TP4 Pipeline division

François AUDOUIN
Abel DIDOUH

Table des matières

Objectifs

L'objectif principal de ce TP est de mettre en place une architecture pipeline. Nous allons pipeliner la division avec restauration signée.

Le pseudo-code de la division avec restauration signée est le suivant :

Algorithm 1 Division Algorithm

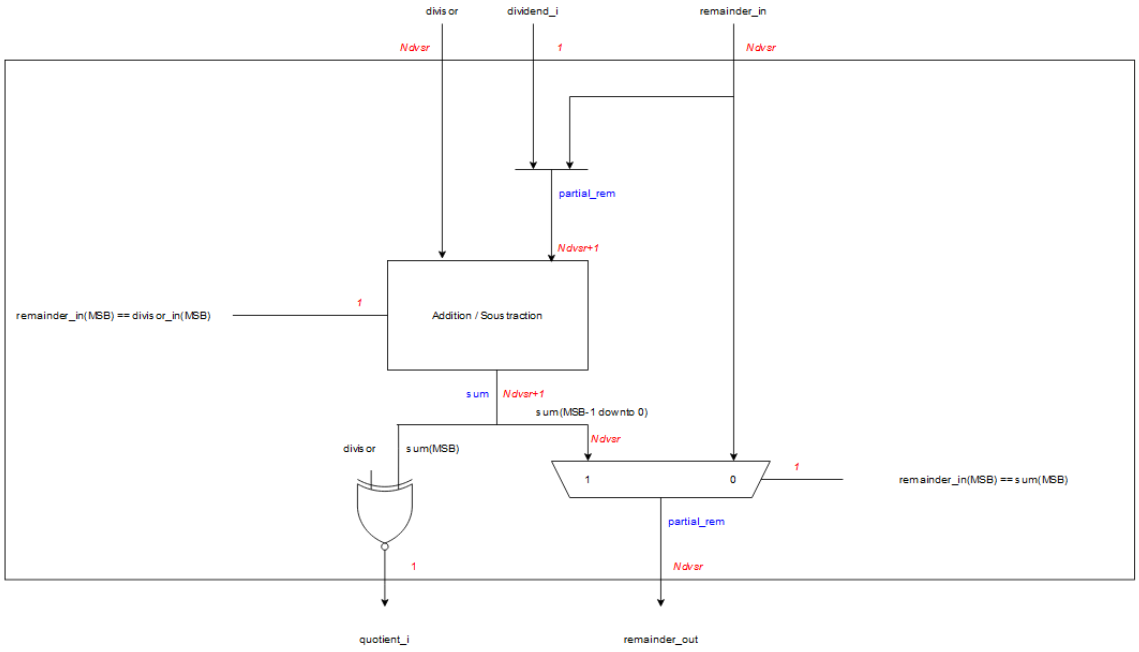
```
1: reste_partiel ← (bit de signe du dividende)
2: for i from N_dividend − 1 to 0 do
3:     reste_partiel ← (reste_partiel << 1)&dividende(i)
4:     if reste_partiel and diviseur are of same sign then
5:         sum ← reste_partiel − diviseur
6:     else
7:         sum ← reste_partiel + diviseur
8:     end if
9:     quotient(i) ← diviseur(MSB) xnor sum(MSB)
10:    if reste_partiel and sum are of same sign (no restoration) then
11:        reste_partiel ← sum
12:    end if
13: end for
14: if reste_partiel and diviseur are of same sign then
15:    sum ← reste_partiel − diviseur
16: else
17:    sum ← reste_partiel + diviseur
18: end if
19: if sum = 0 then
20:    reste_partiel ← sum
21: end if
22: if reste_partiel of opposite sign to diviseur then
23:    quotient ← quotient + 1
24: end if
```

Dans le pseudo-code de la division sans restauration non signée, l'algorithme consiste en 1 seule boucle (qui délivre les résultats quotient et reste). Ici après la boucle, il y a un traitement final pour obtenir le résultat.

Architecture

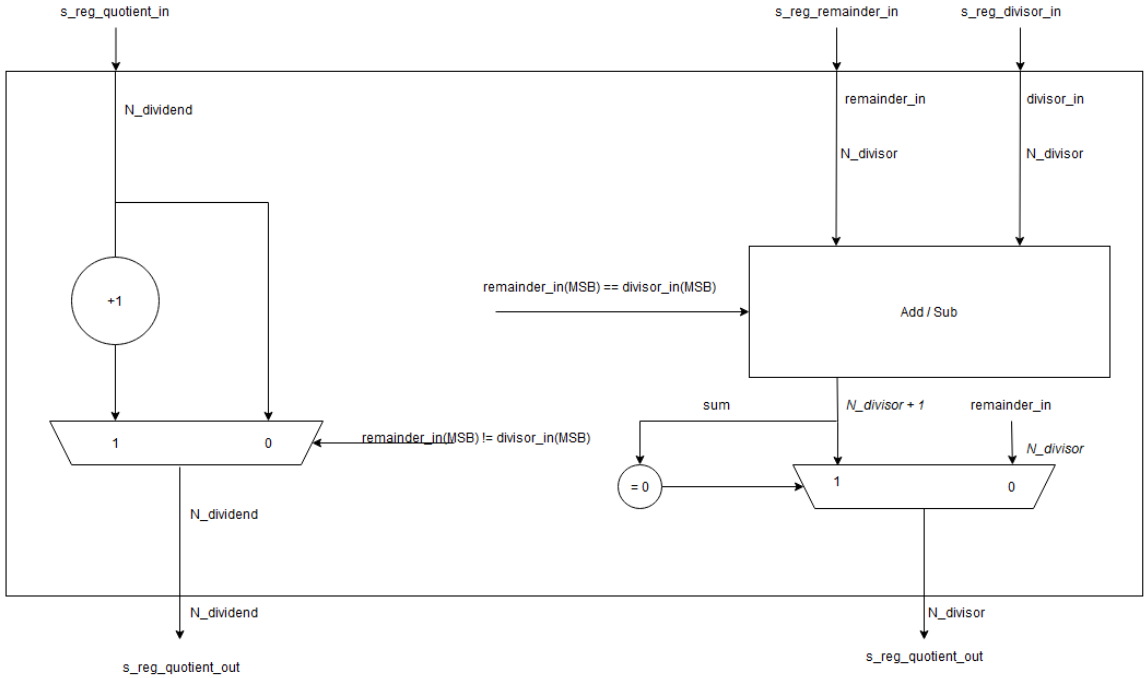
Cellule élémentaire

Nous pouvons faire le schéma RTL (sans registre) de la cellule élémentaire.



