

Ψηφιακά Συστήματα ΗΥ σε Χαμηλά Επίπεδα Λογικής II

Compulsory coursework

Μαρινόπουλος Χριστόφορος 10522

Εαρινό εξάμηνο 2025

Περιεχόμενα

Main module	2
Normalization module	4
Rounding module	4
Exception Module	6
Main module's logic	6
Testbench	7
System Verilog Assertions	8
Immediate Assertions	8
Concurrent Assertions	9

Main module

Σε αυτή την ενότητα θα αναλυθεί το main module, δηλαδή τα modules από τα οποία αποτελείται καθώς και η εσωτερική λογική του main module. Το main module είναι ένας πολλαπλασιαστής κινητής υποδιαστολής ο οποίος έχει ως εισόδους τα σήματα:

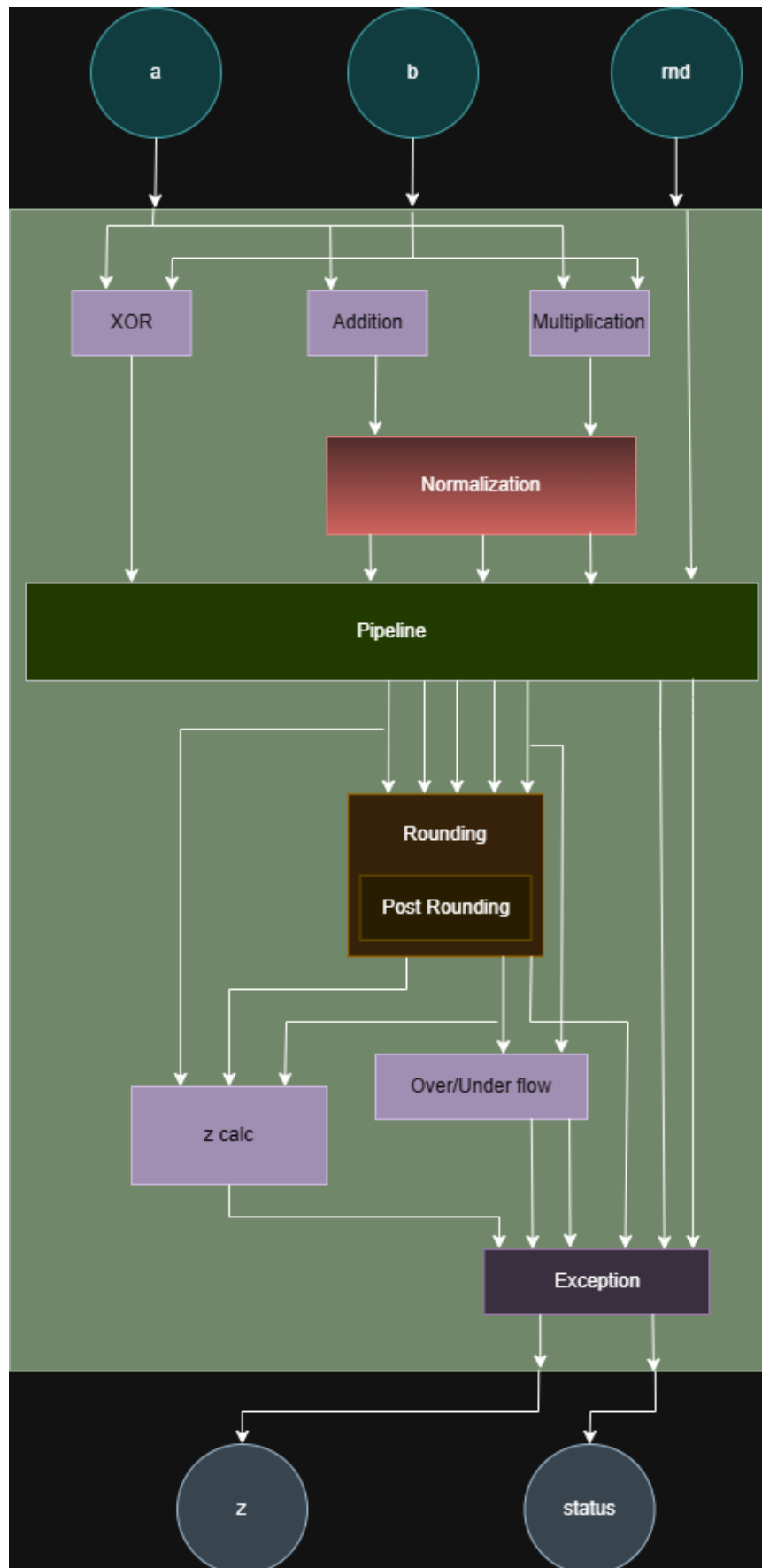
- logic [31:0] a, b: Οι αριθμοί που θα πολλαπλασιαστούν.
- logic [2:0] rnd: Η μέθοδος στρογγυλοποίησης.
- logic clock: Το ρολόι του module.
- logic reset: Το reset του module.

