

$$18 = 001\ 0010 = 0x12$$

2

Chronogramme.

Table de vérité

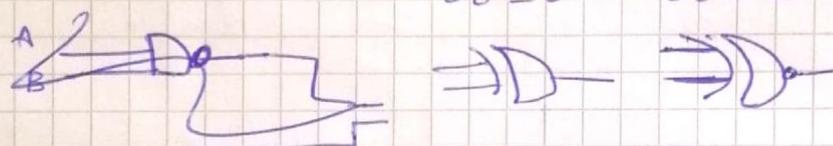
Exemple AND

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

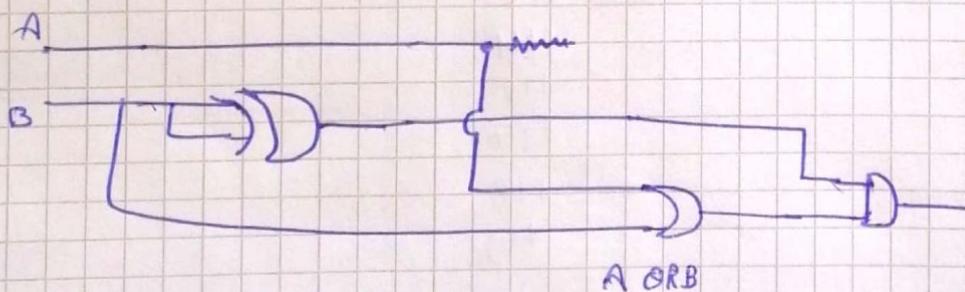
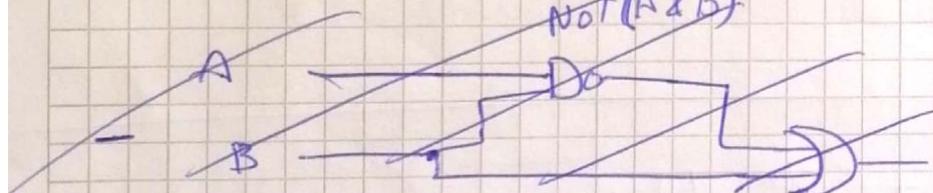
$$A = 0 ; B = 1$$

$XOR \neq XNOR$

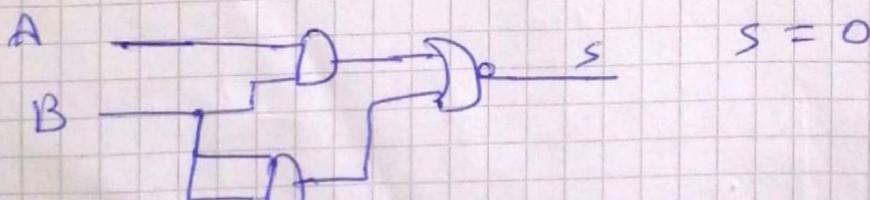
$$00 = 0 \quad 00 = 1$$



$\text{NOT}(A \& B)$

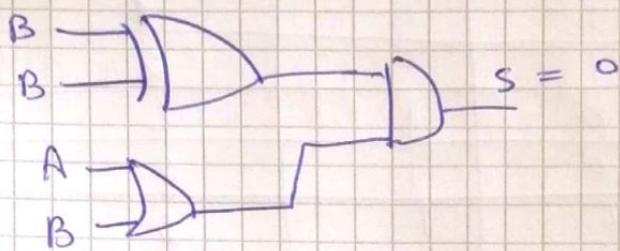


$$\text{Not} [(A \text{ AND } B) \text{ OR } (B \text{ AND } B)]$$



$$S = 0$$

$(B \text{ XOR } A) \text{ AND } (A \text{ OR } B)$



$A + B$ en boolean

$$so = A \text{ XOR } B$$

$$\text{return} = A \text{ AND } B$$

A	B	result	return
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

A (b ₂)	B (b ₂)	S (b ₃)	S[0]	S _b (10)
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	0	2

↑
return
↓
~~so~~

~~so~~

10
2² ~~2²~~

return \equiv carried bit

return \equiv ~~so~~ carried bit

A —

B —

Cin —

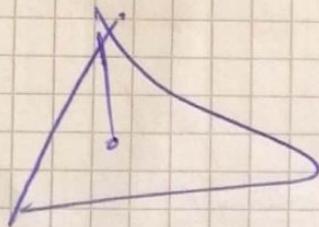
$$\text{cout} = A \text{ AND } B$$

A (b ₂)	B (b ₂)	Cin(b ₂)	cout	S[0]	S _b
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	1
1	1	0	1	1	0
0	0	1	1	0	1
1	0	1	1	1	0
0	1	1	1	1	1
1	1	1	1	1	1