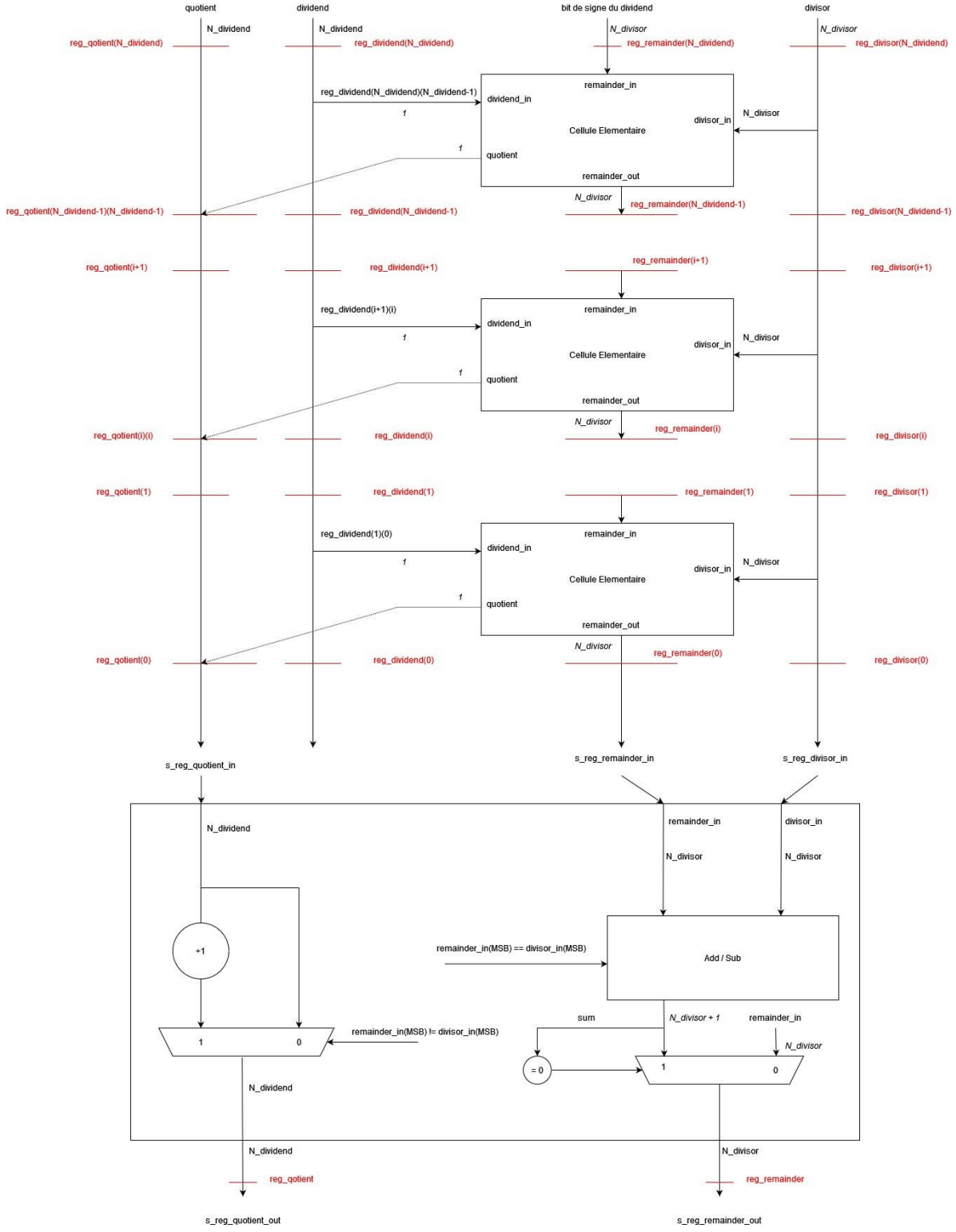
Cellule traitement final

Le schéma RTL du traitement final est le suivant :



Architecture pipeline

Nous pouvons réaliser l'architecture pipeline suivante :

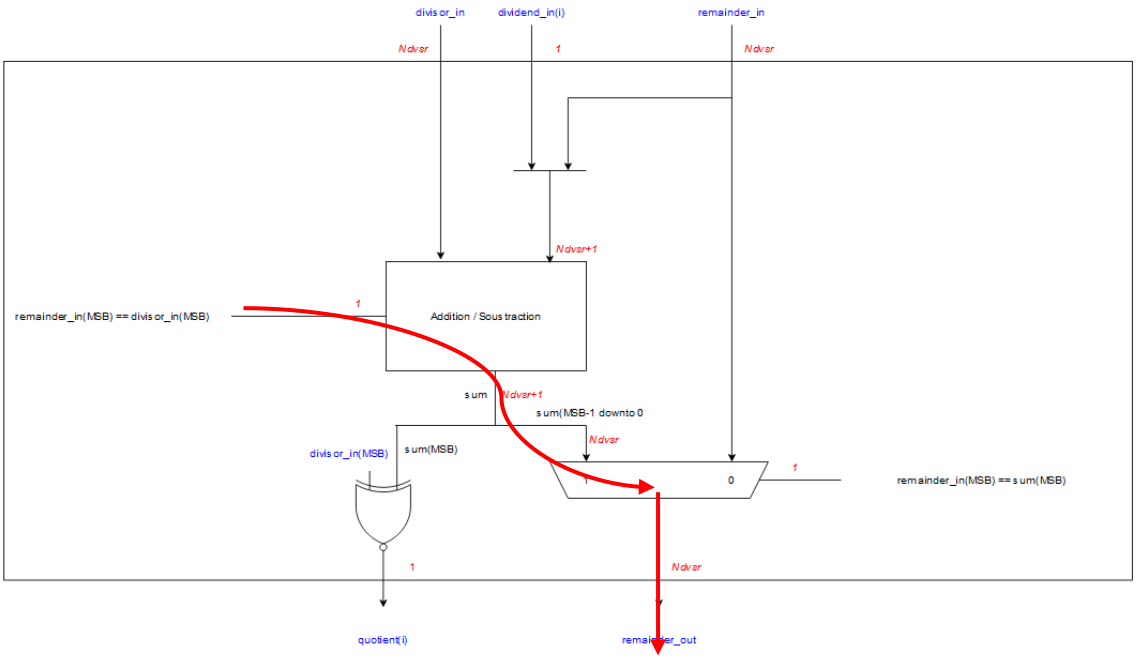


Nous avons représenté la cellule pour l'indice i égale à N_dividend, pour l'indice générale (indice i) et pour l'indice i égale à 0. Nous avons également décidé de ne pas mutualiser les registres quotient et dividend. En effet, dans la suite, il est plus facile de débugger notre architecture RTL lorsque les deux signaux ne sont pas mutualisés. Finalement, nous intégrant le traitement final à la fin de la boucle.

