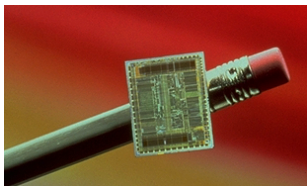


TELECOM  
ParisTech



INSTITUT  
Mines-Télécom



# ICS904/EN2 : Design of Digital Integrated Circuits

L3 : Structural design of digital  
circuits(2/2)

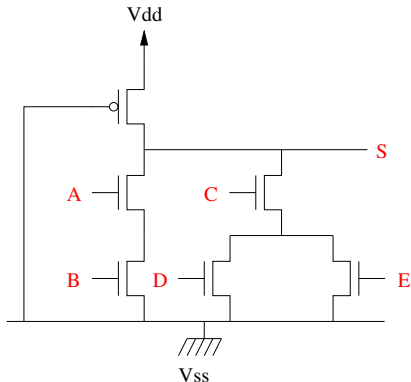
Yves MATHIEU

yves.mathieu@telecom-paristech.fr

université  
PARIS-SACLAY

# From NMOS logic to dynamic logic

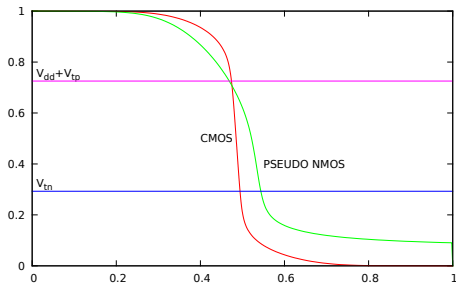
## pseudo-NMOS logic



- Replacement of the PMOS network by a passive load.
- Smaller : The "1" values of the truth table are implicit values.
- CONFLICT : When NMOS network is ON : steady-state current.

# From NMOS logic to Dynamic logic

## pseudo NMOS-logic



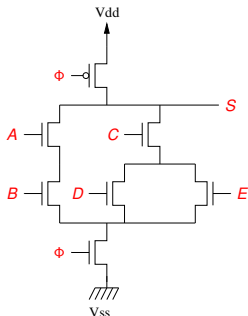
### ■ PMOS/NMOS width balancing :

- Low output level should be less than threshold voltage of NMOS transistor.
- Propagation time for output rising edge should be kept small.

### ■ How to avoid a choice between speed and low-power/robustness.

# Dynamic logic

## Precharge logic



- Only one NMOS network.
- One clock (synchronous context)
- $\phi = 0$  : Output is precharged to 1 (Precharge phase)
- $\phi = 1$  : Conditional computation of the output. (Evaluation phase)
- State "1" is a high impedance state.
- Leakage current of transistors limits the minimum clock frequency (state "1" disappears ...).



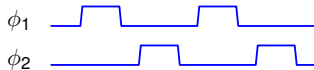
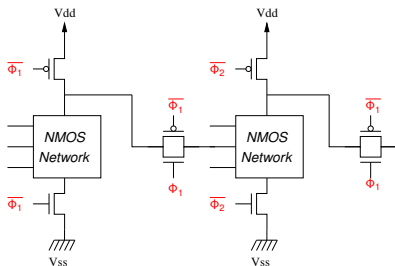
# Dynamic logic

## Constraints

- If the final state is "1", the output should stay to "1" during evaluation phase.
- So inputs should be stable during Evaluation phase.
- Then output of such cell can not be used as inputs of a another cell . . .
- Even if we meet constraints, final output voltage may be less than  $V_{dd}$  (charge sharing inside the NMOS network)

# Dynamic logic

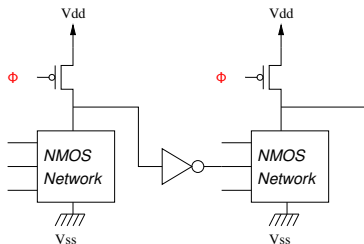
## Two phases dynamic logic



- Two clocks with non overlapping phases.
- Cell isolation using pass transistor logic and acting as a register.
- Fully cascadable, but odd and even cells should alternate
- Deep pipelining (no more than one gate level between each pipeline register)

