Axel François-Charlot Mardi 8/12

3ETI – Groupe C

T.P. n°2 : Premiers pas avec le langage VHDL et l'outil de conception ISE (Xilinx)- Conception d'un système d'addition et de soustraction de deux nombres

Additionneur

Dans la Final Report, on constate que les portes utilisées sont les suivantes :

```
Design Statistics
# IOs                              : 5

Cell Usage :
# BELS                             : 5
#      AND2                        : 2
#      XOR2                        : 3
# IO Buffers                       : 5
#      IBUF                        : 3
#      OBUF                        : 2
```

1. Vue technologique

Voir pdf

On utilise ce schéma technique pou

L'équation correspondante au termes de somme est :

$$S = data_A \oplus data_B \oplus carry_{in}$$

L'équation correspondante au termes de retenue est :

$$R = data_A.data_B \oplus (carry_{in} . (data_A \oplus data_B))$$

On constate que l'équation du terme de sommé est rigoureusement identique à celle obtenue pour le travail tuteuré. En revanche, l'équation du terme de retenue est légèrement différente puisque nous avions obtenu :

$$R = data_A.data_B + carry_{in} . (data_A \oplus data_B)$$

Pour expliquer cette différence il nous faut montrer que l'équation suivante est vrai :

$$X \oplus Y = X + Y$$

Où $X = data_A.data_B$

Et $Y = carry_{in} . (data_A \oplus data_B)$

On a donc :

$$X \oplus Y + X.Y = X + Y$$
$$\rightarrow X.Y = 0$$

C'est-à-dire, il faut montrer : $(data_A.data_B).\left(carry_{in}.(data_A \oplus data_B)\right) = 0$

$$\rightarrow (data_A.data_B).\left(carry_{in}.\left(\overline{data_A}.data_B + data_A.\overline{data_B}\right)\right) = 0$$

$$\rightarrow carry_{in}.(data_A.data_B).\left(\overline{data_A}.data_B + data_A.\overline{data_B}\right)$$

$$\rightarrow carry_{in}.(data_A.data_B.\overline{data_A}.data_B + data_A.data_B.data_A.\overline{data_B})$$

Or $data_A.data_B.\overline{data_A}.data_B = 0$

Et $data_A.data_B.data_A.\overline{data_B} = 0$

Car $A.\overline{A} = 0$

Donc $carry_{in}.\left(data_A.data_B.\overline{data_A}.data_B + data_A.data_B.data_A.\overline{data_B}\right) = 0$

Donc $(data_A.data_B).\left(carry_{in}.(data_A \oplus data_B)\right) = 0$

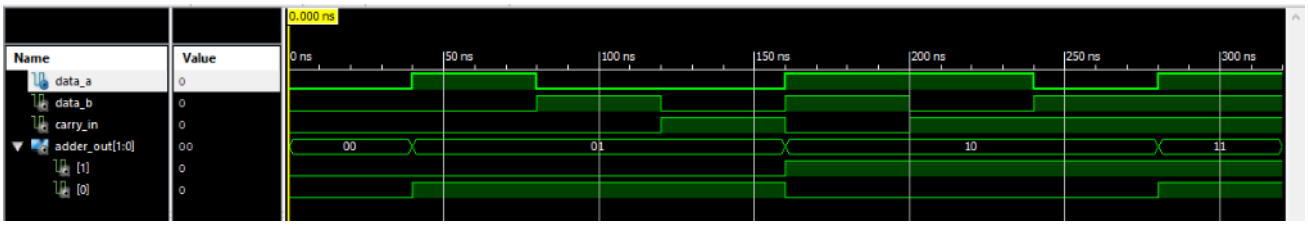
On a donc bien égalité entre les 2 équations. L'outil de synthèse est donc très performant et permet d'optimiser encore plus les résultats obtenus pendant le travail tuteuré.

2. Simulation temporelle Behavioral

Les signaux d'entrées utilisés pour la simulation sont les suivants :

```
data_A <= '0', '1' after 40 ns, '0' after 80 ns, '1' after 160 ns, '0' after 240 ns, '1' after 280 ns;
data_B <= '0', '1' after 80 ns, '0' after 120 ns, '1' after 160 ns, '0' after 200 ns, '1' after 240 ns;
carry_in <= '0', '1' after 120 ns, '0' after 160 ns, '1' after 200 ns;
```

Voici le résultat de la simulation temporelle Behavioral :



Ce chronogramme permet de tester toutes les entrées possibles du composant. Retranscrit en table de vérité, les résultats sont les suivants :

A	B	Retenue in	Retenue	Somme
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1
1	1	0	1	0
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

La table de vérité obtenue lors du travail dirigé était :

A_i	B_i	R_{i-1}	R_i	S_i
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1
1	1	0	1	0
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

Le bon fonctionnement de l'additionneur est donc validé !