Latence, délai & DFF

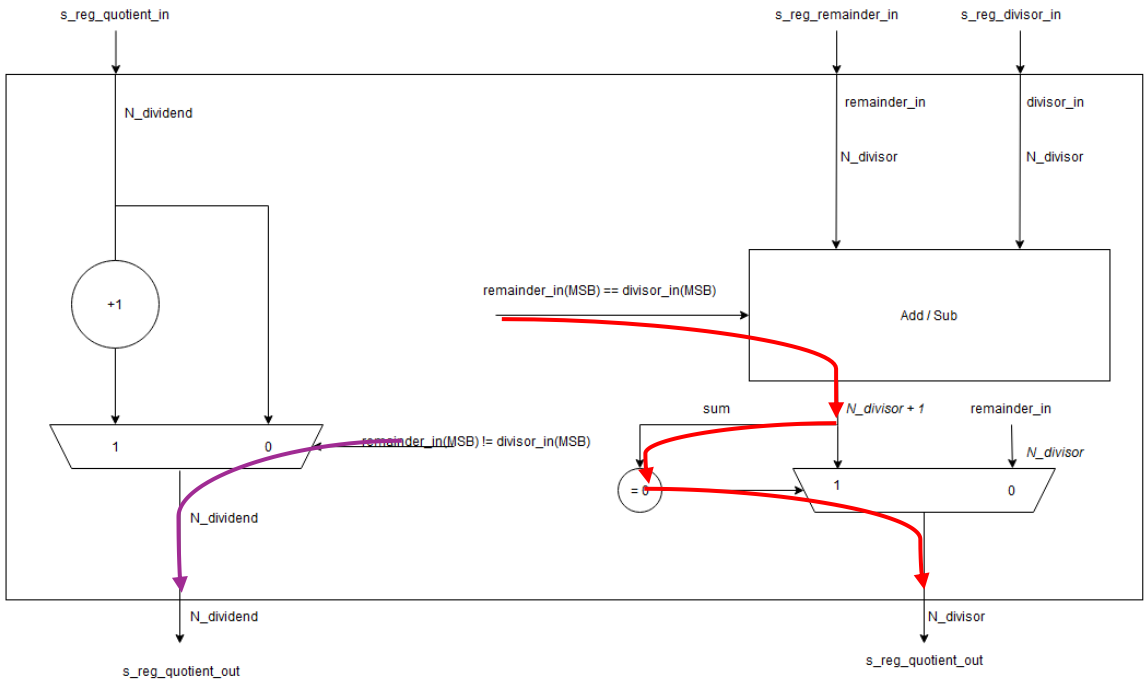
Au début, l'architecture pipeline a besoin de N_dividend + 2 coups d'horloge pour obtenir la première valeur. S'ensuit une donnée cohérente à chaque coup d'horloge.

Nous identifions le chemin critique de la cellule élémentaire :



Le chemin critique de la cellule élémentaire passe par (N_divisor + 1) δFA, d'un δXNOR pour la comparaison et d'un δMUX.

Nous identifions le chemin critique de la cellule finale :



Le chemin critique de la cellule finale passe par (N_divisor + 1) δFA, d'un δXNOR pour la comparaison et d'un δMUX et d'un δ Comparateur.

Il y a un second chemin critique pour la cellule finale. Celui-ci passe par un δXOR puis par N_dividend δFA.

Ainsi, il y a une dépendance en fonction de la taille des opérandes.

Il y a une boucle allant de N_dividend à 0. Il y a N_dividend+1 itérations. A chaque étage nous devons insérer un registre pour chaque signal (dans le cas où nous ne mutualisons pas). Le registre quotient utilise N_dividend DFFs, le registre dividend utilise N_dividend DFFs, le registre remainder utilise N_divisor DFFs tout comme le registre divisor. Ainsi, il y aura :

Nombre de DFF (itérations) = 2*[(N_dividend + 1) * (N_dividend)] + 2*[(N_dividend + 1) * (N_divisor)]

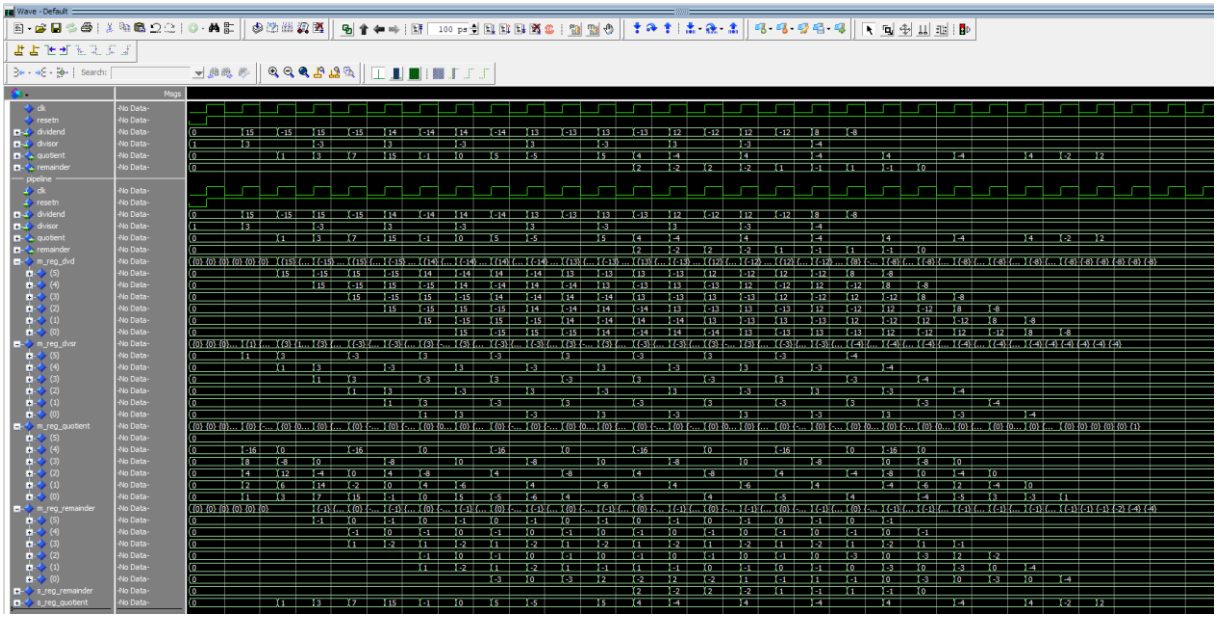
Auquel il faut rajouter un registre de sortie pour le signal quotient sur N_dividend bits et un registre de sortie pour le signal remainder sur N_divisor bits.

Nombre de DFF = 2*[(N_dividend + 1) * (N_dividend)] + 2*[(N_dividend + 1) * (N_divisor)] + N_divisor + N_dividend

