(D) $V_{im} \approx V_{dd}$ (ALMOST)
$V_{out} \approx 0$ (ALMOST)

$V_{GS_p} \approx 0$ $V_{DS_p} \approx -V_{dd} \Rightarrow$ BIG ELECTRIC FIELD (look V_{DS_p}) ALMOST NO CHARGES INSIDE THE CHANNEL (look V_{GS_p})

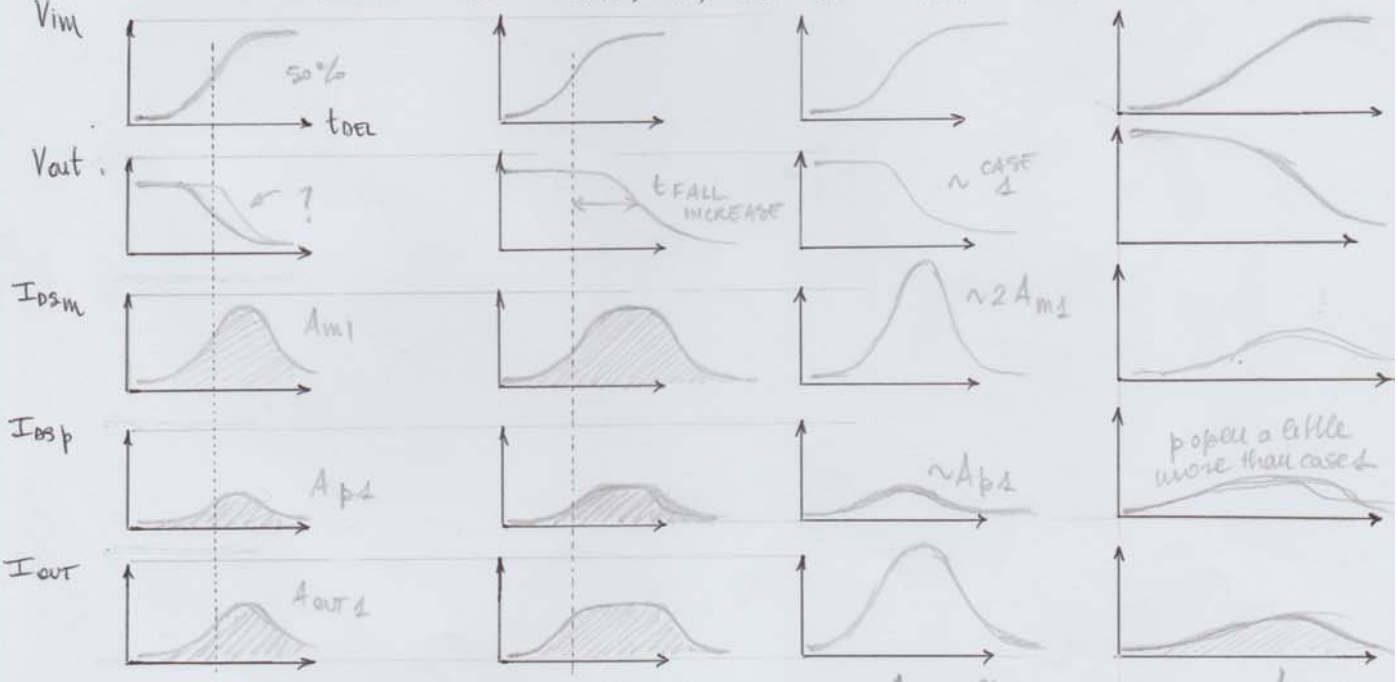
$V_{GS_m} \approx V_{dd}$ $V_{DS_m} \approx 0 \Rightarrow$ ALMOST NO ELECTRIC FIELD ACROSS THE CHANNEL (look V_{DS_m}) A LOT OF CHARGES IN THE CHANNEL (look V_{GS_m})

(E) $V_{im} = V_{dd}$
$V_{out} = 0$
$V_{GS_m} = V_{dd}$ $V_{DS_m} = 0 \Rightarrow$ ~~NO CHARGE~~ NO ELECTRIC FIELD

$V_{GS_p} = 0$ $V_{DS_p} = -V_{dd} \Rightarrow$ NO CHARGES INSIDE CHANNEL

We can try to make a further step: I'm in this condition (??), but I want to consider some variations:

(1) C_{LOAD}, W_m, t_{RISE} (2) $2 \cdot C_{LOAD}, W_m, t_{RISE}$ (3) $2 \cdot C_{LOAD}, 2 \cdot W_m, t_{RISE}$ (4) $C_{LOAD}, W_m, 2t_{RISE}$

