Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΑΝΑΠΤΥΞΗ ΥΛΙΚΟΥ – ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Project - 1η φάση / ορόσημο

Σχεδίαση φίλτρου FIR με χρήση συνθέσιμης VHDL και υλοποίηση σε FPGA

Διδάσκων: Ηλίας Μανωλάκος, Καθηγητής

Βοηθός Εργαστηρίου: Ηλίας Κουσκουμβεκάκης, ΥΔ

Μάρτιος 2023

Εισαγωγή

Στόχοι αυτού του πρώτου οροσήμου του project είναι:

1. Η περαιτέρω εξοικείωση με την αναπτυξιακή κάρτα και τα εργαλεία σχεδίασης,
2. Η εξάσκηση στη παραμετρική περιγραφή ενός Επεξεργαστή Ειδικού Σκοπού (ΕΕΣ) με πολλά στάδια σε VHDL,
3. Η περιγραφή και σύνθεση ενός παραμετρικού συστήματος υλικού, καθώς και ο έλεγχος ολόκληρου του σχεδίου πριν και μετά τη σύνθεση.

Περιγραφή

Καλείστε να σχεδιάσετε και να υλοποιήσετε δύο φίλτρα (**A** και **B**) πεπερασμένης κρουστικής απόκρισης (finite impulse response, FIR) τα οποίο να είναι παραμετρικά ως προς την τάξη τους (πλήθος των συντελεστών, taps) και το εύρος των δεδομένων εισόδου/εξόδου. Θα πρέπει να χρησιμοποιήσετε σωλήνωση (pipelining), ώστε το σύστημα του κάθε φίλτρου να έχει την δυνατότητα να παράγει μια έξοδο σε κάθε κύκλο ρολογιού.

Στην γενική περίπτωση ένα φίλτρο FIR έχει την παρακάτω μαθηματική περιγραφή:

$$y(n) = \sum_{k=0}^{M} h(k)x(n-k) = h(0)x(n) + h(1)x(n-1) + \ldots + h(M)x(n-M)$$

Όπου:

N = M+1: η τάξη του φίλτρου + 1, δηλαδή ο συνολικός αριθμός των taps

x(n): η τιμή εισόδου-δείγματος την χρονική στιγμή n

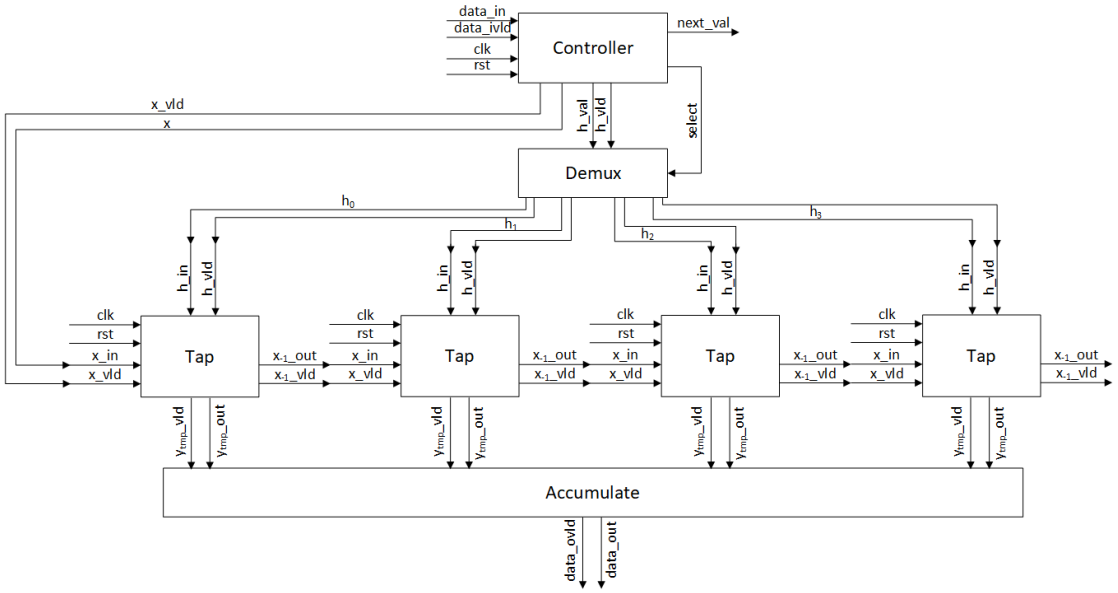
y(n): η έξοδος του φίλτρου την χρονική στιγμή n