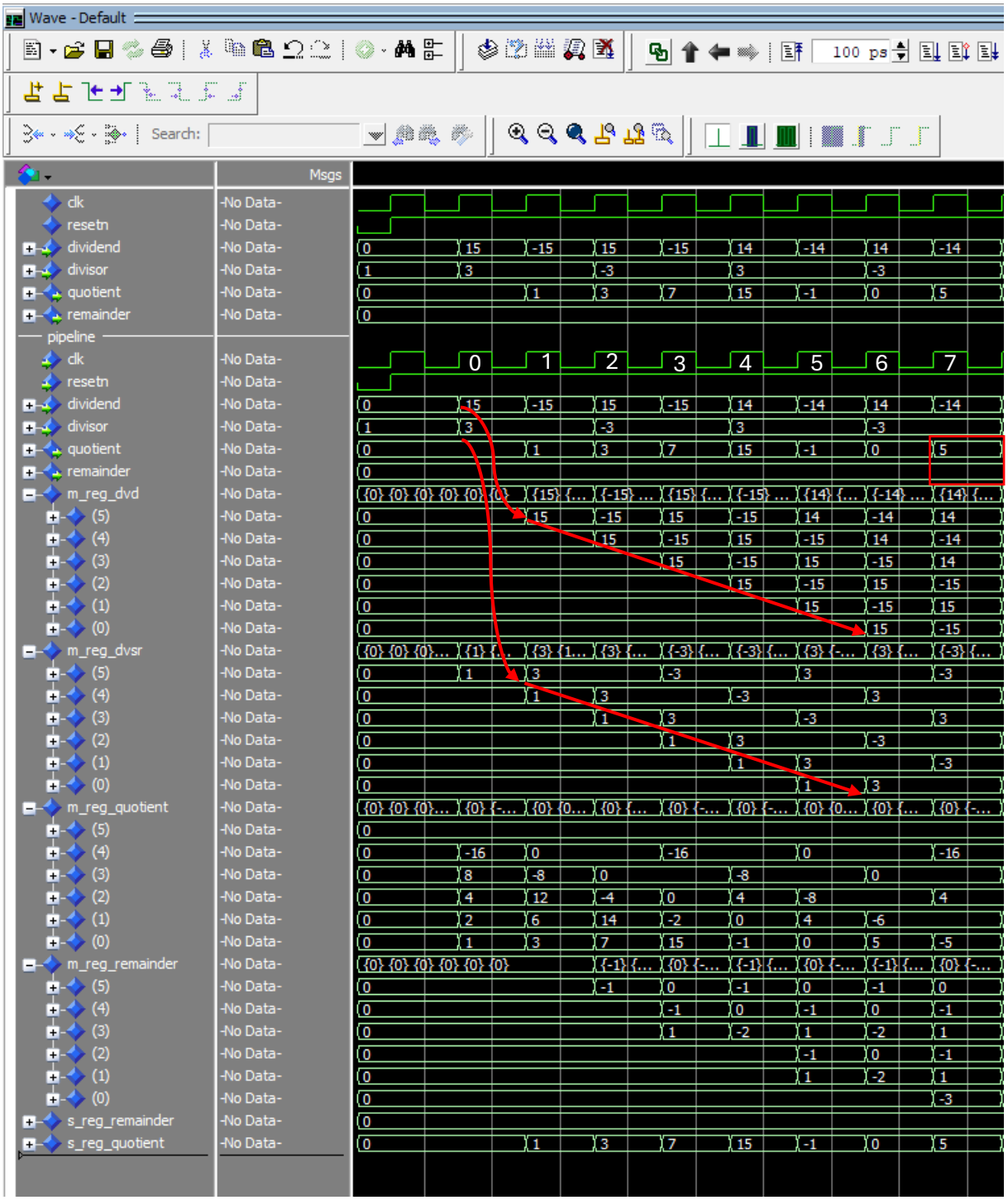
Simulation

Nous allons prouver par simulations que notre architecture RTL est fonctionnelle.

Dans un premier temps, nous avons une vue globale de la simulation Behavioral:

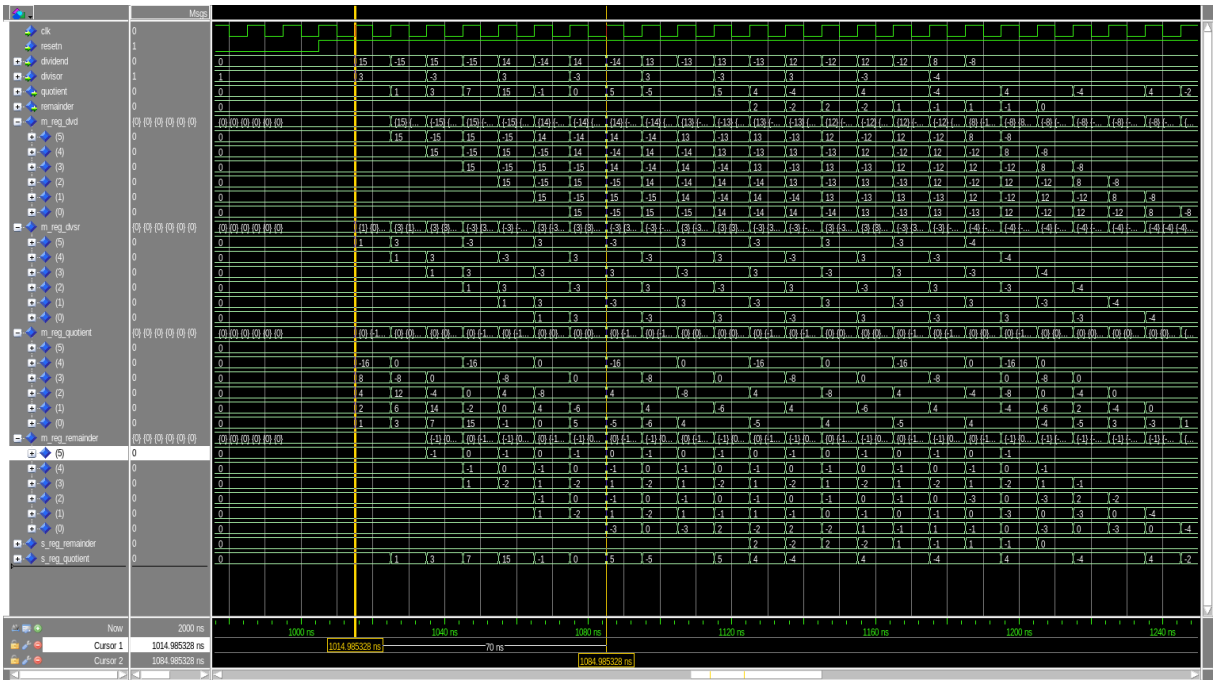


Nous regardons la latence :



Nous modifions le testbench pour rajouter un délai au début. Cela est nécessaire pour la simulation Post-Synthesis Timing Simulation et la simulation Post Implementation car nous devons attendre la fin de l'initialisation du signal GSR.