# Dynamic logic

## Domino logic

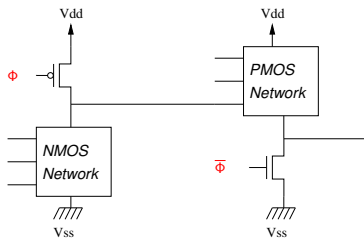


- During precharge phase : All gates inputs are "0" (all NMOS networks are OFF)
- During evaluation phase : some inputs switch to "1"
- Then some NMOS networks switch to ON state.
- Then some gate outputs switch to "0"
- Then some gate inputs switch to "1"
- ...
- $T_{cycle} > \sum T_{propagation}$

■ Warning : only non-inverting gate can be implemented

# Dynamic logic

## Domino N-P logic



- Simplified : no more inverter
- Warning : only inverting gates can be implemented
- Warning : NMOS gates should alternate with PMOS gates...





# Dynamic logic

## Summary

- Very high speed logic (very small capacitive loads)
- Very often used during 80-90 years
- Example : High speed carry chain for arithmetic computation in microprocessors
- But : Clock is loaded by all cells... (power consumption of the clock network)
- But : Precharge phase is wasted time.
- But : Complex design automation (no direct RTL synthesis tool)
- Only used for optimized "full custom" designs.

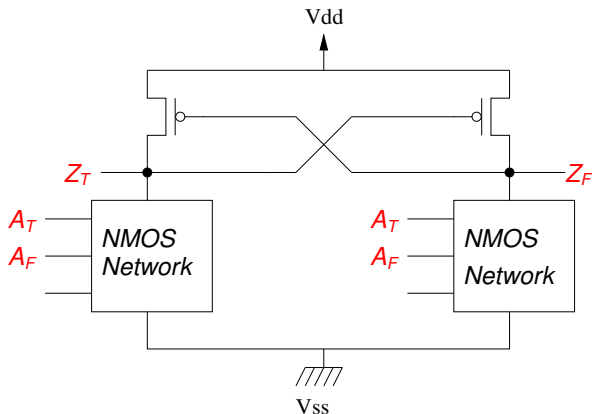
# Differential logic

## Introduction

- All signals are duplicated :  $A \Rightarrow (A_T, A_F)$
- Parallel computation of  $F$  and  $\bar{F}$
- No inverter needed
- Reduces complexity of arithmetic computation.
- 2 NMOS transistors networks of equal size.
- No limitation to inverting or non inverting gates.
- Less number of gates, but more wires . . .

# Differential logic

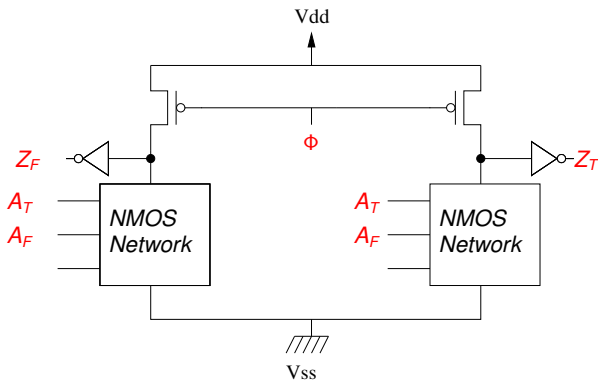
## Cascode Voltage Switch Logic



- Warning : conflicts during output transitions.
- Slower than CMOS logic.
- The two networks can share transistors

# Differential logic

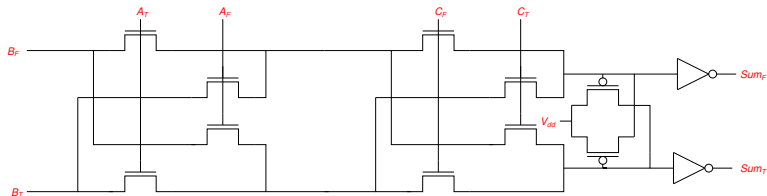
## DCVSL : Dynamic Cascode Voltage Switch Logic



- No conflict during output transitions.
- **Q1** : Design a CVSL 2 inputs XOR gate. Try to minimize the number of transistors.