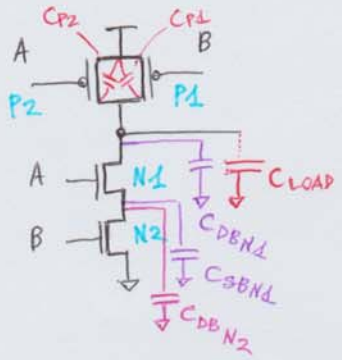


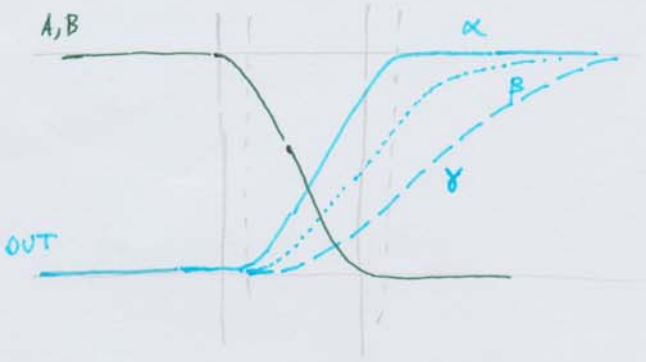
NB I_{DS_m} is changing over time. Consequences of case 2: tFALL increase!

higher power consumption
slower
slows down all following gates
big current in the p

NB: CASE 3: doubling the transistor parasitic capacitance.

DEPENDENCY ON INPUT PATTERN





SITUATION α:

$A_1 \rightarrow 0$

$B_1 \rightarrow 0$

N_1, N_2 ON → OFF

P_1, P_2 OFF → ON

starting situation: for sure $V_{OUT} = 0$ and $C_{LOAD} + C_{DBN1} + C_{SBN1} + C_{DBN2}$ are all discharged completely

OUT → 1 supposing NMOS switch off immediately:

$P_1 + P_2$ CHARGING
- C_{LOAD} CHARGED
- C_{DBN1} CHARGED
- C_{p1} CHARGED
- C_{p2} CHARGED

REASON WHY WHEN DON'T PUT TOO MANY INPUTS IN A SINGLE GATE = BETTER TO SPLIT !..

TRY A NOR

SITUATION β:

$A_1 \rightarrow 0$

$B_1 \rightarrow 1$ (remains at 1)

N_1 ON → OFF

N_2 ON

P_2 OFF → ON

P_1 OFF

OUT → 1, P_2 CHARGING
- C_{LOAD}
- C_{DBN1}
- C_{p1}, C_{p2}

even if one of the p is off. Only one p is doing the job

⇒ slower out: bigger propagation time increasing rise time

SITUATION γ: WORST CASE

$A_1 \rightarrow 1$

$B_1 \rightarrow 0$

N_1 ON

N_2 ON → OFF

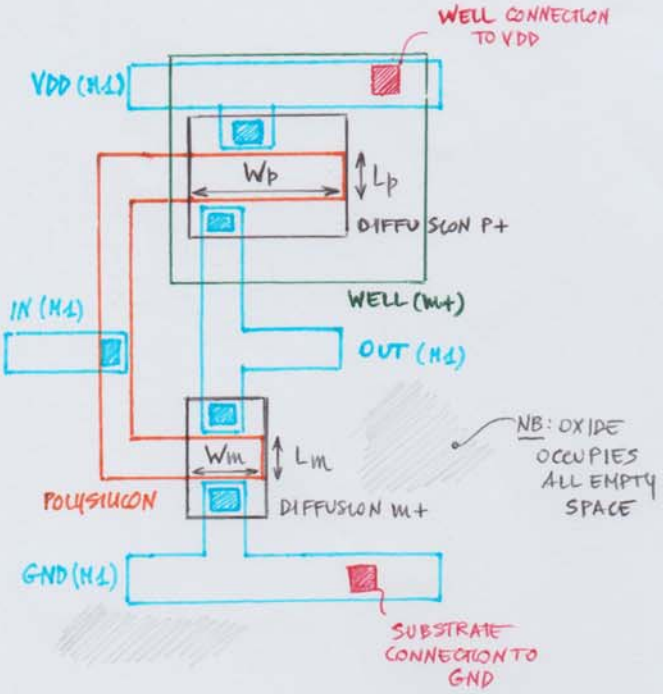
P_2 OFF

P_1 OFF → ON

OUT → 1, P_1 CHARGING
- C_{LOAD}
- C_{DBN1}
- C_{p1}, C_{p2}
- C_{SBN1} → Because N_1 is ON!
- C_{DBN2} →

③ LAYOUT OF EACH CELL

- ALL HEIGHTS H IDENTICALS
- WIDTH CHANGE DEPENDING ON CELL COMPLEXITY & TRANSISTORS SIZING



LAYOUT CHECKED VS / DESIGN RULES
\ SCHEMATIC (the figure represent an invertez!)

④ SPICE EXTRACTION : example of an invertez

```
                NAME OF GATE
.SUBCKT   INV   IN OUT Vdd GND        ALL IN/OUT PINS
X: PORTMAP
P1, N1
ARE TRANSIST.
          XP1   OUT IN VDD VDD
          XN1   OUT IN GND GND        ALL PINS AS USUAL (SOURCE, DRAIN, GATE, BULK)
PARASITIC
CAPS,
RES, ....
          CPAR1 ....
          CPAR2 ....
          CPAR3 ....
          CPAR4 ....
.END
```

⑤ CHARACTERIZATION

Analysis of all delays, behaviours ... by simulating for a set of different values of IN, etc...