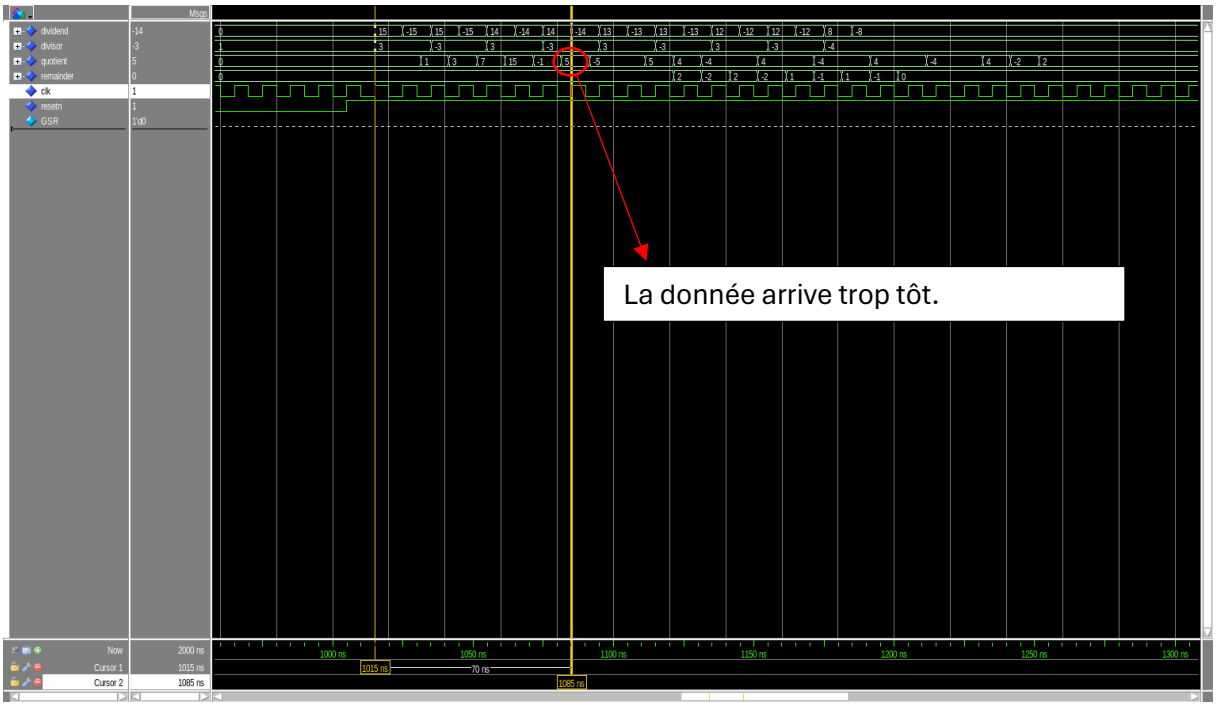
Behavioral Simulation



Il y a une phase d'initialisation correspondant à une latence de 7 périodes. Ensuite l'architecture fournit une donnée cohérente toutes les périodes d'horloges.

La simulation Behavioral est fonctionnelle.

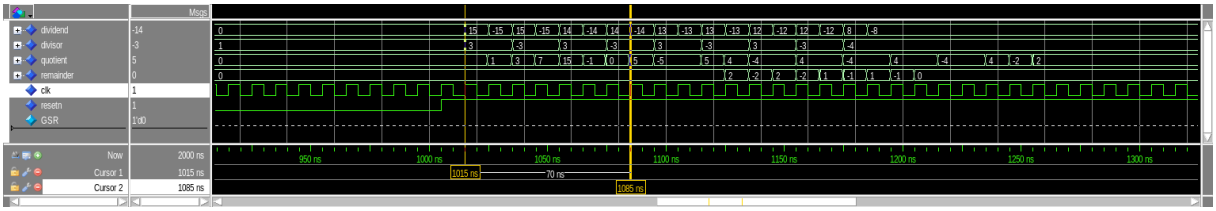
Post-Synthesis Timing Simulation



Nous remarquons que la donnée arrive trop tôt. Pour l'initialisation, nous devrions avoir un résultat cohérent à partir du 7ième coup d'horloge or nous avons ce même résultat un peu avant le 7ième coup d'horloge. Ensuite, nous devrions avoir un résultat cohérent à chaque front d'horloges montants. Cependant dans la simulation Post-Synthesis Timing Simulation chaque donnée cohérente sont en avance.

Nous pouvons dire que la simulation Post-Synthesis Timing Simulation est fonctionnelle. En effet, même si la donnée arrive en avance, pour l'initialisation au bout d'une latence de 7 périodes nous avons une donnée cohérentes correct. Les données (quotient & remainder) qui suivent à chaque coup d'horloges sont également cohérent est correct.

Post-Implementation Timing Simulation



Au bout de 7 coups d'horloges, nous obtenons des résultats cohérents. De même que précédemment, à chaque coup d'horloge après la premier résultats cohérent nous obtenons un résultat cohérent.

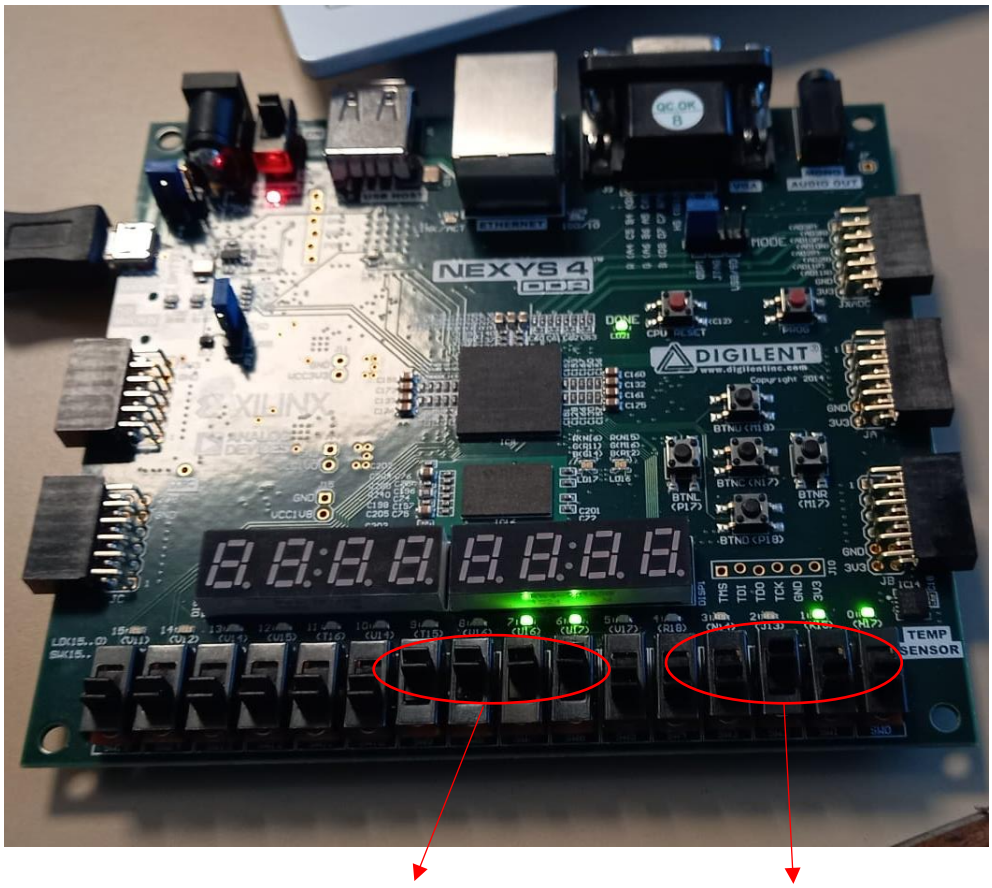
Nous pouvons affirmer que notre architecture RTL est fonctionnelle. Nous pouvons passer à l'implémentation sur carte.

Implémentation

Démonstration

Nous avons effectué une démonstration sur carte.