h(k): ο k-οστός συντελεστής του φίλτρου, k = 0, 1, …, M

Υποθέστε ότι η είσοδος **x(n)** και η έξοδος **y(n)** είναι προσημασμένοι αριθμοί σε μορφή συμπληρώματος ως προς 2 (2's complement), σταθερής υποδιαστολής (fixed point) και έχουν όλοι το ίδιο πλήθος δεκαδικών ψηφίων. Επιπλέον, κάθε τιμή θα παρουσιάζεται στο σύστημά σας με ένα σήμα εγκυρότητας (valid signal). Για να είναι πιο εύκολη η υλοποίηση του κάθε φίλτρου με παραμετρικό πλήθος taps και σωλήνωση, σχεδιάστε με περιγραφή συμπεριφοράς για ένα tap του φίλτρου και στην συνέχεια με χρήση της δομής 'for-generate' δημιουργήστε το FIR φίλτρο που θα περιέχει **N** taps. Το κάθε φίλτρο κατά την αρχικοποίηση θα αναμένει να δεχτεί τους συντελεστές για κάθε tap και κατά την κανονική λειτουργία τα δεδομένα εισόδου.

FIR Φίλτρο A

Στο Σχηματικό 1 παρακάτω παρουσιάζεται η αρχιτεκτονική για το πρώτο FIR φίλτρο που πρέπει να υλοποιήσετε, με **N = 4** και εύρος δεδομένων εισόδου (**x(n)**) **W = 8** bit στη μορφή 5.3 (**K = 5** bit ακέραιο και **L = 3** bit δεκαδικό μέρος).



Σχηματικό 1 - Η μονάδα FIR_Design_A

Το εύρος δεδομένων εξόδου (**y(n)**) θα πρέπει να έχει το ίδιο πλήθος δεκαδικών ενώ το ακέραιο μέρος πρέπει να έχει τουλάχιστον τα διπλάσια της εισόδου. Πρέπει να είναι δηλαδή 10.3 = 13 bits (10 bit integer + 3 bit fractional). Αν θέλετε, για μεγαλύτερη ακρίβεια από αυτήν που απαιτείται, μπορείτε να χρησιμοποιήσετε και 16 bit εύρος δεδομένων εξόδου, να είναι δηλαδή 10.6 (10 bit integer + 6 bit fractional).

Το εύρος δεδομένων εισόδου **W** (ακέραιο μέρος **K** και δεκαδικό μέρος **L**) καθώς και ο αριθμός **N** των taps θα πρέπει οπωσδήποτε να ορίζονται παραμετρικά στο top-level αρχείο, χρησιμοποιώντας ‘generics’ της VHDL, τα οποία θα περνάτε μέσα σε όποια μονάδα τα χρειάζεται με αντίστοιχα ‘generics’ μέσα σε αυτές τις μονάδες και ‘generic maps’ κατά το instantiation αυτών. Ενδεικτικές τιμές των **W, K, L** είναι οι εξής:
W = 8 (**K** = 5, **L** = 3), **W** = 16 (**K** = **L** = 8), **W** = 32 (**K** = **L** = 16), ή **W** = 64 (**K** = **L** = 32).

Η μονάδα **Controller** είναι υπεύθυνη για την δρομολόγηση των δεδομένων και θα πρέπει να υλοποιηθεί με λογική Finite State Machine (FSM). Μετά από κάθε αρχικοποίηση (reset) αναμένει να λάβει τους συντελεστές **h(k)** του φίλτρου και τους στέλνει στην κατάλληλη μονάδα **Tap** προς αποθήκευση, μέσω της μονάδας **Demux**. Μετά την λήψη όλων των συντελεστών ξεκινάει η λήψη των δεδομένων εισόδου **x(n)**

τα οποία αποστέλλονται στην πρώτη μονάδα **Tap** στην είσοδο **x_in** και από εκεί διοχετεύονται στις υπόλοιπες **Tap** μονάδες.