⇒ SPICE SIMULATIONS FOR ALL CASES

⇒ MEASURE: $t_{PROP_{LH}}$ (when OUT ↗)

PROPAGATION TIME = $t_{50\%\,OUT} - t_{50\%\,IN}$: $t_{PROP_{HL}}$ (when OUT ↘)

RISE AND FALL TIME MEASURED BETWEEN 10%–90%: t_{RISE}, t_{FALL} (90% / 10%)

\+ ALL CURRENTS! (SWITCHING, SHORT CIRCUIT, LEAKAGE)

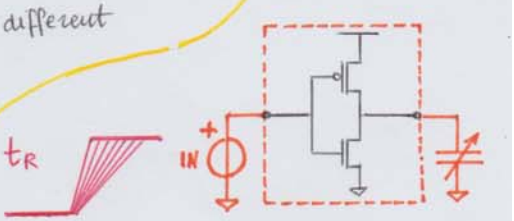
⇒ FOR COMPLEX GATES (more than one input) SIMULATIONS ARE REPEATED FOR ALL POSSIBLE INPUTS SWITCHING CASES!

↳ EACH IN-OUT COMBINATION IS STORED IN A L.U.T.
⇒ ONE L.U.T FOR EACH MEASURE !!
MANY L.U.T.S AT THE END

⇒ DESCRIBED IN LIBERTY FILES (.lib) AND USED FOR FURTHER SYNTESIS PHASES

No matter how many simulations you run, you'll have always a finite number of values. Suppose that your load is not exactly represented in the L.U.T.: just do a weighted average! ⇒ FIND WHAT IS CALLED AN EFFECTIVE VALUE! (INTERPOLATION) is highlighted in the LUT down below...

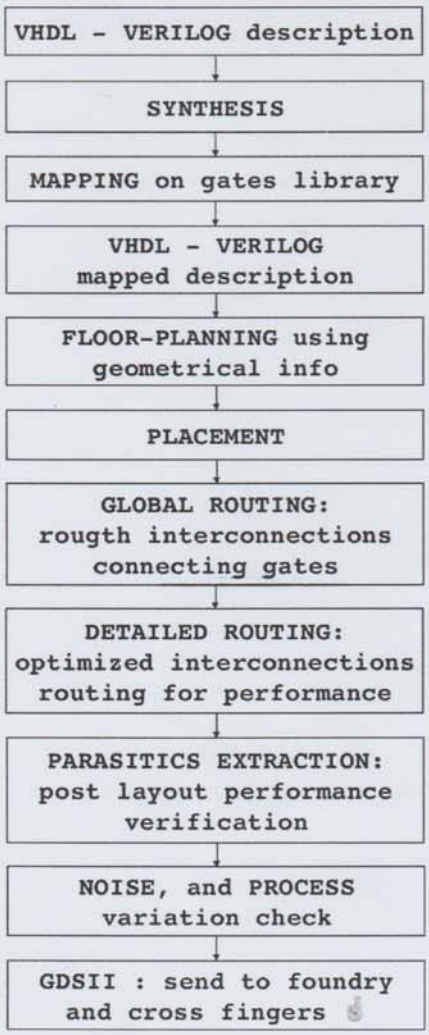
Example: $C_{L2} < C_{EFF} < C_{L3}$, $t_{R2} < t_R^{EFF} < t_{R3}$



	C_{L1}	C_{L2}	C_{L3}	C_{L4}	...
t_{R1}	t_{p11}	t_{p21}	t_{p31}	t_{p41}	...
t_{R2}	t_{p12}	t_{p22}	t_{p32}	t_{p42}	...
t_{R3}	t_{p13}	t_{p23}	t_{p33}	t_{p43}	...
⋮	⋮	⋮	⋮	t_{p44}	...

ASIC design flow (top-bottom)

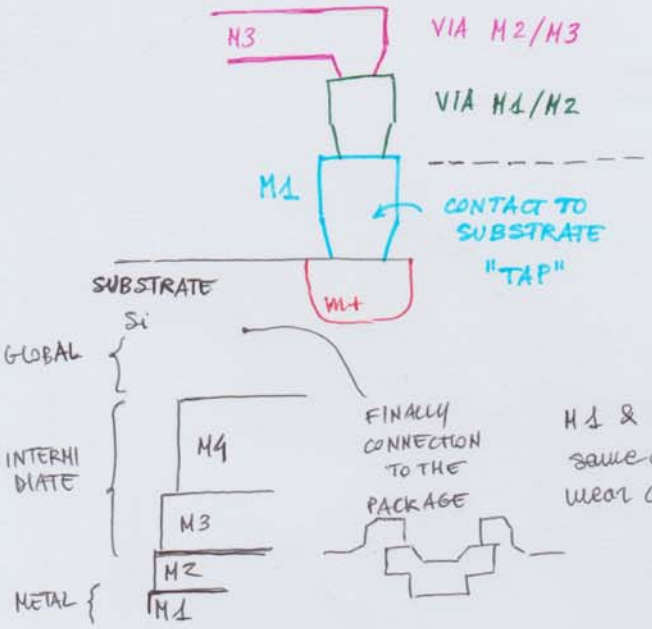
1. Based on a library of gates: a library contains all kinds of different gates (nand, and, or, inv, buffer, xor, mux, flip-flop, latches...), both elementary blocks and more complex ones.
2. Each gate is designed to drive multiple loads (X4, X8, ...). The driving capability is represent as the maximum number of elementary blocks the gate can drive without lose control of delay. Usually an elementary block correspond to an inverter (look documentation)
3. Each gate has different implementations, each one optimized for speed (HS) and/or for power (LL).
4. Each gate is described at
   - Transistor level (spice)
   - Layout level (LEF)
   - Timing-power-boolean (lib)
   - Behavioral (VHDL): VHDL is used as input of the synthesis part. However it can be used also after the synthesis analysis or even after the layout => the estimated delays can be used to run a realistic simulation (simulink). Maybe you need two change layout or architecture entirely because the circuit does not respect constraints.