# Differential logic

## CPL : Complementary Pass Logic



- Pass transistor logic using only NMOS : slow degraded logic one but ...
- "Weak" PMOS pullups restore full scale swing and outputs are buffered by CMOS invertors
- Using *low*  $V_t$  transistor for the network, and *high*  $V_t$  transistors for the invertors helps speed optimization
- Said to be one of the fastest logic style ...



# Differential logic

## Application specific logic styles

- Current Mode Logics
- Adiabatic Logic
- SubTreshold Logic
- ...



# Performance metrics

## Introduction

- Area of the logic gates.
- Speed of the logic gates.
- Power consumption of the logic gates
- Noise margin of the logic gates
- EDP : "Energy Delay Product" of the logic gates.
- Near threshold or Sub-threshold behavior.
- Robustness of the design.
- ...

# Buffer optimization example

## simple inverter

- Problem definition : What is the fastest way to transmit a data from the input of gate A, to the inputs of gates connected to A ?
- The timing model of gate A is known :  $T_{pA} = T_{p0A} + R_A \cdot C_{load}$
- The inputs of the gates connected to A are modeled by a load capacitor  $C_{LdA}$
- A parametrized inverter can be used :
  - $T_{pIV}(\alpha) = T_{p0IV} + (R_{0IV}/\alpha) \cdot C_{load}$
  - $C_{InIV}(\alpha) = C_{0InIV} \cdot \alpha$
  - with  $\alpha \geq 1.0$
- The inverter is inserted between gate A and the other gates.
- Compute the propagation time through the gates  $\alpha$
- Compute the value of  $\alpha$  giving the minimum propagation time.
- Compute the value of the minimum propagation time.



# Buffer optimization example

## several invertors

- The inverter is replaced by two successive invertors with parameters  $\alpha_0$  and  $\alpha_1$ .
- What are the optimal sizes of  $\alpha_0$  and  $\alpha_1$  for a minimum propagation time ?
- Compute the value of the minimum propagation time.
- The two invertors are replaced by  $N$  successive invertors with parameters  $\alpha_0 \dots \alpha_{N-1}$ .
- What are the optimal sizes of the  $N$  parameters  $\alpha_j$  for a minimum propagation time ?
- Compute the value of the minimum propagation time.
- Compute the value of  $N$  that minimize the propagation time
- Compute the value of the minimum propagation time.



# Buffer optimization example

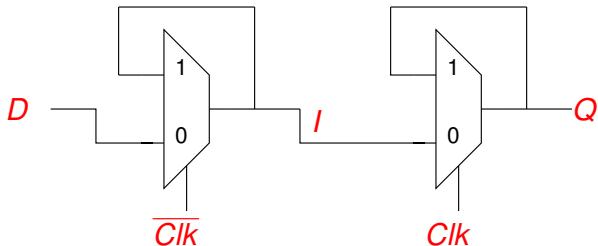
## How to transmit a data on a long wire

- The long wire as a distributed RC model.
- Distributed invertors along the line may help minimizing overall propagation time.
- Same kind of optimization but with a non linear model of the propagation time through the line



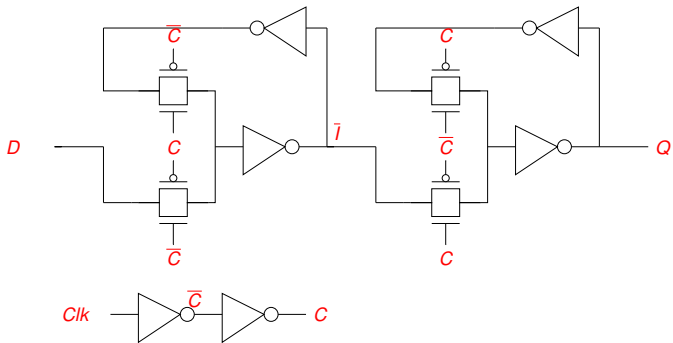
# positive edge-triggered D flip-flop

## Theoretical Master/Slave flip-flop



- Muxes with loops define 2 storage elements (Master and Slave)
- Master(resp. Slave) hold its value while Slave (resp. Master) is transparent.
- Skew between the two clocks should be avoided (race condition).
- D signal should respect timing conditions.
  - Setup time / Hold time