ως εξόδους τα σήματα:

- logic [31:0] z: Το αποτέλεσμα του πολλαπλασιασμού.
- logic [7:0] status: Ένα διάνυσμα ελέγχου.










και είναι σχεδιασμένος βάσει του προτύπου IEEE-754.

Η Εικόνα 1 απεικονίζει τη δομή του main module, ενώ ο Πίνακας 1 περιλαμβάνει την ερμηνεία των χρωμάτων που χρησιμοποιούνται για την αναπαράσταση των λειτουργικών τμημάτων.



Εικόνα 1: Μπλοκ διάγραμμα του main module

Πίνακας 1: Αντιστοίχιση χρωμάτων

	Main module
	Data inputs (a, b, rnd)
	Main modules' internal logic
	Normalization module
	Pipeline data bus
	Rounding module
	Post rounding logic
	Exception handling
	Outputs (z, status)

Normalization module

Το normalization module έχει τις ακόλουθες εισόδους:

- input logic [47:0] mult_res
- input logic [9:0] addition

και τις ακόλουθες εξόδους:

- logic [9:0] exponent
- logic [22:0] mantissa
- logic guard
- logic sticky

Επίσης, αποτελείται από τα εξής εσωτερικά σήματα τα οποία είναι οι πιθανές τιμές των εξόδων:

1. logic [9:0] poss_exponent: Εάν $\text{mult_res}[47] == 1$ τότε $\text{exponent} = \text{poss_exponent}$.
2. logic [22:0] poss_mantissa_1: Εάν $\text{mult_res}[47] == 1$ τότε $\text{mantissa} = \text{poss_mantissa_1}$.
3. logic [22:0] poss_mantissa_0: Εάν $\text{mult_res}[47] == 0$ τότε $\text{mantissa} = \text{poss_mantissa_0}$.
4. logic poss_guard_bit_1: Εάν $\text{mult_res}[47] == 1$ τότε $\text{guard} = \text{poss_guard_bit_1}$.
5. logic poss_guard_bit_0: Εάν $\text{mult_res}[47] == 0$ τότε $\text{guard} = \text{poss_guard_bit_0}$.
6. logic [22:0] poss_sticky_bit_1: Εάν $\text{mult_res}[47] == 1$ τότε $\text{sticky} = \text{poss_sticky_bit_1}$.
7. logic [21:0] poss_sticky_bit_0: Εάν $\text{mult_res}[47] == 0$ τότε $\text{sticky} = \text{poss_sticky_bit_0}$.