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 77 di 114

INTERCONNECTIONS

29 MAY
PT 1



FIRST LAYER COULD BE DONE WITH A DIFFERENT TYPE OF METAL (TUNGSTEN) BECAUSE WE NEED A MECHANICAL STABILITY

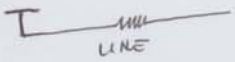
M1 & M2 are used to connect points on the same cell, ... M3 ... to connect points of near cells, ... MX ... to connect macroblocks

HIGHER LAYER = MORE METAL (HIGHER TICKNESS)
BECAUSE OF ~~BIGGER~~ ~~SMALLER~~ SMALLER CURRENT DENSITY

GLOBAL DELIVER A VERY LARGE AMOUNT OF CURRENT

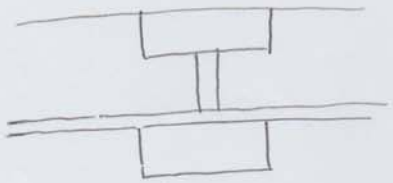
1. Heating

2. Voltage drop (expecially on power supply lines) ≡ Noise



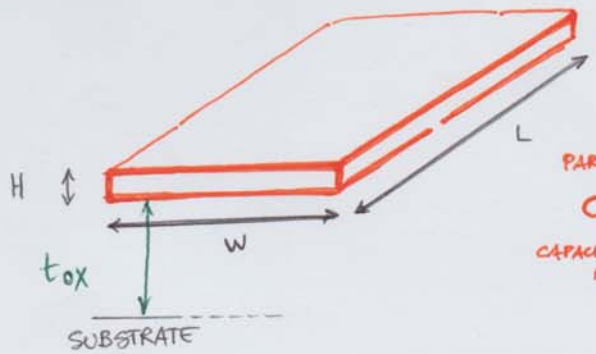
3. Relayability: protection from overcurrent. In a lot of systems there are fuse, that can deliver only a certain amount of current, otherwise it will brake... ELECTROMIGRATION

example: ST_ELECTRONICS



$k = 2.55 = \varepsilon_r$

LOW DIELECTRIC CONSTANT TO REDUCE ~~CAPACITANCE~~; HOWEVER THEY TEND TO BE WEAK ⇒ USE THIN MATERIAL IN THE MIDDLE (IN ORDER TO NOT CONTRIBUTE TOO MUCH ON CAP)



PARASITIC CHARACTERIS.
C, R, L
CAPACITANCE
RESISTANCE
INDUCTANCE

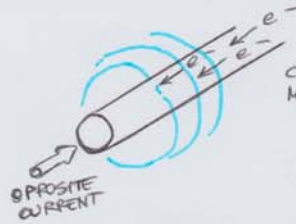
RESISTANCE DEPENDS ON FREQ. AND TEMP.

1. FREQUENCY R(f)

- SECTION OF A CONDUCTOR
- CURRENT IS EXPECTED TO DISTRIBUTE UNIFORMLY IN THE WHOLE SECTION

- IT MIGHT HAPPEN THAT CURRENT CONCENTRATE MORE IN THE PERIPHERY.

⇒ SKIN EFFECT



CHARGE MOVEMENT GENERATES A MAGNETIC FIELD: IT GENERATES AN OPPOSITE CURRENT, THAT IS MORE CONCENTRATE IN THE CENTER

SKIN DEPTH: $\delta = \sqrt{\frac{\rho}{\pi \mu f}}$

$\mu_0 \mu_r$ PERMEABILITY

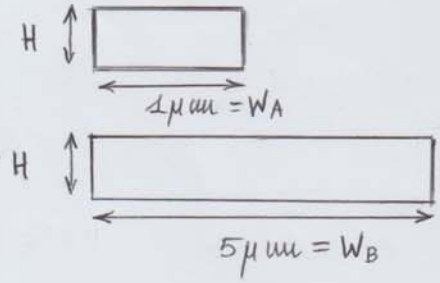
DISTANCE FROM SURFACE WHERE I REDUCES TO $\frac{1}{e} \approx 37\%$

HOW FREQUENCY IS DEFINED FOR SQUARE WAVES?

ex: CLK

$T \rightarrow \frac{1}{T} = f$ + other components

Let's use only main frequency component:



f	δ (SKIN DEPTH)
100 MHz	11 μm = δ_1
500 MHz	5.19 μm = δ_2
1 GHz	3.67 μm = δ_3
2 GHz	2.6 μm = δ_4

$\delta_1, \delta_2, \delta_3, \delta_4 > W_A \Rightarrow$ NO SKIN EFFECT ON CONDUCTOR A

ESPECIALLY ON CASE 4: $\frac{W_B}{2} \approx \delta_4$ SKIN EFFECT IS PRESENT!