## practical design



- Clocks are internally generated in order to avoid unwanted skew.
- Warning  $D$  input is on the Drain of a transistor
- One can mix CMOS logic and pass transistor logic (using tristate inverters).
- **Q1 : Design a D flip/flop using tristate invertors**

# "Standard Cells" versus "Full Custom"

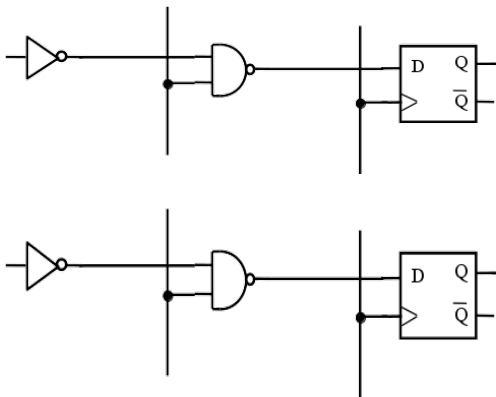
## Full Custom

- Manual layout of all needed cells.
- Long elec. simulations for verif. of a whole block.
- Wide logic styles choice.
- Scripting may help layout phases.
- Ultimate optimisation for speed power or area.
- Only for high value added digital or analog blocs.

## Standard Cells

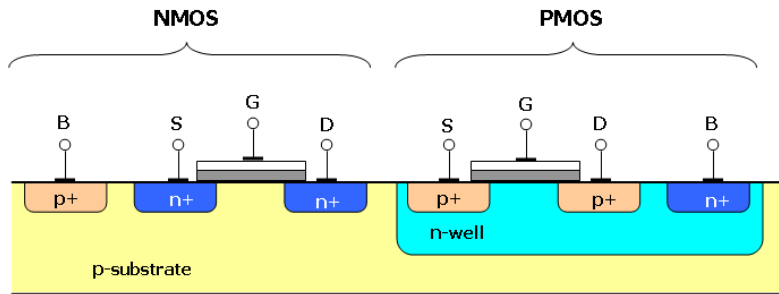
- Layout design limited to generic cells.
- Electrical simulation for cell properties extraction.
- Small logic styles choice.
- Automation of synthesis, place and route phases.
- Suboptimal for speed power and area.
- When time-to-market is the main criterion.

## Standard cell principles



- All cells have same height
- Power supply and Ground connected by abutment.
- Cell design should be free of DRC error.
- Any abutment of any couple of cells should be free of DRC error.
- Wiring inside cell limited to Metal1 level.

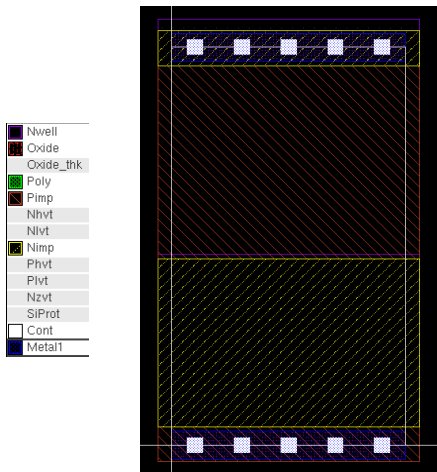
# Reference technology profile





# gpdk045 standard cell template















## practical design

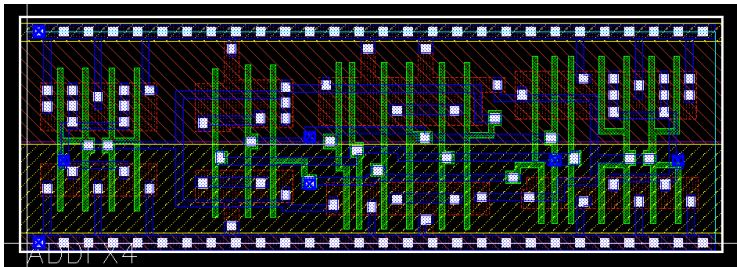


- NMOS areas and PMOS areas already filled.
- Body-ties areas for NMOS and PMOS already filled
- Body-ties already connected to  $V_{dd}$  (for PMOS) or  $V_{ss}$  (for NMOS)
- Simple abutment of cells fill an raw of cells with NMOS and PMOS areas.