Η ανάθεση των τιμών στις εξόδους ακολουθεί κατά γράμμα τις οδηγίες της εκφώνησης και υλοποιείται μέσω των `always_comb` blocks του module. ■

Rounding module

Το rounding module έχει τις ακόλουθες εισόδους:

- logic [2:0] round: είδος στρογγυλοποίησης
- logic [9:0] pre_round_exponent: {2 overflow underflow bits, Εκθετικό μέρος του πολλαπλασιασμού (σύμφωνα με το πρότυπο IEEE-754) }

- logic [23:0] mantissa: {hidden bit (1), δεκαδικό μέρος του πολλαπλασιασμού (σύμφωνα με το πρότυπο IEEE-754)}
- logic guard: επόμενο bit της mantissa
- logic sticky: or όλων των bit μετά το guard
- logic sign: Πρόσημο του αποτελέσματος του πολλαπλασιασμού

και τις ακόλουθες εξόδους:

- logic [24:0] result: overflow bit, hidden bit (1), Η mantissa του αποτελέσματος μετά το rounding
- logic inexact: Σήμα που υποδεικνύει εάν ο πολλαπλασιασμός δεν ήταν ακριβής (κάποιο ψηφίο του αποτελέσματος δεν αποτυπώνεται στην mantissa λόγω του προτύπου)
- logic [9:0] post_round_exponent Το εκθετικό μέρος του αποτελέσματος μετά το rounding.

Θέτει το σήμα inexact στο 1 εάν το δεκαδικό μέρος του αποτελέσματος του πολλαπλασιασμού (mantissa) δεν είναι ακριβές. Αυτό το καταλαβαίνει από τις εισόδους guard, sticky. Εάν το guard είναι 1, αυτό σημαίνει πως υπάρχει σημαντικό ψηφίο αμέσως μετά την mantissa, το οποίο μπορεί να επηρεάσει τη στρογγυλοποίηση. Εάν το sticky είναι 1, αυτό σημαίνει πως υπήρχε πληροφορία στο δεκαδικό μέρος του αποτελέσματος του πολλαπλασιασμού που δεν μπορεί να αποτυπωθεί στα διαθέσιμα bit του σήματος αποτελέσματος, θα χρειαζόταν δηλαδή mantissa με μεγαλύτερο μήκος. Συνεπώς, inexact = guard | sticky.

Θέτει το είδος πιθανής στρογγυλοποίησης ανάλογα με την είσοδο round μέσω ενός always_comb block. Η κωδικοποίηση του σήματος στρογγυλοποίησης παρουσιάζεται στον Πίνακα 2.

Πίνακας 2: Αντιστοίχιση round με round_enum

round	Είδος στρογγυλοποίησης (round_enum)
000	IEEE_near
001	IEEE_zero
010	IEEE_pinf
011	IEEE_ninf
100	near_up
101	away_zero
default	IEEE_near

Εάν το inexact είναι 1 τότε θα πραγματοποιήσει το είδος της στρογγυλοποίησης στην mantissa που του υποδεικνύει το σήμα round_enum.

Εάν το σήμα inexact δεν είναι 1 τότε δεν πραγματοποιεί κάποια στρογγυλοποίηση στην mantissa καθώς δεν υπάρχει ανάγκη. Τέλος, αφού πραγματοποιήσει την στρογγυλοποίηση της mantissa, εάν αυτή χρειάζεται, ελέγχει αν υπάρχει overflow στην mantissa (το msb της αν είναι 1 δηλαδή).

(Το συγκεκριμένο κομμάτι λογικής θα μπορούσε να είχε υλοποιηθεί εκτός του rounding module, η επιλογή αυτή έγινε καθώς η περιγραφή αυτής της λειτουργίας, από την εκφώνηση, έγινε εντός της ενότητας rounding module).

