ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΗΜΜΥ

Ψηφιακά Συστήματα VLSI

Ακαδημαϊκό έτος 2020-21

3η Εργαστηριακή Άσκηση

Υλοποίηση FIR Φίλτρου

Αιμιλία Σιόκουρου – 03117703

Νάταλυ Πεγειώτη – 03117707

**Όλοι οι κώδικες της VHDL παρατίθενται σε ξεχωριστά αρχεία*

Στη γενική περίπτωση, η σχέση εισόδου - εξόδου ενός FIR φίλτρου είναι η ακόλουθη:

$$y[n] = \sum_{k=0}^{M} h[k]\,x[n-k] = h[0]\,x[n] \;+\; h[1]\,x[n-1] \;+\; \ldots \;+\; h[M]\,x[n-M]$$

όπου

- *M* είναι η **τάξη του φίλτρου**
- *y*[*n*] είναι η **έξοδος του φίλτρου** τη διακριτή χρονική στιγμή *n*
- *h*[*k*] είναι ο ***k*-οστός συντελεστής του φίλτρου**, με *k* = 0, 1, 2, ..., *M*
- *x*[*n*] είναι η **τιμή του σήματος εισόδου** τη διακριτή χρονική στιγμή *n*

Το μήκος L της εξόδου y, έχει υπολογιστεί για όλες τις περιπτώσεις στα 19 bits, ως εξής:

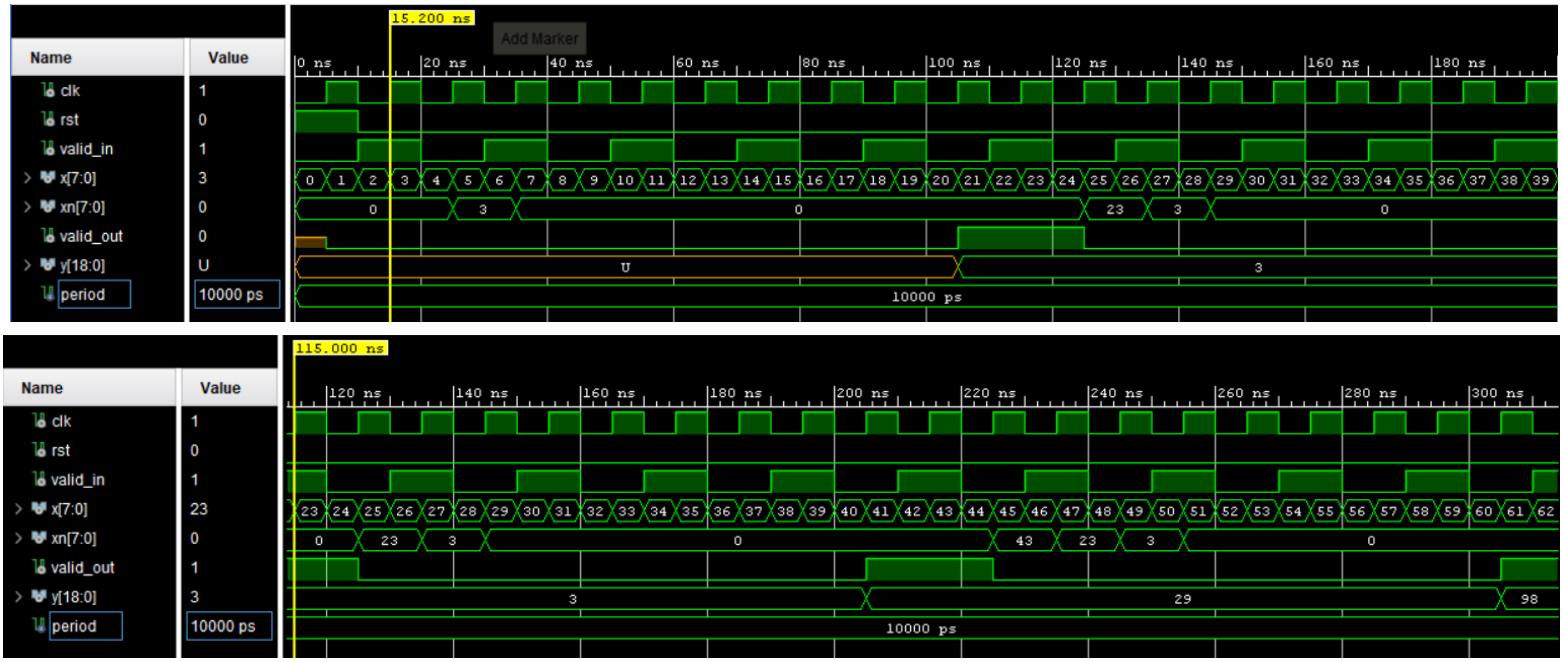
Στη χειρότερη περίπτωση πολλαπλασιάζονται οι 8-bit $(11111111)_2 = (FF)_{16}$
Επομένως προκύπτει ένας 16-bit αριθμός $FF*FF = (FE01)_{16} = (1111111000000001)_2$