A	A(b ₂)	B(b ₂)	Cin(b ₂)	S(b ₁₀)	Carry SΣB	Σ ₀
0	0	0	0	0	0	0
1	0	0	0	1	0	1
0	1	0	0	1	0	1
1	1	0	0	2	1	0
0	0	1	0	2	0	1
1	0	1	0	2	1	0
0	1	1	0	2	1	0
1	1	1	1	3	1	1

le full adder

TD: Réaliser un circuit



Rapport de nos à empêcher en fin de module + tout à faire dans la partie Ajc dans exercices

- * passage de la table de vérité vers le schéma full-adder
- * analyse architecture

NiVado

25/04/2023

slides 21

ports logique

XOR

$$X = A \oplus B \quad 0 \oplus 0 = 0 ; 1 \oplus 1 = 0$$

XNOR

$$X = \overline{A \oplus B}$$

NAND

$$X = \overline{A \cdot B}$$

NOR

$$X = \overline{A + B}$$

Slide 24

$$C_0 = (\bar{A}, B, C_{in}) + (A, \bar{B}, C_{in}) + (A, B, \bar{C}_{in}) + (A, \bar{B}, \bar{C}_{in})$$

$$S = (\bar{A} \cdot \bar{B} \cdot C_{in}) + (\bar{A} \cdot B \cdot \bar{C}_{in}) + (A \cdot \bar{B} \cdot \bar{C}_{in}) + (A \cdot B \cdot C_{in})$$

Factorisations:

$$C_o = C_{in} (\bar{A} \cdot B + A \cdot \bar{B}) + A \cdot B (\underbrace{\bar{C}_{in} + C_{in}}_1)$$

$$(\bar{A} \cdot B + A \cdot \bar{B}) \Leftrightarrow A \vee B$$

$$\overline{A \cdot B} + A \cdot \overline{B} = A \text{XNOR} B$$

$\vdash s \vdash \overline{\overline{A \oplus B}}$

$$C_{in}(\overline{A \oplus B}) + \overline{C_{in}(A \oplus B)}$$

A	B	
0	0	1
0	1	0
1	0	0
1	1	1

table de Karnaugh (K map)

1 entrée \Rightarrow on a 2¹ cas diff' de sortie

b	a	X
0	0	L
0	1	L
1	0	1
1	1	0

x	$\text{Not}(a)$	a
$\text{Not}(b)$	1	1
b	-1	0

done to report is NOT bona fide

$$\overline{A} \times \overline{B} = \overline{A \cdot B}$$

$$\begin{aligned}
 S &= \bar{A} \cdot \bar{B} + \bar{A} \cdot B + A \cdot \bar{B} \\
 &= \bar{A} (\bar{B} + B) + A \cdot \bar{B} \\
 &= \bar{A} + A \cdot \bar{B} = \underbrace{(\bar{A} + A)}_1 (\bar{A} + \bar{B}) \\
 &= \bar{A} \cdot 1
 \end{aligned}$$

$$\begin{array}{cc}
 0 & 0 \\
 \bar{A} \cdot \bar{A} = 0 \\
 1 & 1
 \end{array}$$

$$\underbrace{\bar{A} \cdot \bar{A} + \bar{A} \cdot \bar{B} + A \cdot \bar{A}}_{\bar{A}} + \underbrace{A \cdot \bar{B}}_0$$

$$\begin{aligned}
 &\bar{A} + \bar{A} \cdot \bar{B} + A \cdot \bar{B} \\
 &\bar{A} (1 + \bar{B}) + A \cdot \bar{B}
 \end{aligned}$$

A	$\bar{B} \cdot \bar{C}_{in}$	$\bar{B} \cdot \bar{C}_{in}$	$\bar{B} \cdot \bar{C}_{in}$	$B \cdot \bar{C}_{in}$	$B \cdot \bar{C}_{in}$
\bar{A}			1		1
A	1			1	

diagonal \Rightarrow XOR

$$S = A \cdot \bar{B} \cdot \bar{C}_{in} \text{ XOR } \bar{A} \cdot \bar{B} \cdot C_{in} \text{ XOR } A \cdot B \cdot C_{in} \text{ XOR } \bar{A} \cdot B \cdot \bar{C}_{in}$$