Αν είναι 1:

```
result = {1'b1, pre_round_mantissa[24:1]}
post_round_exponent = pre_round_exponent + 1
```

Αν δεν είναι 1:

```
result = pre_round_mantissa
post_round_exponent = pre_round_exponent + 1
```



Exception module

Το exception module δέχεται ως είσοδο τα ακόλουθα σήματα:

- logic [31:0] a,b: Οι αριθμοί που πολλαπλασιάζονται
- logic [31:0] z_calc: Το αποτέλεσμα προτού ελεγχθεί αν είναι λανθασμένο
- logic inexact: Σήμα που υποδεικνύει πως το αποτέλεσμα δεν είναι ακριβές.
- logic overflow: Σήμα που υποδεικνύει εάν υπήρξε overflow στον υπολογισμό του αποτελέσματος
- logic underflow: Σήμα που υποδεικνύει εάν υπήρξε logic underflow στον υπολογισμό του αποτελέσματος
- logic [2:0] round: Σήμα που υποδεικνύει το είδος της στρογγυλοποίησης που πραγματοποίησε, εάν χρειαζόταν, το rounding module.

και έχει ως εξόδους:

- logic zero_f: Μέρος διανύσματος ελέγχου.
- logic inf_f: Μέρος διανύσματος ελέγχου.
- logic nan_f: Μέρος διανύσματος ελέγχου.
- logic tiny_f: Μέρος διανύσματος ελέγχου.
- logic huge_f: Μέρος διανύσματος ελέγχου.
- logic inexact_f: Μέρος διανύσματος ελέγχου.
- logic [31:0] z: Τελικό αποτέλεσμα.

Αφού μηδενίσει το διάνυσμα ελέγχου (status) καλεί δύο φορές την συνάρτηση num_interp με ορίσματα a,b και ανάλογα με τα δεδομένα επιστροφής της συνάρτησης υπολογίζεται το τελικό αποτέλεσμα καθώς και ενεργοποιούνται τα σωστά flags.

Σημείωση: Η συνάρτηση z_num υλοποιήθηκε καθώς υποδεικνύεται από την εκφώνηση αλλά δεν χρησιμοποιήθηκε, διότι η λογική της εξόδου προέκυψε από bitwise χειρισμό. ■

Main module's logic

Το main module αποτελείται από 3 υπο modules (normalization, rounding, exception) τα οποία έχουν αναλυθεί παραπάνω, ένα pipeline stage καθώς και μερικά κομμάτια λογικής. Τα:

1. XOR
2. Addition
3. Multiplication

ακολουθούν πιστά την εκφώνηση και δεν υπάρχει ανάγκη ανάλυσής τους.

Το σήμα z_calc παράγεται ως εξής:

1. [31] z_calc = sign
2. [30:23] z_calc = [7:0] post_round_exponent
3. [22:0] z_calc = [22:0] result

και η παραγωγή του είναι αυτή καθώς τα σήματα result και post_round_exponent έχουν παραχθεί κατάλληλα εντός του Post Rounding block του Rounding module.

To Over/Under flow block παράγει τα σήματα overflow underflow με τον εξής τρόπο:

- overflow = (post_round_exponent[9] == 1'b0 && post_round_exponent > 9'd254)
- underflow = (post_round_exponent[9] == 1'b1 || post_round_exponent[8:0] == 9'b0)

To 254 αντιστοιχεί στον μεγαλύτερο επιτρεπτό εκθέτη (biased) πριν εισέλθουμε στην κατηγορία των special values (Inf/-NaN), σύμφωνα με το IEEE-754. Ο εκθέτης 255 δηλώνει Infinity ή NaN, οπότε το μέγιστο αποδεκτό πριν από overflow είναι 254.

To 1 αντιστοιχεί στον μικρότερο επιτρεπτό εκθέτη (biased) σύμφωνα με το IEEE-754. Έτσι εάν ο εκθέτης είναι μικρότερος του 1 (αρνητικός ή 0) υπάρχει underflow. ■

Testbench

To testbench αφού κάνει instantiate το fp_mult_top και bind τα assertion modules με το fp_mult_top

1. Παράγει το σήμα ρολογιού clock που έχει περίοδο 10 ns.
2. Πραγματοποιεί για κάθε rounding mode 60000 πολλαπλασιασμούς και ελέγχει το παραγόμενο αποτέλεσμα με το αποτέλεσμα της συνάρτησης multiplication.
3. Πραγματοποιεί κάθε έναν από του γωνιακούς συνδυασμούς (corner cases) και ελέγχει το παραγόμενο αποτέλεσμα με το αποτέλεσμα της συνάρτησης multiplication.