Στην χειρότερη περίπτωση γίνεται η πρόσθεση του πιο πάνω αριθμού 8 φορές.
Επομένως η τελική έξοδος είναι $8*FE01 = (7F008)_{16} = (11111110000000001000)_2$, δηλαδή 19 bits.

Ζητούμενο 1: Για την πρώτη αρχιτεκτονική θα πρέπει σχεδιάσετε ένα σύστημα που να υλοποιεί το δομικό διάγραμμα του Σχήματος 1. Η συγκεκριμένη αρχιτεκτονική χρησιμοποιεί μια Multiply-Accumulate (MAC) μονάδα για τη υλοποίηση των πολλαπλασιασμών και των αθροίσεων. Η υλοποίηση να γίνει για N=8 bits εύρος δεδομένων x και M=8 συντελεστές.

Οι **Sources κώδικες** του φίλτρου **FIR**, των μονάδων **MAC, ROM, RAM** και **Control Unit**, καθώς και το αντίστοιχο **Testbench** παρατίθενται σε αρχεία VHDL στον φάκελο "1".

Η αντίστοιχη προσομοίωση (**simulation**) φαίνεται πιο κάτω:



Παραθέτουμε δύο στιγμιότυπα από την προσομοίωση για την καλύτερη επεξήγησή της.

Για την απόδειξη της ορθότητας της υλοποίησης μπορούμε να δούμε μερικά από τα αποτελέσματα της προσομοίωσης:

1) Στο 1ο στιγμιότυπο παρατηρούμε ότι η πρώτη valid τιμή που δέχεται το φίλτρο είναι το 3, τιμή η οποία προκύπτει στην έξοδό του μετά από 9 κύκλους και είναι ορθή καθώς y[0] = h[0]*x[0] = 1*3 = 3.
   Παρατηρούμε ότι το valid_out γίνεται 1 για την έξοδο του αποτελέσματος.
2) Σε 8 κύκλους μετά από την πρώτη είσοδο, παρατηρούμε ότι η τιμή που δέχεται το φίλτρο δεν είναι valid, επομένως το φίλτρο δέχεται την αμέσως επόμενη valid τιμή που είναι το 23.
   Η είσοδος αυτή φαίνεται και στο 2ο στιγμιότυπο.
   Έπειτα από 9 κύκλους το valid_out γίνεται 1 και προκύπτει η ορθή έξοδος του φίλτρου y[1] = h[0]*x[1] + h[1]*x[0] = 1*23 + 2*3 = 23 + 6 = 29.
3) Ομοίως, έπειτα από 8 κύκλους, η αμέσως επόμενη valid_in τιμή είναι το 43 και εισάγεται στο φίλτρο.
   Έπειτα από 9 κύκλους το valid_out γίνεται 1 και προκύπτει η ορθή έξοδος του φίλτρου y[1] = h[0]*x[2] + h[1]*x[1] + h[2]*x[0] = 1*43 + 2*23 + 3*3 = 43 + 46 + 9 = 98.