$$S = C_{in} \oplus A \oplus B$$

$$A \oplus B = A\bar{B} + A.B$$

$$\underbrace{A \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + A \cdot B \cdot C + \bar{A} \cdot B \cdot \bar{C}}$$

$$A \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C$$

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

$$\Rightarrow \cancel{A \cdot \bar{B} \cdot \bar{C}} \cdot \overbrace{[A + B + \bar{C}]}^{\cancel{A}} = \cancel{A \cdot A \cdot \bar{B} \cdot \bar{C} + B \cdot A \cdot \bar{B} \cdot \bar{C} + \bar{C} \cdot A \cdot \bar{B} \cdot \bar{C}}$$

$$= \cancel{A \cdot \bar{B} \cdot \bar{C}} + A \cdot \bar{B} \cdot \bar{C}$$



$$= (\bar{A} + B + C) (\bar{A} \cdot \bar{B} \cdot C) + A \cdot \bar{B} \cdot \bar{C} \cdot (A + B + \bar{C})$$

$$\cancel{\bar{A} \cdot \bar{A} \cdot \bar{B} \cdot C} + B \cdot \cancel{\bar{A} \cdot \bar{B} \cdot C} + C \cdot \cancel{\bar{A} \cdot \bar{B} \cdot C} + A \cdot \bar{B} \cdot \bar{C} \cdot A + A \cdot \bar{B} \cdot \bar{C} \cdot B + A \cdot \bar{B} \cdot \bar{C} \cdot \bar{C}$$

$$= \cancel{\bar{A} \cdot \bar{B} \cdot C} + C \cdot \cancel{\bar{A} \cdot \bar{B}} + A \cdot \bar{B} \cdot \bar{C}$$

$$= \cancel{\bar{A} \cdot \bar{B} \cdot C} + A \cdot \cancel{\bar{B} \cdot \bar{C}} = \cancel{\bar{B}} (\cancel{A \cdot C} + A \cdot \bar{C})$$

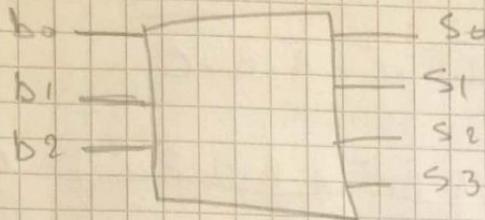
$$= \cancel{\bar{B}} (A \text{ XOR } C)$$

$$A \cdot B \text{ XOR } B \cdot C = \bar{A} \cdot \bar{B} \cdot B \cdot C + \bar{B} \cdot C \cdot A \cdot B$$

$$= \cancel{\bar{A} \cdot B \cdot C} + \cancel{\bar{B} \cdot B \cdot C} + \cancel{\bar{B} \cdot A \cdot \bar{B}} + \bar{C} \cdot A \cdot B$$



proposer un circuit réalisant l'opération de multiplication par 2 d'une entrée seu 3 bits, la sortie sera donnée sur 4 bits (faire un arbre à droite inférieur)



b2/b1/b0	s3	s2	s1	s0
0 0 0	0	0	0	0
0 0 1	0	0	1	0
0 1 0				
0 1 1				
1 0 0				
1 0 1				
1 1 0				
1 1 1				

PV: process unité

Memory:

Cpu:

Cpu: cache (multithreading)
multicore

arduino: microcontrôleur (pas de cache)

Clipset: pu et mémoie

synchrone → asynchrone

analyse des flux critiques

BRA M ! Embedded RAM

Volatile mémoire temporaire

BRA M - L - Y304150

c'est un espace pour sauvegarder des données temporaires
elle perd les données si on coupe le courant

USB Embarquée : par exemple, utilisé pour l'Etanol, USB

mémoire volatile

// non volatile

B RAB

Cellules logiques

CPU est incapable de calculs logiques



Le CPU gère l'ensemble des tâches nécessaires au bon fonctionnement de tous les logiciels du serveur.

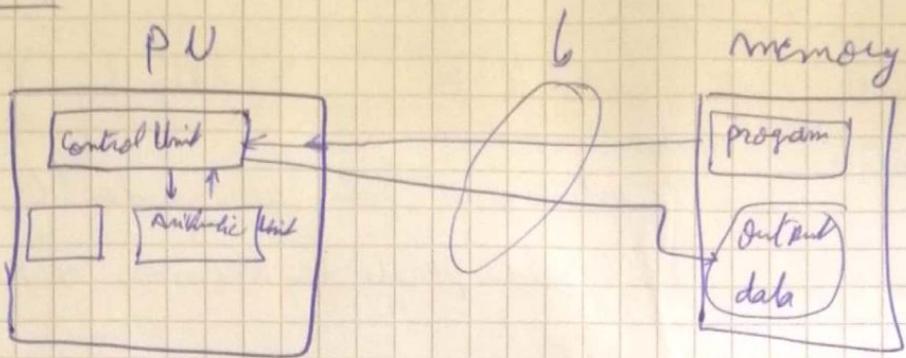
Le CoCPU permet au CPU d'effectuer des calculs simultanés.