Skin effect has an impact: effective resistance increasing

it is like conductor B has a smaller section: smaller current flow ⇒ INCREASE RESISTANCE

TECHNOLOGY DEFINES MAXIMUM WIDTH, IN ORDER TO AVOID THE SKIN EFFECT

$$I_{RMS} = \sqrt{\frac{1}{T}\int_0^T I^2(t)\,dt}$$

$f \uparrow \; I_{RMS} \uparrow \; P_m \uparrow \; T_m \uparrow \; R \uparrow$

$W \uparrow \; \theta \downarrow \; T_m \downarrow \; R \downarrow$

$t_{ox} \uparrow \; \theta \uparrow \; T_m \uparrow \; R \uparrow$ but HIGHER M.LAYER $H \uparrow \; R \downarrow$

HIGHER M.LAYER

MIN 34 - 37 UNTIL E.M.

ELECTRO MIGRATION

If electrons in a conductor have sufficient energy VOIDS (open circuits) or HILLOCKS (short circuits) could be generated.

ELECTRONS FLOW (with certain density and speed)

Electronic wind can accelerate electrons that can trasfer some quantity of motion to metal atoms. If it happens continuously several atoms could move as an avalanche

More probable if:

1. many electrons are present (METAL, POLY SILICON)
2. high current density ($J = I/WH$)
3. electrons free to move — BOUNDARIES / DEFECTS / GRAIN BOUNDARIES

BIG W reduce resistance but increases the risk of electromigration.

BLACK'S LAW (empirical):

$$M.T.F. \propto \frac{1}{J^2} e^{\frac{Q}{KT_m}}$$

M.T.F. = MEAN TIME TO FAILURE

Q = ACTIVATION ENERGY depends on the type of metal lattice

T_m = ACTUAL METAL TEMPERATURE

K = BOLTZMAN CONSTANT

J = CURRENT DENSITY ← BETTER ROUTE POWER LINES ON HIGHER M. LAYERS IN ORDER TO REDUCE J ($H \uparrow$ MTF $\uparrow$)

M.T.F. is usually measured in years (consummer electronics has 10 years of MTF).

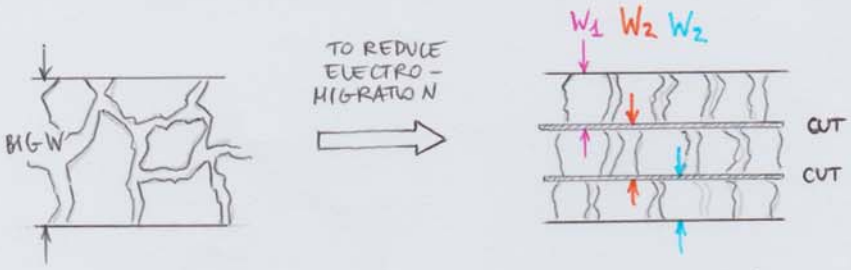
M.T.F. it is exponential dependent on the temperature!

5

30 MAY

RECAP: grain boundaries are a characteristic of polycristalline materials, like the metal used for interconnections. They are defects in the cristal structure, and tend to decrease both electrical and thermal conductivity of the material. For our analysis, is important to remember that atoms of metal hit by electrons (current = flow/motion of electrons) ~~can~~ can move into the grain boundaries.

By reducing the order of magnitude of W (width) of the wire, grain boundaries has a perpendicular ($\perp$) direction with respect to the electrons directions.
This physical layout reduce the probability that an atom of metal, hitten by electrons, falls into the grain boundaries.



Several cuts creates the equivalent of several small lines in parallel.

$W_1 + W_2 + W_3 \approx W$

Same current driving capabilities.

$W_1, W_2, W_3 \approx$ SIZE OF GRAIN

When a lot of current has to flow, a big wire is required: BIG CURRENT = BIG WIRE

Bamboo effect is favoured ⇒ Electro-migration risk reduced.

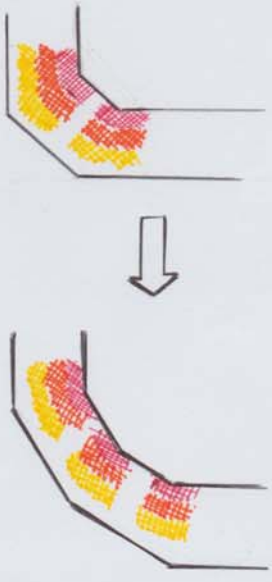
CHEESED INTERCONNECT

Especially used in power and ground lines

CORNERS:

Usually in interconnection layouts, lines are strait, or characterized by 90° corners.
However:

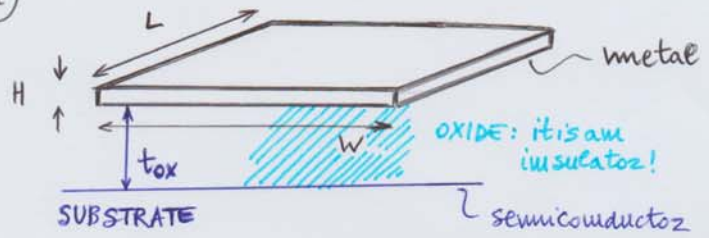
SOLUTION: FROM MANHATTAN STYLE ⇒ TO A MORE BROKEN-LINE LAYOUT (ROUNDED)

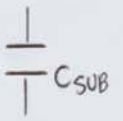


Explanation: high speed electrons choose always the minimum resistance path. This means higher temperature on the corner inside ⇒ Increase the risk of electromigration

CAPACITANCE 2 main contributions → // PLATE CAP. / FRINGE CAP.

(1)





modeled as a parallel plate cap.

$$C_{SUB} = \varepsilon_0 \varepsilon_r \frac{WL}{t_{ox}}$$

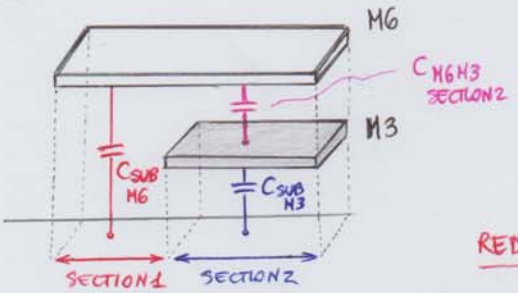
ε_r: SILICON DIOXIDE (SiO_2) DIELECTRIC CONSTANT = 3.9

W ↑ C ↑ R ↓

HIGH LEVEL M.L. t_{ox} ↑ C ↓ T_m ↑ R ↑

capacitance opposite w.r.t. resistance

Example:



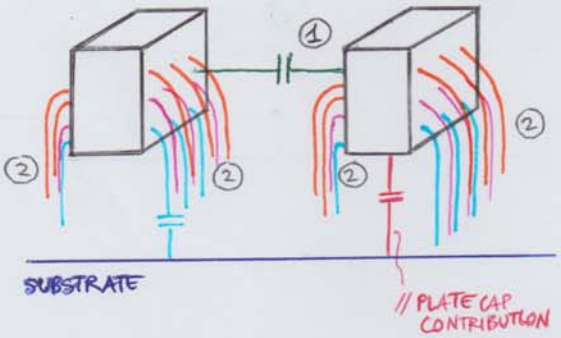
TOP VIEW

supposing $W_{M3} = W_{M6}$ ...

$$C_{M6M3} = \varepsilon_0 \varepsilon_r \frac{W_{M3} L_{SECTION2}}{t_{ox\,M6-M3}}$$

REDUCE CAP: a way is to reduce ε_r by using a different material, instead of SiO_2. There are some organics materials with lower ε_r, but they are more flexible ... interleave them with some thin stronger material, that does not introduce too much cap. ε_r ↓ K_{ox} ↑ TERMAL CONDUCTIVITY

(2) FRINGE CAP



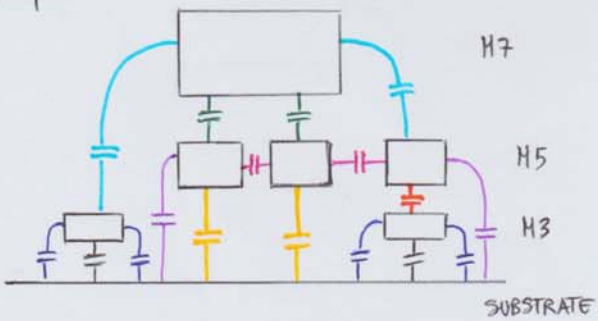
Due to electric field lateral component:

1. contribution from the side of the wire (conductor) to the side of nearby conductor (wire, usually)
2. contribution from the side of the wire to the substrate.

This happens a lot in buses, because they are long parallel wires (# depends on parallelism)

FAT WIRES = BIG HEIGHT = HIGH M.L.

Example:

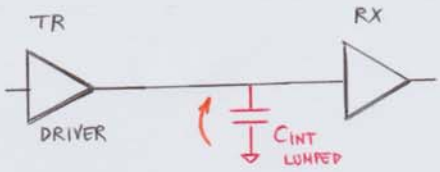


3

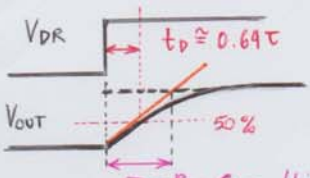
5/05/18

① LUMPED-RC

The capacitance is distributed across the line; however the model considers it as a concentrated capacitance. It neglects completely the resistance of the line



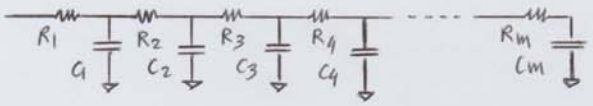
equivalent Thevenin circuit:



$$V_{OUT} = V_{DR}(1 - e^{-t/\tau})$$

$\tau = R_{DR} C_{INT}$ (time constant)