Τα δεδομένα εισόδου **data_in** θα πρέπει να έχουν εύρος **D** = 32, 64, 128, ή 256 bit, παραμετροποιήσιμο με αντίστοιχο generic. Η λήψη νέων δεδομένων γίνεται μετά από αίτηση, η οποία επιτυγχάνεται με το σήμα **next_val**. Αυτό σημαίνει πως, για π.χ **W = 8** και **D = 32**, σε κάθε κύκλο ρολογιού που αυτό το σήμα είναι ενεργό θα μπαίνουν στη μονάδα **Controller** 4 τιμές των συντελεστών **h(k)** ή 4 τιμές δεδομένων εισόδου του φίλτρου **x_in**. Συνεπώς το σήμα **next_val** θα πρέπει να ενεργοποιείται κατά μέγιστο μια φορά (κύκλο) κάθε 4 κύκλους ρολογιού, ώστε να προλαβαίνουν να φύγουν τα προηγούμενα δεδομένα από τη μονάδα **Controller** πριν έρθουν τα επόμενα και έτσι τελικά να μη χάνονται δεδομένα. Για **W = 16 ή 32** θα πρέπει να μπαίνουν 2 ή 1 τιμές αντίστοιχα. Ομοίως για άλλους συνδυασμούς των **D** και **W**.

Η μονάδα **Demux** είναι υπεύθυνη για την προώθηση των συντελεστών **h(k)** στην κατάλληλη μονάδα **Tap**. Προσπαθήστε να περιγράψετε την υλοποίηση με τέτοιο τρόπο ώστε να λειτουργεί για παραμετρικό πλήθος από taps. Επίσης δεν είναι υποχρεωτικό να φτιάξετε ξεχωριστή μονάδα - οντότητα και μπορείτε για ευκολία να ενσωματώσετε την υλοποίηση της στην αρχιτεκτονική της μονάδας **FIR_Design_A** που περιγράφεται παρακάτω.

Η μονάδα **Tap** είναι υπεύθυνη για τον πολλαπλασιασμό της εισερχόμενης τιμής **x(n)** με τον συντελεστή της **h(k)**. Η μονάδα εξάγει την τιμή **x_in** μετά από έναν κύκλο στο **x_out** και την τιμή εξόδου που υπολογίστηκε στο **y_out**.

Η μονάδα **Accumulate** είναι υπεύθυνη για την άθροιση όλων των γινομένων **y_out**, ώστε να παραχθεί η τελική τιμή εξόδου **y(n)** του φίλτρου (**data_out**), κάθε χρονική στιγμή. Έγκυρες τιμές εξόδου **y(n)** θεωρούνται αυτές που παράγονται όταν και τα **N** taps βγάλουν μη μηδενικές τιμές **y_out,** οπότε το σήμα **data_valid** θα πρέπει να ενεργοποιείται μόνο για αυτές.

Θα πρέπει υποχρεωτικά να υλοποιήσετε τις μονάδες **Controller** και **Tap** ως ξεχωριστες μονάδες - οντότητες (entities) σε ξεχωριστά αρχεία VHDL. Οι όποιες υπόλοιπες μονάδες (π.χ οι μονάδες **Demux** και **Accumulate** αν επιλέξετε να είναι ξεχωριστές μονάδες) καθώς και το insantiation (port map) των ξεχωριστών μονάδων **Controller** και **Tap**, θα πρέπει να τοποθετηθούν σε μια μονάδα **FIR_Design_A** η οποία θα έχει τις εισόδους και εξόδους που απεικονίζονται στο παραπάνω σχήμα.