# gpdk045 adder

## practical design















|   |           |
|---|-----------|
|  | Nwell     |
|  | Oxide     |
|  | Oxide_thk |
|  | Poly      |
|  | Pimp      |
|  | Nhvt      |
|  | Nlvt      |
|  | Nimp      |
|  | Phvt      |
|  | Plvt      |
|  | Nzvt      |
|  | SiProt    |
|  | Cont      |
|  | Metal1    |

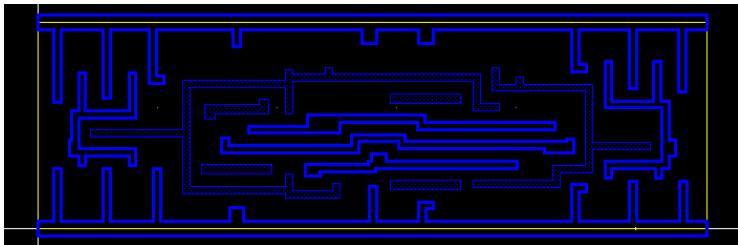


- All transistors have horizontal orientation.
- Maximal width defined by NMOS and PMOS areas height.
- Use parallel transistors for larger widths.
- Drain/Source implants may be used for local short wires (beware the resistivity).
- Global optimisation of Eulerian Paths (N(P)MOS subcircuits are graphs which visits every edge exactly once)

# gpdk045 adder abstract

## practical design

|   |           |
|---|-----------|
|  | Nwell     |
|  | Oxide     |
|  | Oxide_thk |
|  | Poly      |
|  | Pimp      |
|  | Nhvt      |
|  | Nlvt      |
|  | Nimp      |
|  | Phvt      |
|  | Plvt      |
|  | Nzvt      |
|  | SiProt    |
|  | Cont      |
|  | Metal1    |

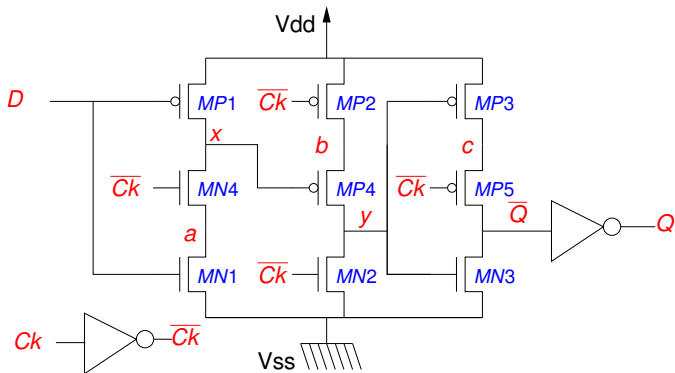


- Only needed informations for Place and Route.
- Wires connected to Input/Output pins of the cell
- Wires that are obstacles for wiring
- The router may use Metal1 has a wiring layer if enough room inside the cell



# Dynamic positive edge-triggered D flip-flop

## TSPC : True Single Phase Clock logic

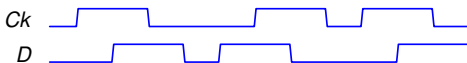


- Very compact structure.
- Supposed to be faster than standard CMOS cell.

# Dynamic positive edge-triggered D flip-flop

## Homework

- Question : How does this cell works ?
- Setup a table D, Q, Ck and all internal nodes of the cell.
- Add also the states of the MOS transistors (ON or OFF).
- Each internal node may have the states 0, 1, U(Unknown), Z0 (high-impedance 0), Z1 (high-impedance 1), or ZU (high-impedance Unknown).
- Imagine the following input sequence.



- Fill the table with the successive values of the signals
- Check if the cell is a D flip-flop.