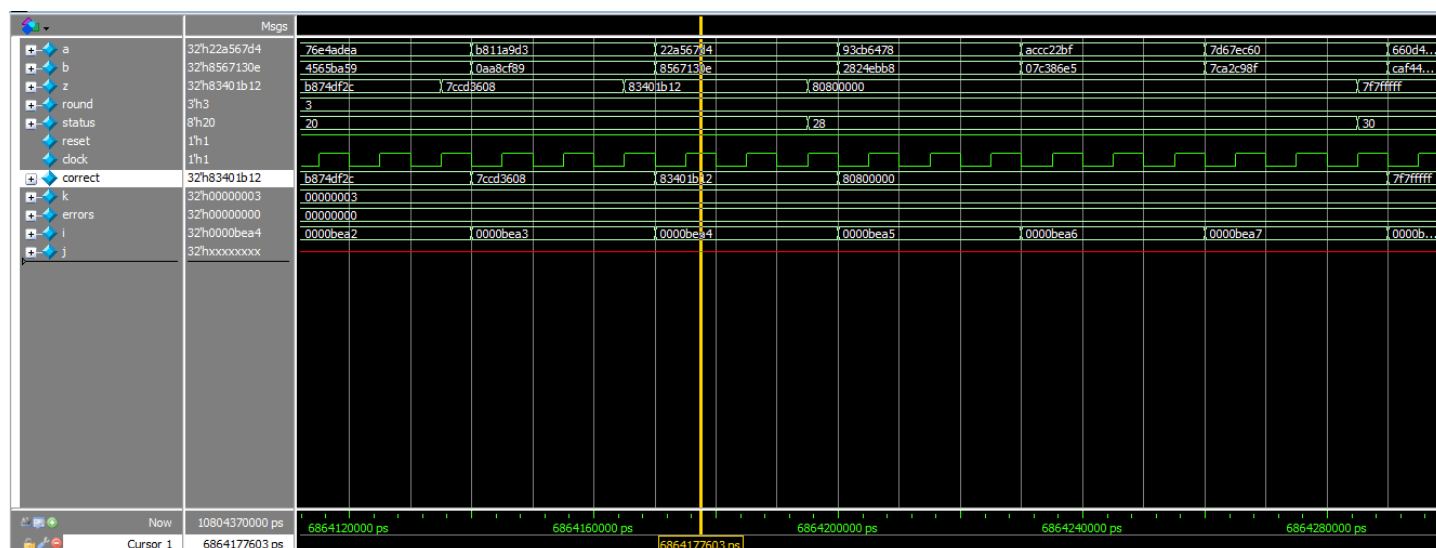
Κατά την διάρκεια των πολλαπλασιασμών, εάν το παραγόμενο αποτέλεσμα δεν είναι ίσο με το αποτέλεσμα της συνάρτησης multiplication αυξάνεται η μεταβλητή errors και ενημερώνει τον χρήστη πρώτα με ένα μήνυμα σφάλματος και τελικά με ένα μήνυμα που αναφέρει το συνολικό πλήθος των σφαλμάτων.