PU peut être ~~CPU + GPU~~ contient de CPU ou de GPU
des deux, peut être de microcontrôleur
graphique

CoCPU : multithread (traité en parallèle ; parallelisation des tâches)

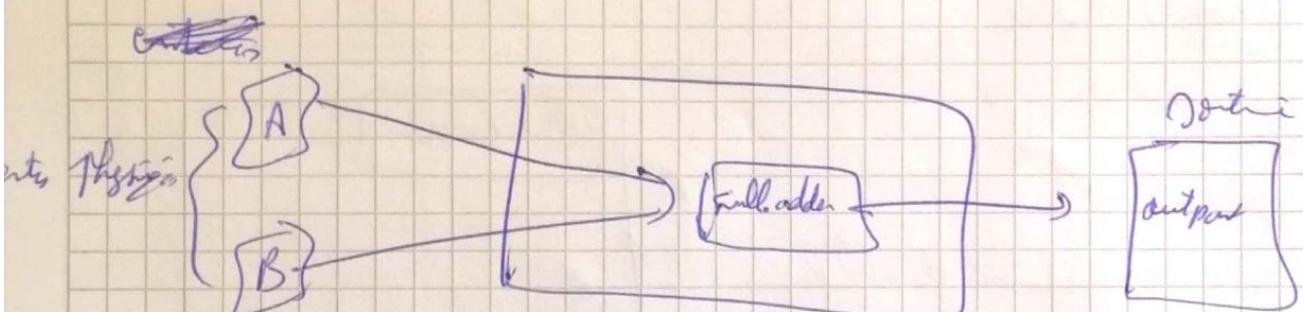
CPU : traité séquentiel

Cadence / Cet Hz : le processus peut travailler jusqu'à 1 GHz



On voit ça sur le slide 61 sur la carte raspberry PI

sur l'FPGA, on n'a pas ça : voir slide 83



là on a pas besoin de PU

HDL : Hardware Description language (HDL)

Description des entrées, sorties et du comportement d'une architecture numérisée

$$\begin{array}{c} A \\ \text{---} \\ B \end{array} \quad \text{LD} \quad \begin{array}{c} C \\ \text{---} \\ D \end{array} = C \leftarrow A \text{ and } B$$

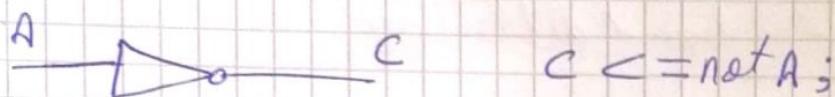
VHDL : nous permet de simuler le code sur le logiciel avec (haut niveau) de l'implémenter sur la carte

Vhdl : plus bas niveau que le VHDL

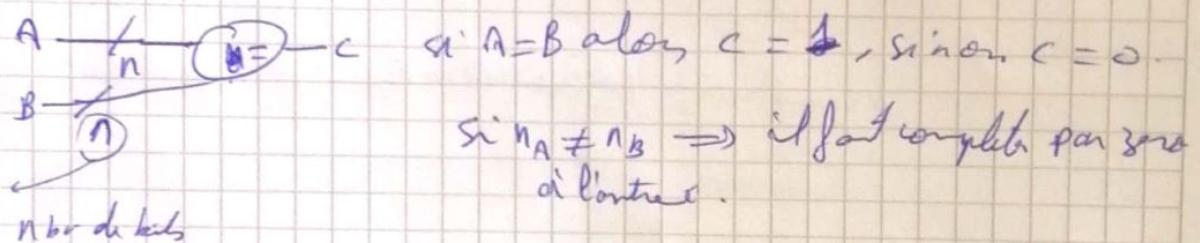
~~Théma 27.10~~

On ne peut pas modifier l'entrée, on utilise des signaux internes.