Με παρόμοιο τρόπο προκύπτουν και τα υπόλοιπα αποτελέσματα.
Στην προσομοίωση παρατηρούμε την προσθήκη μιας επιπλέον μεταβλητής – πίνακα xn[7:0], 8 θέσεων, όπου παρατηρούμε τις valid εισόδους του φίλτρου. Η προσθήκη αυτή είναι καθαρά για την ευκολία των υπολογισμών μας καθώς παρατηρούμε την προσομοίωση.

Critical Path

Κρίσιμο είναι το μονοπάτι με την μεγαλύτερη συνολικη καθυστέρηση, δηλαδή max{Total Delay}, όπου στην περίπτωση αυτή είναι το Path 1.

Reports | Design Runs | Timing

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	∞	9	9	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[17]/D	5.550	2.483	3.067	∞
Path 2	∞	9	9	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[18]/D	5.458	2.391	3.067	∞
Path 3	∞	9	9	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[16]/D	5.437	2.370	3.067	∞
Path 4	∞	8	8	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[13]/D	5.436	2.369	3.067	∞
Path 5	∞	8	8	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[15]/D	5.417	2.350	3.067	∞
Path 6	∞	8	8	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[14]/D	5.344	2.277	3.067	∞
Path 7	∞	8	8	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[12]/D	5.323	2.256	3.067	∞
Path 8	∞	7	7	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[9]/D	5.312	2.545	2.767	∞
Path 9	∞	7	7	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[11]/D	5.293	2.526	2.767	∞
Path 10	∞	7	7	23	rom_unit/rom_out_reg[1]/C	mac_unit/output_reg[10]/D	5.220	2.453	2.767	∞

Κατανάλωση πόρων του FPGA

Δεξιά παρατηρούμε τους πόρους του FPGA που χρησιμοποιούνται.