Un exemple de fonctionnement :



Dividend : 15

Divisor : 4

Nous obtenons pour le remainder : 3

Nous obtenons pour le quotient : 3

Analyse des rapports

Nous analysons les rapports de synthèse et d'implémentation.

Pour N_dividend = 10 et N_divisor = 6 :

Analyse du rapport de synthèse

```
---------------------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
            2 Input   10 Bit       Adders := 1
            3 Input    7 Bit       Adders := 10
            3 Input    6 Bit       Adders := 1
+---XORs :
            2 Input      1 Bit         XORs := 10
+---Registers :
                        10 Bit    Registers := 21
                         6 Bit    Registers := 23
+---Muxes :
            2 Input    7 Bit        Muxes := 10
            2 Input    6 Bit        Muxes := 12
---------------------------------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------------------------------
```

Vivado informe qu'il a reconnu :

- 348 DFFs
- 1 additionneurs sur 10 bits
- 10 additionneurs sur 7 bits
- 1 additionneur sur 6 bits
- 10 XORs
- 22 Muxes

En théorie, nous devrions avoir :

- 368 DFFs
- 33 XORs
- 10 additionneurs sur 7 bits
- 1 additionneur sur 6 bits
- 1 additionneurs sur 10 bits

Nous avons une différence de 20 DFFs. Nous devrions obtenir 33 XORs or nous avons 10 XORs. Vivado fait la différence entre un mux et un XOR de ce fait, le XOR qui est utilisé pour effectuer la comparaison dans le MUX n'est pas pris en compte. Nous trouvons 32 XORs lorsque l'on prend comme principe que Vivado intègre au sein de son muxe un XOR pour effectuer la comparaison.

Report Cell Usage:

	Cell	Count
1	BUFG	1
2	CARRY4	22
3	LUT2	57
4	LUT3	8
5	LUT4	109
6	LUT5	4
7	LUT6	6
8	SRL16E	14
9	FDCE	177
10	FDRE	14
11	IBUF	18
12	OBUF	16

Il n'y a pas de latch.

Relevé des ressources

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs	171	0	0	63400	0.27
LUT as Logic	157	0	0	63400	0.25
LUT as Memory	14	0	0	19000	0.07
LUT as Distributed RAM	0	0			
LUT as Shift Register	14	0			
Slice Registers	191	0	0	126800	0.15
Register as Flip Flop	191	0	0	126800	0.15
Register as Latch	0	0	0	126800	0.00
F7 Muxes	0	0	0	31700	0.00
F8 Muxes	0	0	0	15850	0.00

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
177	Yes	-	Reset
0	Yes	Set	-
14	Yes	Reset	-

2. Slice Logic Distribution

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice	69	0	0	15850	0.44
SLICEL	48	0			
SLICEM	21	0			
LUT as Logic	157	0	0	63400	0.25
using O5 output only	0				
using O6 output only	130				
using O5 and O6	27				
LUT as Memory	14	0	0	19000	0.07
LUT as Distributed RAM	0	0			
LUT as Shift Register	14	0			
using O5 output only	14				
using O6 output only	0				
using O5 and O6	0				
Slice Registers	191	0	0	126800	0.15
Register driven from within the Slice	112				
Register driven from outside the Slice	79				
LUT in front of the register is unused	25				
LUT in front of the register is used	54				
Unique Control Sets	2		0	15850	0.01

* * Note: Available Control Sets calculated as Slice * 1, Review the Control Sets Report for more information regarding control sets.

Relevé des ressources :

- 69 slices
- 171 LUTS
- 191 DFF

3. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	135	0.00
RAMB36/FIFO*	0	0	0	135	0.00
RAMB18	0	0	0	270	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

4. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	240	0.00

5. IO and GT Specific

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	34	0	0	210	16.19
IOB Master Pads	16				
IOB Slave Pads	17				
Bonded IPADs	0	0	0	2	0.00
PHY_CONTROL	0	0	0	6	0.00
PHASER_REF	0	0	0	6	0.00
OUT_FIFO	0	0	0	24	0.00
IN_FIFO	0	0	0	24	0.00
IDELAYCTRL	0	0	0	6	0.00
IBUFDS	0	0	0	202	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	24	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	24	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	300	0.00
ILOGIC	0	0	0	210	0.00
OLOGIC	0	0	0	210	0.00

6. Clocking

Site Type	Used	Fixed	Prohibited	Available	Util%
BUFGCTRL	1	0	0	32	3.13
BUFIO	0	0	0	24	0.00
MMCME2_ADV	0	0	0	6	0.00
PLLE2_ADV	0	0	0	6	0.00
BUFMRCE	0	0	0	12	0.00
BUFHCE	0	0	0	96	0.00
BUFR	0	0	0	24	0.00

Relevé des ressources :

- 0 block RAM
- 0 DSP
- 1 arbre d'horloge (BUFGCTRL)
- 0 PLL

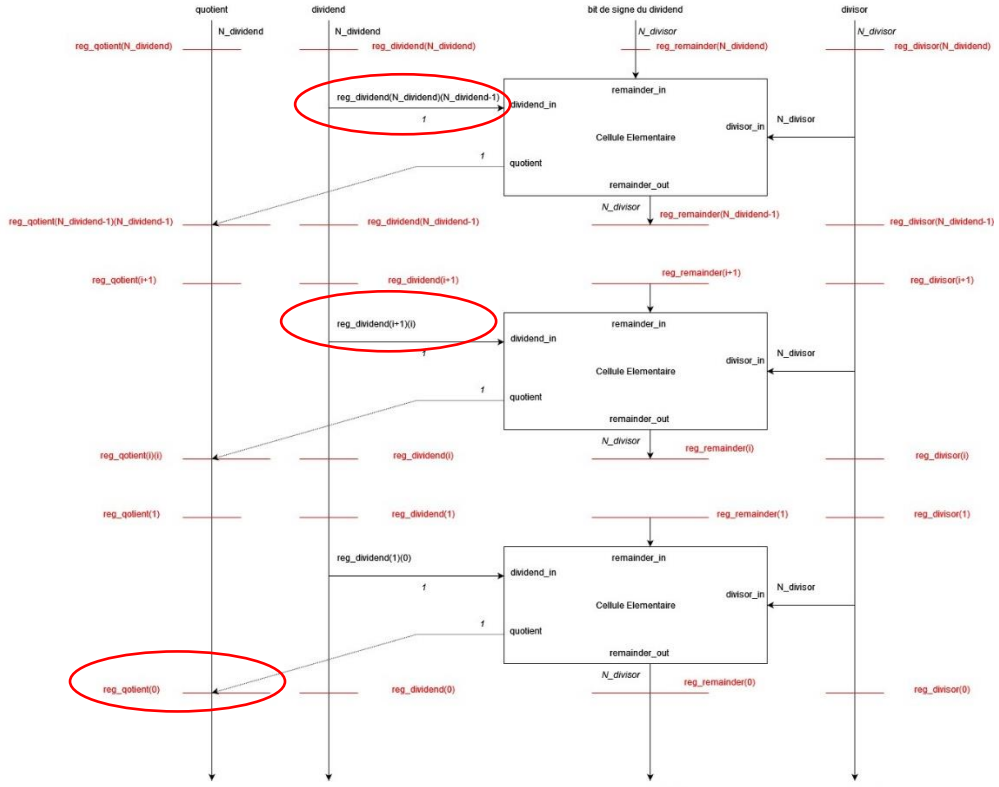
8. Primitives

Ref Name	Used	Functional Category
FDCE	177	Flop & Latch
LUT4	109	LUT
LUT2	57	LUT
CARRY4	22	CarryLogic
IBUF	18	IO
OBUF	16	IO
SRL16E	14	Distributed Memory
FDRE	14	Flop & Latch
LUT3	8	LUT
LUT6	6	LUT
LUT5	4	LUT
BUFG	1	Clock