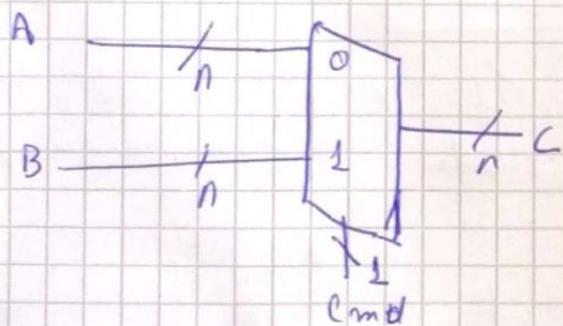
- * Std-logic ; c'est type ambig (0 ou 1)



Sémant RTL (avec lessy bits)



Les multiplexeurs:



Si Cmd = 0 alors $C = A$; si Cmd = B;
là on a que deux entrées qui sont A et B;
si on a plus de sorties \Rightarrow Cmd est codé sur plusieurs bits

* logique séquentielle :

* * horloge: signal périodique . état haut état bas

* la logique combinatoire: \rightarrow les rats logiques

* instruction concurrentes

* instruction séquentielles;

process (clk, resetn)

i. il n'interrrompt pas les autres process
quand il se ~~re~~ réveille.

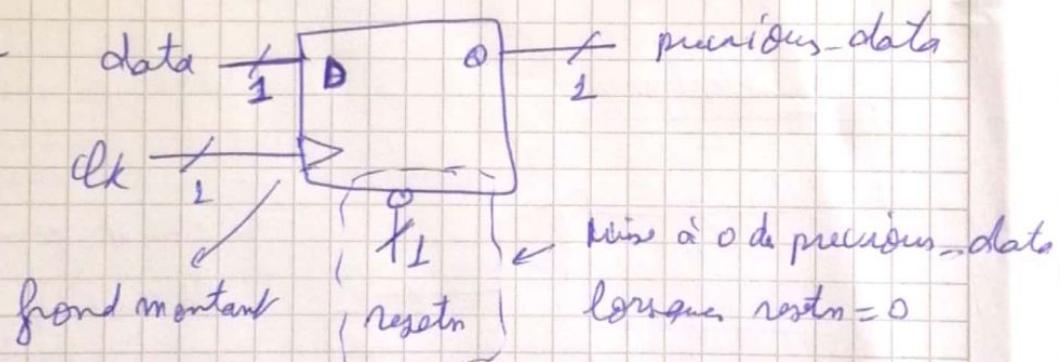
synchrones parce qu'il dépend de l'horloge

if (resetn = 0) then

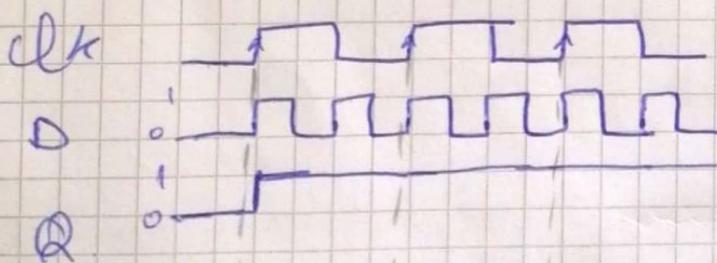
asynchrones parce qu'il ne dépend pas de l'horloge.

Les registres :

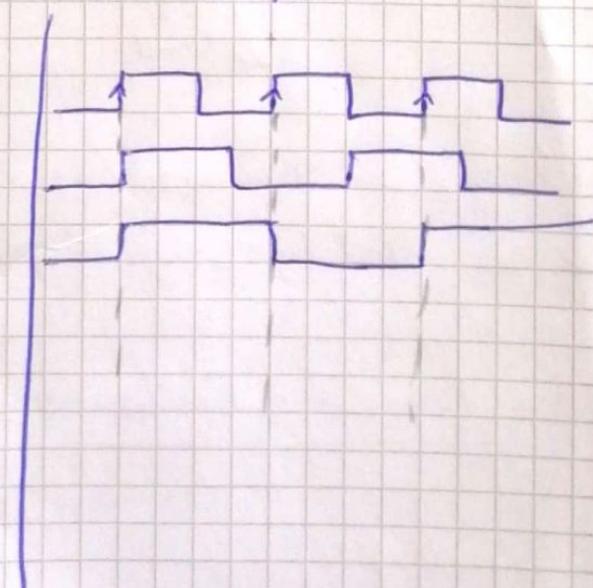
slide II:



Expl 1



Expl 2



macaron à état:

process synchrone \neq process combinatoire

lorsq'on change l'entrée
la sortie sera mise
à jour automatiquement

décalage:

output <= bit_msb & data[n-1:1];

0

bit n° 2
~~bit 1~~
bit n° n-1

n! bit de bits

data [n-2:0]

Synthèse VHDL:

on remet pas ; à la fin de la dernière ligne

Bitstream: une technologie de conversion des signaux analogiques en signaux numériques et de signaux numériques en signaux analogiques.