Utilization — Post-Synthesis | Post-Implementation

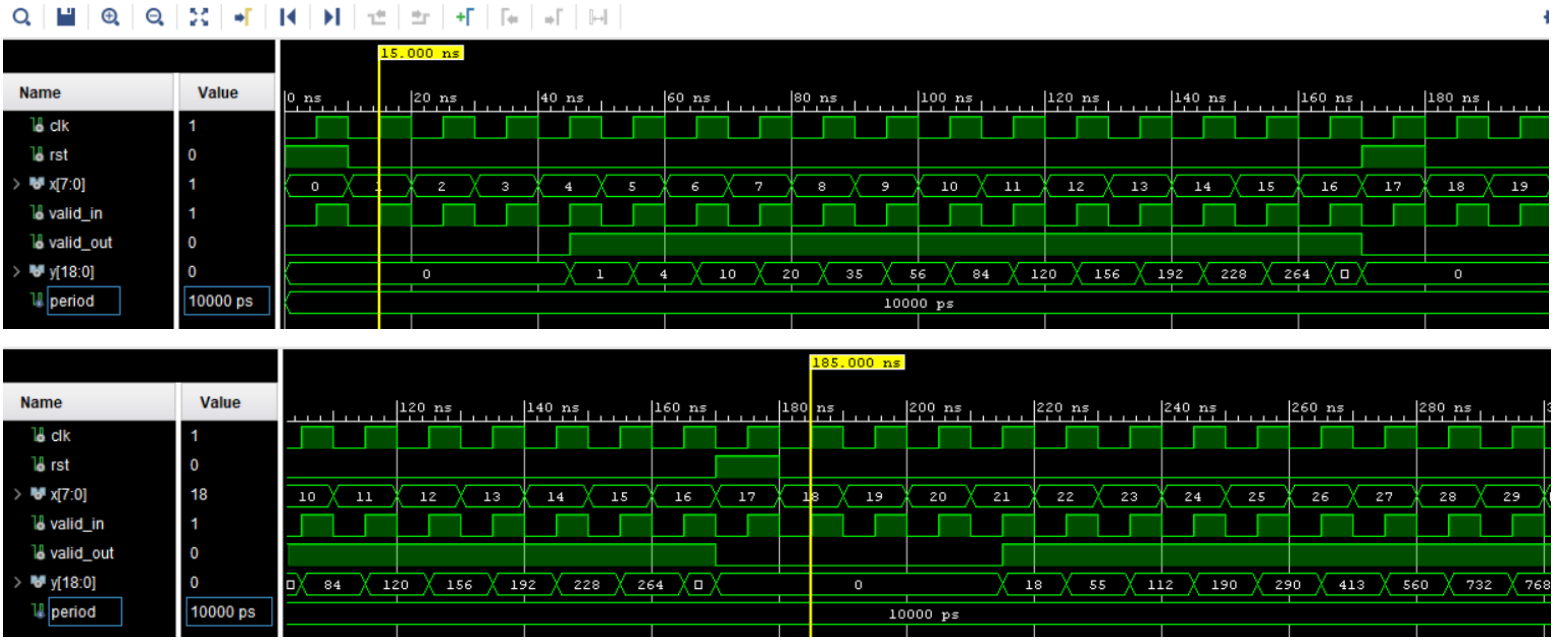
Graph | Table

Resource	Utilization	Available	Utilization %
LUT	84	17600	0.48
FF	147	35200	0.42
IO	48	100	48.00
BUFG	1	32	3.13

Ζητούμενο 2: Στην δεύτερη αρχιτεκτονική ζητείται (α) να "ξεδιπλώσετε" το φίλτρο που υλοποιείσατε προηγουμένως και (β) να το μετατρέψετε σε πλήρως pipeline. Στο Σχήμα 2 δίνεται μια προτεινόμενη αρχιτεκτονική για ένα 4-tap FIR φίλτρο στην οποία μπορεί να βασιστεί η σχεδίαση σας. Η συγκεκριμένη αρχιτεκτονική χρησιμοποιεί πολλαπλές μονάδες πολλαπλασιαστών και αθροιστών. Η υλοποίηση να γίνει για N=8 bits εύρος δεδομένων x και M=8 συντελεστές.

Ο **source κώδικας** του ερωτήματος και το αντίστοιχο **testbench** βρίσκονται σε αρχεία VHDL στον φάκελο "2".

Η προσομοίωση (**simulation**) του κυκλώματος φαίνεται πιο κάτω:



Παρατηρούμε ότι σε αυτή την περίπτωση το φίλτρο δέχεται νέα είσοδο x σε κάθε κύκλο ρολογιού εφόσον το valid_in ισούται με 1. Έπειτα από κάποια αρχική καθυστέρηση $T_{latency}$ προκύπτει έξοδος σε κάθε κύκλο ρολογιού.

Για την απόδειξη της ορθότητας της υλοποίησης μπορούμε να δούμε μερικά από τα αποτελέσματα της προσομοίωσης:

1) Στον 1ο κύκλο ρολογιού πραγματοποιείται το reset.
2) Στον 2ο κύκλο και όταν το valid_in γίνει 1, το φίλτρο δέχεται την πρώτη είσοδο, x = 1.
   Έπειτα από 3 κύκλους (αρχικό $T_{latency}$) παρατηρούμε να προκύπτει η πρώτη έξοδος y[0] και το valid_out να γίνεται 1.
   Η έξοδος είναι ορθή αφού y[0] = h[0]*x[0] = 1*1 = 1.
3) Στον 3ο κύκλο και όταν το valid_in γίνει 1, το φίλτρο δέχεται την δεύτερη είσοδο, x = 2.
   Παρατηρούμε ότι η έξοδος (λόγω του pipeline) δεν έχει πλέον καθυστέρηση, αλλά προκύπτει στον επόμενο κύκλο από την 1η έξοδο.
   Η τιμή της εξόδου είναι ορθή αφού y[1] = h[0]*x[1] + h[1]*x[0] = 1*2 + 2*1 = 2+2 = 4.
4) Στον 4ο κύκλο το φίλτρο δέχεται είσοδο x = 3.
   Η τιμή της εξόδου είναι ορθή αφού y[2] = h[0]*x[2] + h[1]*x[1] + h[2]*x[0] = 1*3 + 2*3 + 3*1 = 3 + 4 + 3 = 10.