L’architecture utilise 14 primitives SRL16E. Cette primitive est un Shift Register Look-up table.

```
Static Shift Register Report:
+-----------+---------------------------+--------+-------+--------------+-------------------+-------------------+--------+---------+
|Module Name | RTL Name                  | Length | Width | Reset Signal | Pull out first Reg | Pull out last Reg | SRL16E | SRLC32E |
+-----------+---------------------------+--------+-------+--------------+-------------------+-------------------+--------+---------+
|demo        | div/m_reg_dvd_reg[7][6]   | 4      | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_dvd_reg[6][5]   | 5      | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_dvd_reg[5][4]   | 6      | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_dvd_reg[4][3]   | 7      | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_dvd_reg[3][2]   | 8      | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_dvd_reg[2][1]   | 9      | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_dvd_reg[1][0]   | 10     | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_quotient_reg[0][9] | 10  | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_quotient_reg[0][8] | 9   | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_quotient_reg[0][7] | 8   | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_quotient_reg[0][6] | 7   | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_quotient_reg[0][5] | 6   | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_quotient_reg[0][4] | 5   | 1     | YES          | NO                | YES               | 1      | 0       |
|demo        | div/m_reg_quotient_reg[0][3] | 4   | 1     | YES          | NO                | YES               | 1      | 0       |
+-----------+---------------------------+--------+-------+--------------+-------------------+-------------------+--------+---------+
```

Nous retrouvons bien le décalage de chaque bit quotient et dividende.



Nous pouvons supposer que la primitive SRL16E est utilisé pour remplir le registre quotient. La primitive SRL16E est utilisé pour acheminer le bit provenant du registre reg_dividend(i+1)(i) dans le calculus.

Analyse du rapport de timings

```
------------------------------------------------------------------------------
| Design Timing Summary
| ---------------------
------------------------------------------------------------------------------
WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints
-------      -------  ---------------------  -------------------
  4.831        0.000                      0                  188

WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints
-------      -------  ---------------------  -------------------
  0.166        0.000                      0                  188

WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
--------     --------  ----------------------  --------------------
   4.020        0.000                       0                   206


------------------------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------------------------

Clock        Waveform(ns)       Period(ns)      Frequency(MHz)
-----        ------------       ----------      --------------
sys_clk_pin  {0.000 5.000}      10.000          100.000


------------------------------------------------------------------------------
| Intra Clock Table
| -----------------
------------------------------------------------------------------------------
Clock            WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints
-----            -------      -------  ---------------------  -------------------
sys_clk_pin        4.831        0.000                      0                  188

WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints
-------      -------  ---------------------  -------------------
  0.166        0.000                      0                  188

WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
--------     --------  ----------------------  --------------------
   4.020        0.000                       0                   206
```

Relevé :
Slack = WNS = 4.831 ns
WNS ≥ 0 et WHS ≥ 0
Calcul fréquence maximale :

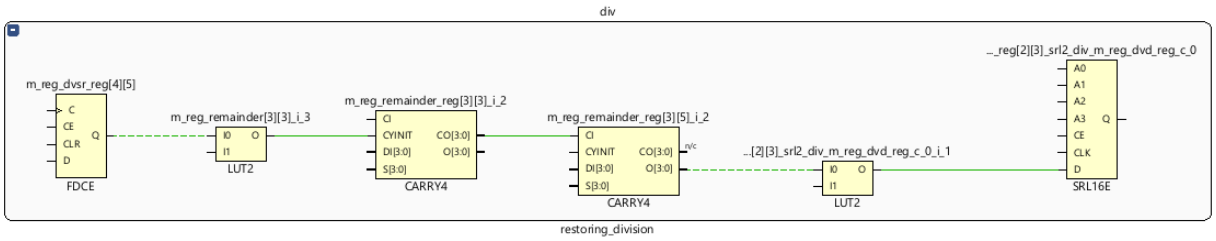
$$fmax = \frac{1}{(period - WNS)} = \frac{1}{(10-4.831)} = 193{,}4\ MHz$$

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             4.831ns  (required time - arrival time)
  Source:                 div/m_reg_dvsr_reg[4][5]/C
                            (rising edge-triggered cell FDCE clocked by sys_clk_pin
{rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            div/m_reg_quotient_reg[2][3]_srl2_div_m_reg_dvd_reg_c_0/D
                            (rising edge-triggered cell SRL16E clocked by sys_clk_pin
{rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:             sys_clk_pin
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            10.000ns  (sys_clk_pin rise@10.000ns - sys_clk_pin rise@0.000ns)
  Data Path Delay:        4.866ns  (logic 1.865ns (38.324%)  route 3.001ns (61.676%))
  Logic Levels:           4  (CARRY4=2 LUT2=2)
  Clock Path Skew:        -0.035ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    5.013ns = ( 15.013 - 10.000 )
    Source Clock Delay      (SCD):    5.307ns
    Clock Pessimism Removal (CPR):    0.259ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
                         (clock sys_clk_pin rise edge)
                                                      0.000     0.000 r
    E3                                                0.000     0.000 r  clk (IN)
                         net (fo=0)                   0.000     0.000    clk
    E3                   IBUF (Prop_ibuf_I_O)         1.482     1.482 r  clk_IBUF_inst/O
                         net (fo=1, routed)           2.025     3.506    clk_IBUF
    BUFGCTRL_X0Y16       BUFG (Prop_bufg_I_O)         0.096     3.602 r  clk_IBUF_BUFG_inst/O
                         net (fo=205, routed)         1.704     5.307    div/clk_IBUF_BUFG
    SLICE_X5Y75          FDCE                                        r
div/m_reg_dvsr_reg[4][5]/C
  -------------------------------------------------------------------    -------------------
    SLICE_X5Y75          FDCE (Prop_fdce_C_Q)         0.419     5.726 r  div/m_reg_dvsr_reg[4][5]/Q
                         net (fo=8, routed)           1.201     6.927    div/p_0_in11_in
    SLICE_X3Y74          LUT2 (Prop_lut2_I0_O)        0.299     7.226 r  div/m_reg_remainder[3][3]_i_3/O
                         net (fo=1, routed)           0.548     7.774    div/v_sum113_out
    SLICE_X3Y75          CARRY4 (Prop_carry4_CYINIT_CO[3])
                                                      0.580     8.354 r  div/m_reg_remainder_reg[3][3]_i_2/CO[3]
                         net (fo=1, routed)           0.000     8.354    div/m_reg_remainder_reg[3][3]_i_2_n_0
    SLICE_X3Y76          CARRY4 (Prop_carry4_CI_O[2])
                                                      0.239     8.593 r  div/m_reg_remainder_reg[3][5]_i_2/O[2]
                         net (fo=7, routed)           0.623     9.215    div/p_0_in14_in
    SLICE_X4Y76          LUT2 (Prop_lut2_I0_O)        0.328     9.543 r  div/m_reg_quotient_reg[2][3]_srl2_div_m_reg_dvd_reg_c_0_i_1/O
                         net (fo=1, routed)           0.630    10.173    div/m_reg_quotient_reg[2][3]_srl2_div_m_reg_dvd_reg_c_0_i_1_n_0
    SLICE_X2Y76          SRL16E                                  r  div/m_reg_quotient_reg[2][3]_srl2_div_m_reg_dvd_reg_c_0/D
  -------------------------------------------------------------------    -------------------