27/04/2023

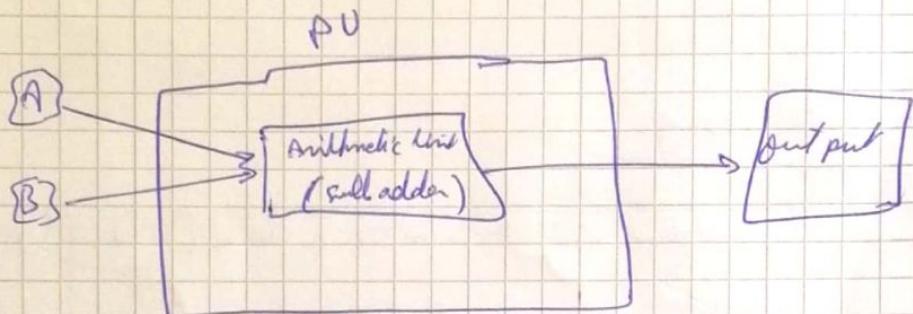
HDL coder

→ Verilog → VHDL

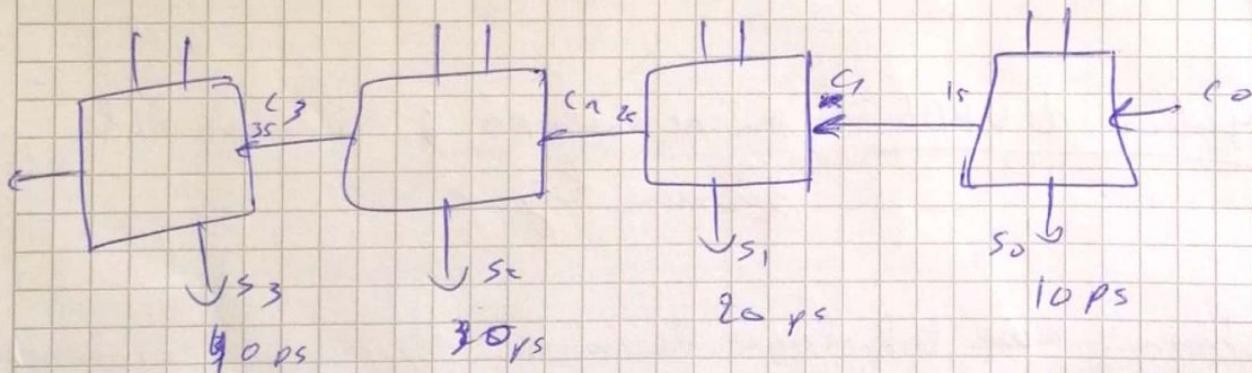
pour accéder à la carte, de codage

Lattice ice 40

yo sy.



* Le shamin critique : le plus long. (qui prend le plus de temps)



la net list

* le code de testbench ne sera jamais implementé physiquement sur la carte, il est seulement pour la simulation du test comportemental.

~~ce qui s'implante à l'entrée~~ le code dans le fichier full-adder-HDL

Le fichier de constraints : c'est pour indiquer à l'utilitaire `lava`, les interconnexions entre les cellules qui se trouvent dans la carte + avoir contrainte de timing (pour minimiser la latence) ...

Une slice : c'est que toutes les portes logiques sont regroupées dans un block qu'on l'appelle slice.

~~Logic~~

lut :

laboires de routage ; c'est pour rajouter un peu de latence

~~slice~~ - slice = logic cell

o 5 à 30ns sans cache 4ns

Circuit 2 slices par芯

pic de consommation. Vient ~~lorsque~~ le parage -

les deux mémos ne réagissent pas en même temps ~~donc~~ et ya un moment où la latence à due à la marre. parce que les deux Mémos sont finies en même moment.

MTM2

on n'appelle pas compilation on l'appelle synthèse

RAM ≠ ROM

↑

↓

mémoire
vite

mémoire morte

↓

données figées

données
non configurables

notre carte RAM

Cora Z7: basée sur SRAM \Rightarrow RAM volatile

\Rightarrow donc on va être en volatile avec notre carte Cora Z7

* On peut

bitstream \rightarrow bits