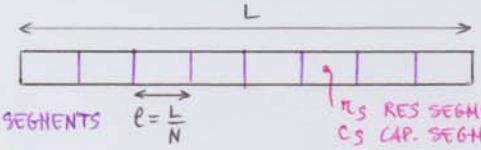
② ELMORE



50% DELAY

$$t_D \approx C_1 R_1 + C_2(R_1+R_2) + \dots C_m(R_1+R_2+\dots+R_m) =$$
$$= R_1(C_1+C_2+\dots C_m) + R_2(C_2+\dots+C_m) + \dots$$
$$+ R_m C_m$$

we can use this model to describe interconnects:



INTERCONNECT IN A INTEGRATED CIRCUIT

Divide the interconnection is N segment of length ℓ. Each one will be characterized by a certain resistance and capacitance:

$r_s = r\,\ell = r\,L/N$ (RES per unit length) $\quad c_s = c\,\ell = c\,L/N$ (cap per unit length)

We can apply now the elmore model:

$$t_D \approx c_s r_s + c_s(r_s + r_s) + \dots + c_s(\overbrace{r_s + \dots + r_s}^{N})$$
$$= c_s r_s + c_s(2 r_s) + \dots + c_s(N r_s)$$
$$= r c \frac{L^2}{N^2}(1+2+3+\dots+N) = r c \frac{L^2}{N^2}\frac{N(N+1)}{2}$$

if N is big $\Rightarrow N+1 \approx N$

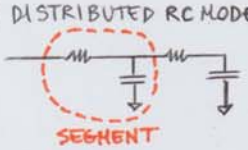
$$= \frac{rc}{2} L^2$$

DEPENDENCY ON TECH PARAMETERS

recall example 

SQUARE DEPENDENCY ON L

③ DISTRIBUTED RC MODEL



$\ell \to \Delta\ell$ $\quad r_s \to r\Delta\ell$ $\quad c_s \to c\Delta\ell$

solve differential equations:

$t_D \approx 0.38\, RC$ — TOTAL CAP, TOTAL RES

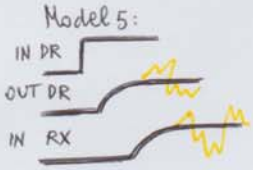
④ DISTRIBUTED RCL MODEL

$\ell \to \Delta\ell$ $\quad r_s \to r\Delta\ell$ $\quad c_s \to c\Delta\ell$ $\quad l_s \to l\Delta\ell$

⑤ TRANSMISSION LINE

Up to MODEL 4:

Model 5:

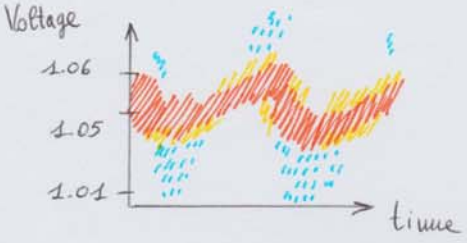


$$V_2 = V_1 \Gamma_{RX}$$

$$\Gamma_{RX} = \frac{z_{RX} - z_{LINE}}{z_{RX} + z_{LINE}}$$

EXAMPLE: measured supply noise from 90 nm Itanium Microprocessor

PT2 @ 1.30 (1 REG ONLY)



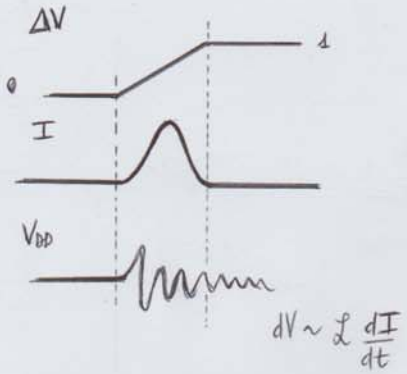
The Itanium µprocessor works with a 1.1 Volts power supply...

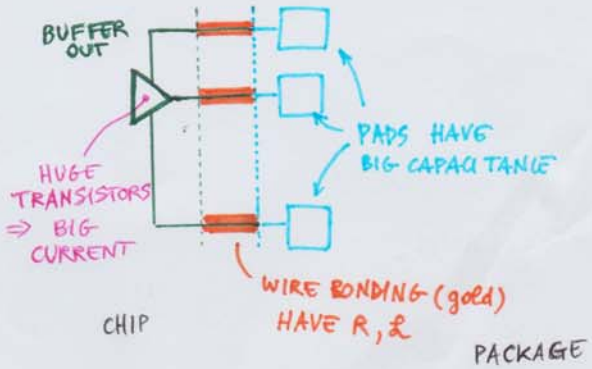
However the average voltage is around 1.054 V and in general power supply has a sinusoidal behaviour/progression...

There are also some sporadic variations on even lower voltage...