                         (clock sys_clk_pin rise edge)
                                                     10.000    10.000 r
    E3                                                0.000    10.000 r  clk (IN)
                         net (fo=0)                   0.000    10.000    clk
    E3                   IBUF (Prop_ibuf_I_O)         1.411    11.411 r  clk_IBUF_inst/O
                         net (fo=1, routed)           1.920    13.331    clk_IBUF
    BUFGCTRL_X0Y16       BUFG (Prop_bufg_I_O)         0.091    13.422 r  clk_IBUF_BUFG_inst/O
                         net (fo=205, routed)         1.590    15.013    div/clk_IBUF_BUFG
    SLICE_X2Y76          SRL16E                                      r
div/m_reg_quotient_reg[2][3]_srl2_div_m_reg_dvd_reg_c_0/CLK
                         clock pessimism              0.259    15.272
                         clock uncertainty           -0.035    15.236
    SLICE_X2Y76          SRL16E (Setup_srl16e_CLK_D)
                                                     -0.232    15.004
div/m_reg_quotient_reg[2][3]_srl2_div_m_reg_dvd_reg_c_0
  -------------------------------------------------------------------
                         required time                         15.004
                         arrival time                         -10.173
  -------------------------------------------------------------------
                         slack                                 4.831
```

Le chemin critique se trouve dans le niveau hiérarchique div entre le registre divisor et le registre quotient (srl2 à décalage).



Il traverse 2 LUT2 et 2 CARRY4.

Dépendance

Afin de vérifier la dépendance en fonction du nombre de bits des différents opérandes, nous comparons nos estimations théoriques et nos résultats relevés après implémentations (LUT, DFF, fréquence, ...) pour 2 paramètres génériques différents.

Pour N_dividend = 10	Théorique	Pratique
LUT	-	69
DFF	368	348
Fréquence	-	193 MHz
XOR	33	32
Additionneurs (10 bits)	1	1
Additionneurs (7 bits)	10	10
Additionneurs (6 bits)	1	1
Muxes	12	22

Pour N_dividend = 12	Théorique	Pratique
LUT	-	115
DFF	432	408
Fréquence	-	262.8 MHz
XOR	27	38
Additionneurs (12 bits)	1	1
Additionneurs (5 bits)	12	12
Additionneurs (4 bits)	1	1
Muxes	14	26

Tout d'abord, lorsque le nombre de bit dividend augmente le nombre de DFF augmente. Nous remarquons que nous n'avons pas exactement le bon nombre de DFF entre la partie théorique et la partie pratique.

Le synthétiseur vient optimiser l'architecture. Pour cela, il va supprimer les registres non utilisés.

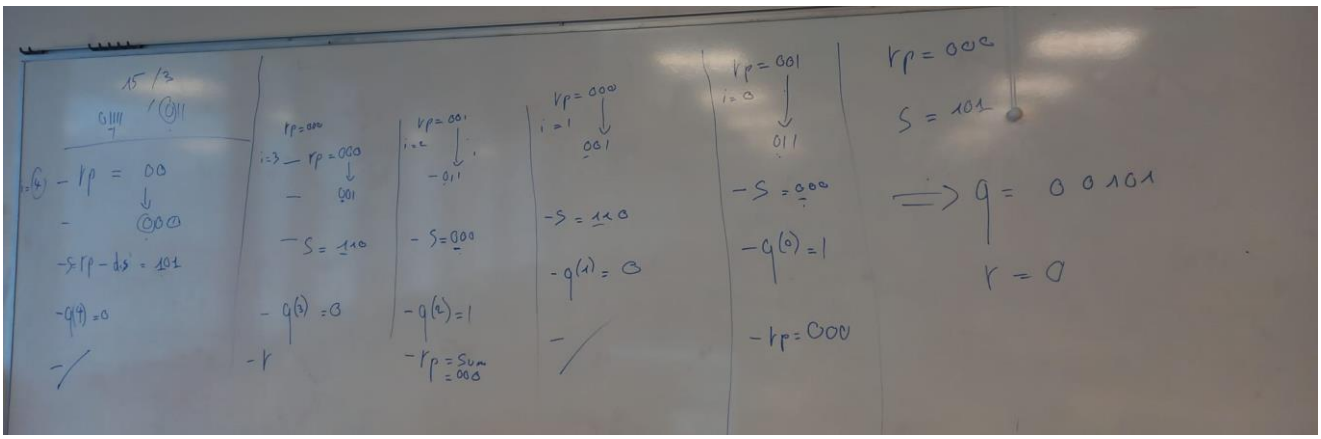
```
INFO: [Synth 8-3886] merging instance 'div/m_reg_dvd_reg[10][9]' (FDC) to 'div/m_reg_remainder_reg[10][0]'
INFO: [Synth 8-3886] merging instance 'div/m_reg_remainder_reg[10][0]' (FDC) to 'div/m_reg_remainder_reg[10][1]'
INFO: [Synth 8-3886] merging instance 'div/m_reg_remainder_reg[10][1]' (FDC) to 'div/m_reg_remainder_reg[10][2]'
INFO: [Synth 8-3886] merging instance 'div/m_reg_remainder_reg[10][2]' (FDC) to 'div/m_reg_remainder_reg[10][3]'
INFO: [Synth 8-3886] merging instance 'div/m_reg_remainder_reg[10][3]' (FDC) to 'div/m_reg_remainder_reg[10][4]'
INFO: [Synth 8-3886] merging instance 'div/m_reg_remainder_reg[10][4]' (FDC) to 'div/m_reg_remainder_reg[10][5]'
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][9]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][8]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][7]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][6]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][5]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][4]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][3]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][2]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][1]) is unused and will be removed from module demo.
WARNING: [Synth 8-3332] Sequential element (div/m_reg_quotient_reg[10][0]) is unused and will be removed from module demo.
```

Nous remarquons également que la fréquence de fonctionnement augmente lorsque le nombre dividend en bits augmente.

Annexe :

Calcul et raisonnement sur le tableau blanc de la 3401

Etude de l'algorithme à la main : 15 / 3



Vérification de notre code avec un exemple 15 / 3