Παρατηρούμε ότι σε κάθε επόμενο κύκλο ρολογιού προκύπτει η ορθή έξοδος.
Όταν το reset γίνει 1, όλα μηδενίζονται και το φίλτρο ξεκινά από την αρχή με νέα είσοδο.
Μετά από κάποιο αρχικό $T_{latency}$ προκύπτει ξανά σε κάθε κύκλο η αναμενόμενη έξοδος.

Critical Path

Unconstrained Paths - NONE - NONE - Setup

Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	∞	11	11	5	product_reg[3][3]/C	output_reg[13]/D	6.091	3.273	2.818	∞
Path 2	∞	11	11	5	product_reg[3][3]/C	output_reg[14]/D	5.993	3.175	2.818	∞
Path 3	∞	11	11	5	product_reg[3][3]/C	output_reg[12]/D	5.978	3.160	2.818	∞
Path 4	∞	10	10	5	product_reg[3][3]/C	output_reg[9]/D	5.977	3.159	2.818	∞
Path 5	∞	10	10	5	product_reg[3][3]/C	output_reg[11]/D	5.958	3.140	2.818	∞
Path 6	∞	10	10	5	product_reg[3][3]/C	output_reg[10]/D	5.885	3.067	2.818	∞
Path 7	∞	10	10	5	product_reg[3][3]/C	output_reg[8]/D	5.864	3.046	2.818	∞
Path 8	∞	9	9	5	product_reg[3][3]/C	output_reg[7]/D	5.476	2.658	2.818	∞
Path 9	∞	9	9	5	product_reg[3][3]/C	output_reg[6]/D	5.211	2.534	2.677	∞
Path 10	∞	8	8	3	product_reg[6][2]/C	output_reg[5]/D	4.841	2.334	2.507	∞

Το Path 1 είναι το κρίσιμο μονοπάτι.

Κατανάλωση πόρων του FPGA

Πιο κάτω παρατηρούμε τους πόρους του FPGA που χρησιμοποιούνται.

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	93	17600	0.53
FF	144	35200	0.41
IO	31	100	31.00
BUFG	1	32	3.13

Ζητούμενο 3: Στην τρίτη αρχιτεκτονική ζητείται να μετατρέψετε το φίλτρο που υλοποιήσατε στο Ζητούμενο 2 σε παράλληλο. Συγκεκριμένα, θα πρέπει να σχεδιάσετε ένα φίλτρο το οποίο μπορεί να δεχτεί δύο δεδομένα εισόδου παράλληλα και να παράξει τα αντίστοιχα αποτελέσματα συγχρόνως. Η υλοποίηση να γίνει για N=8 bits εύρος δεδομένων x και M=8 συντελεστές.
Η μετατροπή ενός FIR φίλτρου (π.χ. 4-tap) σε παράλληλο με βαθμό παραλληλίας 2 γίνεται ως εξής:

- ***Η αρχική εξίσωση του φίλτρου είναι: y[n] = x[n]*h[0] + x[n-1]*h[1] + x[n-2]*h[2] + x[n-3]*h[3]***