FINO A 1.37

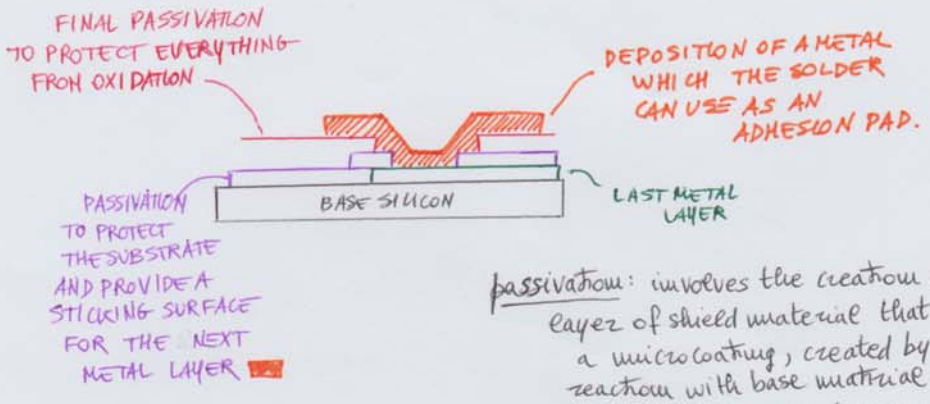
SSN: SIMULTANOUSLY SWITCHING NOISE





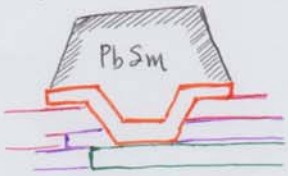
C4 process steps:

① metalization of contact pads:

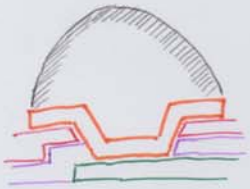


passivation: involves the creation of an outer layer of shield material that is applied as a microcoating, created by chemical reaction with base material, or allowed to build from spontaneous oxidation in the air.

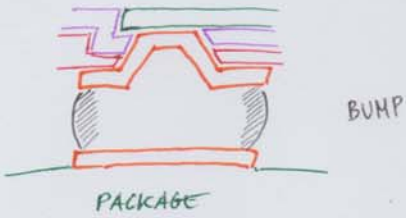
② solder deposition



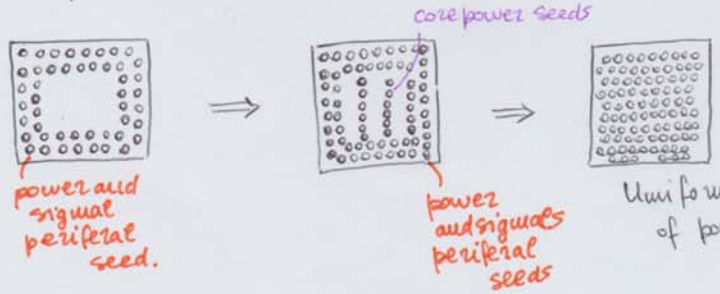
③ Reflow process: with heat, the solder melts into a ball



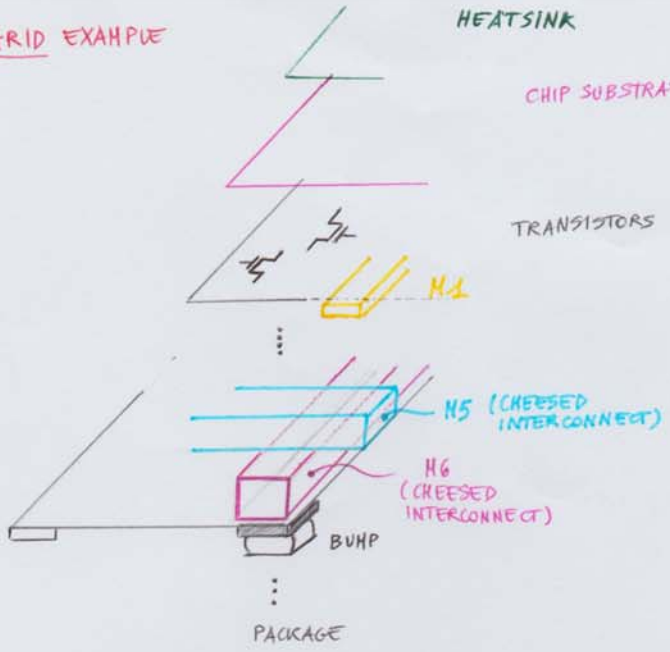
④ Bonding process: the chip is flipped, and connected to the package. (Alignment!!)



Historically:



POWER GRID EXAMPLE

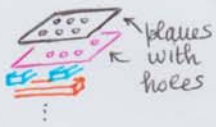


Usually the two highest metal layers have ⊥ directions and just carry Vdd and GND.

- 3 metal layer approach: at the beginning the metal 3 was used for power supply/clock routing. The organization was only one direction:

- 4 metal layer approach: in order to reach in a more precise way all parts of the circuit, a metal layer M4 is introduced, ⊥ with respect to M3. It also give more spots to put power supply pads. Still M3 and M4 mostly used for power supply / clock. Usually more than 50% of total metal is used for Vdd/Gnd. } POWER GRID

- 6 metal layer approach (planes). Instead of two Metal layers dedicated to Vdd and Gnd, two solid planes are used. This reduce a lot the resistance! Also, because there are less straight thin metal line, also chip inductance is reduced. This approach however is very expensive.



- Low cost, low frequency applications: routed power distribution on two stacked layers of metal (one for VDD, one for GND).

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 97 di 114