Με την εκτέλεση της προσομοίωσης, παρατηρείται στην Εικόνα 2, πως τα αποτελέσματα του fp_mult_top είναι ίδια με της συνάρτησης multiplication και δεν προέκυψε κανένα σφάλμα.

```
VSIM 46> run -all
# Rounding mode is : (IEEE_near)
# Rounding mode is : (IEEE_zero)
# Rounding mode is : (IEEE_pinf)
# Rounding mode is : (IEEE_ninf)
# Rounding mode is : (near_up)
# Rounding mode is : (away_zero)
# Round mode check pass. No errors encountered
# Corner case check pass. No errors encountered
# ** Note: $stop      : C:/Users/makat/Desktop/Assignments/HW2/fp_mult_tb.sv(168)
#   Time: 10804370 ns  Iteration: 0   Instance: /fp_mult_tb
# Break in Module fp_mult_tb at C:/Users/makat/Desktop/Assignments/HW2/fp_mult_tb.sv line 168
```

Εικόνα 2: Transcript after simulation

Επίσης, στην Εικόνα 3 παρατηρείται ένα στιγμιότυπο από τις κυματομορφές κατά τη διάρκεια της προσομοίωσης.



Εικόνα 3: Στιγμιότυπο των κυματομορφών κατά τη διάρκεια της προσομοίωσης.

System Verilog Assertions

Immediate Assertions

Το πρώτο ζητούμενο αυτού του υποερωτήματος είναι ο καθορισμός των status bits που δεν επιτρέπεται να είναι ταυτόχρονα ενεργά. Ο Πίνακας 3 παρουσιάζει αυτούς τους συνδυασμούς καθώς και την αιτιολόγησή τους.

Πίνακας 3: Μη επιτρεπτοί συνδυασμοί status bits και η αιτιολόγησή τους.

Status Bit 1	Status Bit 2	Reason
zero_f	inf_f	Zero and Infinity cannot occur simultaneously
zero_f	nan_f	Zero cannot be NaN
zero_f	huge_f	Zero cannot be the result of an overflow
inf_f	tiny_f	Infinity cannot be tiny
nan_f	tiny_f	NaN is not a result of underflow
nan_f	huge_f	NaN is not a result of overflow
nan_f	inexact_f	NaN does not come from inexact
tiny_f	huge_f	Underflow and overflow cannot occur together

Σημείωση: Στην παρούσα υλοποίηση, οι NaNs αντιμετωπίζονται ως ισοδύναμα των infinities. Συνεπώς, ο συνδυασμός nan_f και inf_f θεωρείται επιτρεπτός.

Το δεύτερο ζητούμενο είναι η κατασκευή ενός module το οποίο θα γίνει bind με DUT και θα ελέγχει εάν υπάρχουν ταυτόχρονα ενεργά status bits τα οποία δεν θα έπρεπε να είναι. Αυτό το module είναι το test_status_bits και έχει ως εισόδους:

- logic clk: Πολόι.
- logic [7:0] status: Διάνυσμα ελέγχου.

Εντός του, γίνεται ο έλεγχος εάν υπάρχει ενεργός κάποιος αδύνατος συνδυασμός και αν υπάρχει τυπώνεται ένα μήνυμα σφάλματος. Στην Εικόνα 2 παρατηρείται πως δεν υπήρξε κανένας αδύνατος συνδυασμός κατά την διάρκεια της προσομοίωσης.

Concurrent Assertions

Για τα concurrent assertions δημιουργήθηκε ένα module με όνομα `test_status_z_combinations` και έχει ως εισόδους:

- logic clk: Ρολόι
- logic [7:0] status: Διάνυσμα ελέγχου.
- logic [31:0] z: Τελικό αποτέλεσμα.
- logic [31:0] a,b: Αριθμοί που πολλαπλασιάζονται.

Τα assertions που δημιουργήθηκαν ακολουθούν τις εντολές της εκφώνησης και τυπώνουν ένα μήνυμα σφάλματος εάν δεν είναι επιτυχημένα. Επιτηδευμένα δεν τυπώνουν κάποιο μήνυμα επιτυχίας καθώς στο testbench ελέγχονται πάνω από 300000 συνδυασμοί και εύκολα κάποιο μήνυμα λάθους θα χανόταν μέσα σε αυτούς.

Στην Εικόνα 2 παρατηρείται πως δεν υπήρξε κάποιο assertion που να μην ήταν πετυχημένο.