3. Implémentation, de l'additionneur élémentaire

Le tableau résumé des ressources utilisées est :

RESOURCES SUMMARY

Macrocells Used	Pterms Used	Registers Used	Pins Used	Function Block Inputs Used
2/64 (4%)	6/224 (3%)	0/64 (0%)	5/33 (16%)	3/160 (2%)

PIN RESOURCES

Signal Type	Required	Mapped	Pin Type	Used	Total
Input	3	3	I/O	3	26
Output	2	2	GCK/IO	0	3
Bidirectional	0	0	GTS/IO	2	4
GCK	0	0	GSR/IO	0	1
GTS	0	0	DGE/IO	0	-1
GSR	0	0			

GLOBAL RESOURCES

Global clock net(s) used	0
Global output enable net(s) used	0
Global set/rest net(s) used	0

Les équations réellement utilisées pour réaliser la fonction sont :

```
********** Mapped Logic **********

adder_out(0) <= data_A
    XOR ((data_B AND NOT carry_in)
    OR (NOT data_B AND carry_in));

adder_out(1) <= ((data_A AND data_B)
    OR (data_A AND NOT data_B AND carry_in)
    OR (NOT data_A AND data_B AND carry_in));

Register Legend:
    FDCPE (Q,D,C,CLR,PRE,CE);
    FDDCPE (Q,D,C,CLR,PRE,CE);
    FTCPE (Q,D,C,CLR,PRE,CE);
    FTDCPE (Q,D,C,CLR,PRE,CE);
    LDCP (Q,D,G,CLR,PRE);
```

Les équations sont écrites de manière différentes que celles déterminées par la vue technologique, mais son en réalité équivalente :

$$S = data_A \oplus (data_B.\overline{carry_{in}} + \overline{data_B}.carry_{in})$$

3ETI – Groupe C
Ce qui équivaut bien à : $S = data_A \oplus data_B \oplus carry_{in}$

Car $X \oplus Y = \overline{X}.Y + X.\overline{Y}$

Et

$$\begin{aligned} R &= data_A.data_B + data_A.\overline{data_B}.carry_{in} + \overline{data_A}.data_B.carry_{in} \\ &= data_A.data_B + carry_{in}\left(data_A.\overline{data_B} + \overline{data_A}.data_B\right) \\ &= data_A.data_B + carry_{in}.(data_A \oplus data_B) \end{aligned}$$

Les équations présentées sont donc juste mais pas écrite sous leur forme la plus simplifiée.

Le tableau résumé des temps de prorogation des signaux est :

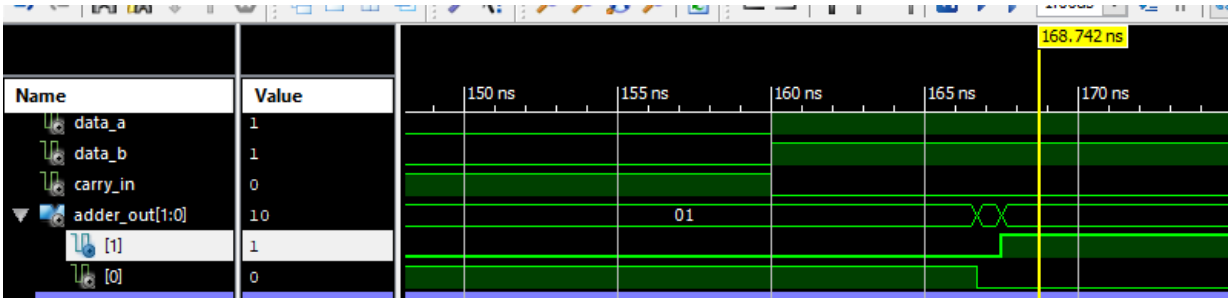
Data Sheet Report

Pad to Pad List

Source Pad	Destination Pad	Delay
carry_in	adder_out<0>	7.500
carry_in	adder_out<1>	7.500
data_A	adder_out<1>	7.500
data_B	adder_out<0>	7.500
data_B	adder_out<1>	7.500
data_A	adder_out<0>	6.700

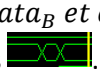
4. Simulation temporelle Post-Fit

Contrairement à la simulation Behaviorial, la simulation Post-Fit tient compte des temps de propagation et de réponses des composants. Pour cette raison, les réponses des sorties aux variations de l'entrée ne sont pas instantanées et ont un temps de réponses visible sur le chronogramme. Ce temps de réponse est, comme indiqué précédemment, de 6,7ns ou 7,5ns selon la variable étudiée :



Ce chronogramme correspond à la transition de ces 2 lignes de la table de vérité :

A	B	Retenue in	Retenue	Somme
0	0	1	0	1
1	1	0	1	0

On remarque que, encore une fois conformément au tableau précédent, que $data_A$ réagit en premier après 6,7ns puis que la variation de $data_B$ et $carry_{in}$ réagit ensuite après 7,5ns. Cela a pour effet de créer cette réponse erronée sur 0,8ns  .