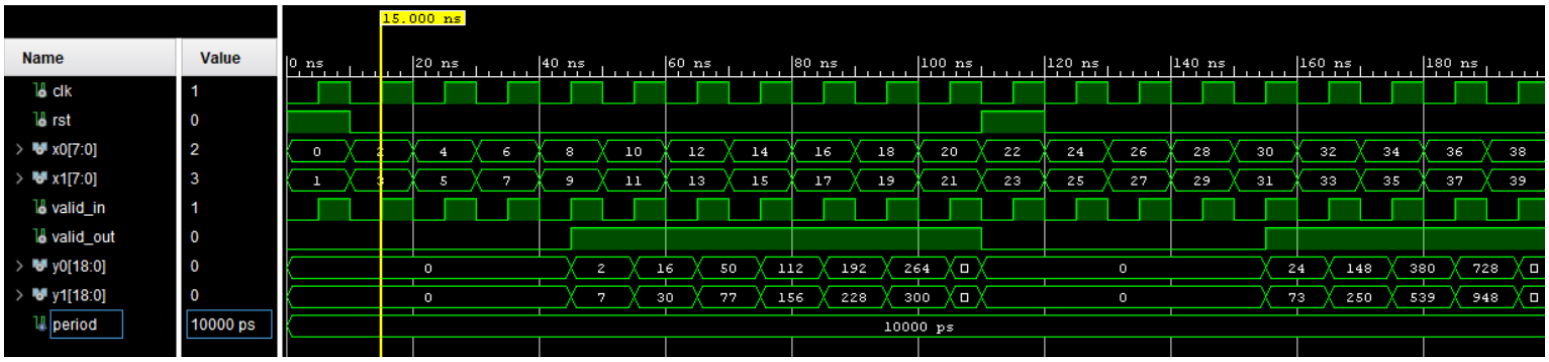
- ***Αφού θέλουμε να επεξεργαζόμαστε 2 σήματα εισόδου παράλληλα, αντικαθιστώντας στην αρχική εξίσωση το n με 2k και 2k+1, έχουμε:***

y[2k] = x[2k]*h[0] + x[2k-1]*h[1] + x[2k-2]*h[2] + x[2k-3]*h[3]
y[2k+1] = x[2k+1]*h[0] + x[2k+1-1]*h[1] + x[2k+1-2]*h[2] + x[2k+1-3]*h[3]

Οι δύο αυτές εξισώσεις δίνουν τα αποτελέσματα για την επεξεργασία των δύο εισόδων παράλληλα.

Ο **source κώδικας** του ερωτήματος και το αντίστοιχο **testbench** βρίσκονται σε αρχεία VHDL στον φάκελο “3”.

Η προσομοίωση (**simulation**) του κυκλώματος φαίνεται πιο κάτω:



Παρατηρούμε ότι σε αυτή την περίπτωση το φίλτρο δέχεται δύο νέες εισόδους x0, x1 σε κάθε κύκλο ρολογιού εφόσον το valid_in ισούται με 1. Έπειτα από κάποια αρχική καθυστέρηση $T_{latency}$ προκύπτουν δύο εξόδοι σε κάθε κύκλο ρολογιού.

Για την απόδειξη της ορθότητας της υλοποίησης μπορούμε να δούμε μερικά από τα αποτελέσματα της προσομοίωσης:

1) Στον 1ο κύκλο ρολογιού πραγματοποιείται το reset.
2) Στον 2ο κύκλο και όταν το valid_in γίνει 1, το φίλτρο δέχεται την πρώτη και δεύτερη είσοδο, x0 = 2 και x1 =3 .
   Έπειτα από 3 κύκλους (αρχικό $T_{latency}$) παρατηρούμε να προκύπτει η πρώτη y[0] και δεύτερη y[1] έξοδος, ενώ το valid_out γίνεται 1.
   Η έξοδος είναι ορθή αφού y[0] = h[0]*x[0] = 1*2 = 2 και
   y[1] = h[0]*x[1] + h[1]*x[0] = 1*3 + 2*2 = 3 + 4 = 7
3) Στον 3ο κύκλο και όταν το valid_in γίνει 1, το φίλτρο δέχεται την τρίτη και τέταρτη είσοδο, x0 = 4 και x1 = 5.
   Παρατηρούμε ότι η έξοδος (λόγω του pipeline) δεν έχει πλέον καθυστέρηση, αλλά

προκύπτει στον επόμενο κύκλο από την 1η έξοδο.
Η τιμή της εξόδου είναι ορθή αφού
y[2] = h[0]*x[2]+h[1]*x[1]+h[2]*x[0] = 1*4 + 2*3 + 3*2 = 4 + 6 + 6 = 16 και
y[3] = h[0]*x[3]+h[1]*x[2]+h[2]*x[1]+h[3]*x[0] = 1*5 + 2*4 + 3*3 + 4*2 = 5 + 8 + 9 + 8 = 30

4) Στον 4ο κύκλο και όταν το valid_in γίνει 1, το φίλτρο δέχεται την πέμπτη και έκτη είσοδο, x0 = 6 και x1 = 7.
Η τιμή της εξόδου είναι ορθή αφού
y[4] = h[0]*x[4]+h[1]*x[3]+h[2]*x[2]+h[3]*x[1]+h[4]*x[0]
= 1*6 + 2*5 + 3*4 + 4*3 + 5*2 = 50 και
y[5] = h[0]*x[5]+h[1]*x[4]+h[2]*x[3]+h[3]*x[2]+h[4]*x[1]+h[5]*x[0]
= 1*7 + 2*6 + 3*5 + 4*4 + 5*3 + 6*2 = 7 + 12 + 15 + 16 + 15 + 12 = 77

Παρατηρούμε ότι σε κάθε επόμενο κύκλο ρολογιού προκύπτει η ορθή έξοδος.
Όταν το reset γίνει 1, όλα μηδενίζονται και το φίλτρο ξεκινά από την αρχή με νέα είσοδο.
Μετά από κάποιο αρχικό $T_{latency}$ προκύπτει ξανά σε κάθε κύκλο η αναμενόμενη έξοδος.

Critical Path

Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	11	11	9	product0_reg[3][3]/C	output0_reg[13]/D	6.104	3.273	2.831
Path 2	∞	11	11	8	product1_reg[3][3]/C	output1_reg[13]/D	6.101	3.273	2.828
Path 3	∞	11	11	9	product0_reg[3][3]/C	output0_reg[14]/D	6.006	3.175	2.831
Path 4	∞	11	11	8	product1_reg[3][3]/C	output1_reg[14]/D	6.003	3.175	2.828
Path 5	∞	11	11	9	product0_reg[3][3]/C	output0_reg[12]/D	5.991	3.160	2.831
Path 6	∞	10	10	9	product0_reg[3][3]/C	output0_reg[9]/D	5.990	3.159	2.831
Path 7	∞	11	11	8	product1_reg[3][3]/C	output1_reg[12]/D	5.988	3.160	2.828
Path 8	∞	10	10	8	product1_reg[3][3]/C	output1_reg[9]/D	5.987	3.159	2.828
Path 9	∞	10	10	9	product0_reg[3][3]/C	output0_reg[11]/D	5.971	3.140	2.831
Path 10	∞	10	10	8	product1_reg[3][3]/C	output1_reg[11]/D	5.968	3.140	2.828

Το Path 1 είναι το κρίσιμο μονοπάτι.

Κατανάλωση πόρων του FPGA

Δεξιά παρατηρούμε τους πόρους του FPGA που χρησιμοποιούνται.

Utilization | Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	187	17600	1.06
FF	230	35200	0.65
IO	58	100	58.00
BUFG	1	32	3.13

Να καταγράψετε και να συγκρίνετε τους πόρους (resources) που χρησιμοποιήθηκαν για την υλοποίηση του κάθε φίλτρου καθώς και τη συχνότητα λειτουργίας του FPGA ελέγχοντας το κρίσιμο μονοπάτι.

Παρατηρούμε ότι στις δύο πρώτες υλοποιήσεις οι πόροι που χρησιμοποιούνται είναι στα ίδια επίπεδα, ενώ στην τελευταία υλοποίηση του φίλτρου οι πόροι είναι αρκετοί περισσότεροι, καθώς υπολογίζονται διπλά αποτελέσματα σε κάθε κύκλο ρολογιού. Το κρίσιμο μονοπάτι με τη μεγαλύτερη καθυστέρηση προκύπτει από την παράλληλη υλοποίηση του φίλτρου.