FIR Φίλτρο B

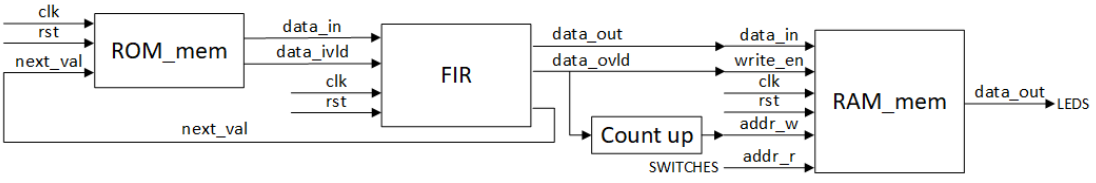
Το FIR φίλτρο A εχει κρίσιμο μονοπάτι του οποίου το μήκος εξαρτάται από το πλήθος **N** των μονάδων **Tap** που περιέχει λόγω της δομής των αθροιστών που συσσωρεύει τα επιμέρους γινόμενα και παράγει την έξοδο σε κάθε κύκλο. Αφού ολοκληρώσετε και επαληθεύσετε πλήρως την ορθή λειτουργία του φίλτρου A, καλείστε να σχεδιάσετε και να υλοποιήσετε ένα βελτιστοποιημένο FIR φίλτρο B το οποίο να έχει κρίσιμο μονοπάτι ανεξάρτητο του πλήθους **N** των μονάδων **Tap** που περιέχει. Μπορείτε να χρησιμοποιήσετε ήδη υλοποιημένες μονάδες από το πρώτο φίλτρο εφόσον αυτές είναι κατάλληλες, ή να τις αλλάξετε ώστε να ταιριάζουν.

Σύστημα Δοκιμής (Φίλτρο A & B)

Για να είναι δυνατός ο έλεγχος του κάθε φίλτρου και άρα των μονάδων **FIR_Design_A/B** στην αναπτυξιακή κάρτα, θα πρέπει δημιουργήσετε μια top-level **FIR_Test** μονάδα στην οποία θα τοποθετήσετε την κάθε μονάδα **FIR_Design_A/B** συνδεδεμένη με τις **ROM** και **RAM** block μνήμες εισόδου και εξόδου αντίστοιχα που σας δίνονται έτοιμες στο eClass (‘ROM_mem.vhd’, ‘RAM_mem.vhd’). Το Σχηματικό 2 παρακάτω παρουσιάζει ένα ενδεικτικό σχέδιο της μονάδας δοκιμής **FIR_Test**.

Είναι σημαντικό οι μονάδες **FIR_Design_A/B** μέσα τους να ΜΗΝ περιέχουν τις μονάδες δοκιμής (π.χ. **ROM** και **RAM**) που περιγράφονται στις επόμενες παραγράφους, ώστε να μπορέσετε στην επόμενη φάση του project (επόμενη άσκηση) να τις επαναχρησιμοποιήσετε χωρίς καθόλου αλλαγές. Επίσης εφόσον τα δεδομένα εισόδου και εξόδου μπορεί να έχουν μεταβλητό εύρος θα πρέπει οι **ROM** και **RAM** μνήμες σας να είναι σωστά παραμετροποιημένες.

Στη **ROM** μνήμη εισόδου υπάρχουν αρχικά 4 συντελεστές, ένας για κάθε ένα από τα **N = 4** ενδεικτικά taps του φίλτρου. Οι επόμενες 8 ενδεικτικές τιμές της μνήμης είναι για την είσοδο **x(n)** του φίλτρου και επαναλαμβάνονται συνεχώς αν ζητηθούν πάνω από 8 αναγνώσεις.



Σχηματικό 2 - Η μονάδα FIR_Test

Τα αποτελέσματα **y(n)** θα πρέπει να αποθηκεύονται στη **RAM** μνήμη εξόδου και στην συνέχεια να εμφανίζονται στα 8 LED της αναπτυξιακής κάρτας. Επειδή το εύρος δεδομένων των αποτελεσμάτων **y(n)** είναι μεγαλύτερο από 8 bits που χωράνε στα LED, μπορείτε να μην εμφανίζετε κάποια από τα bits του ακέραιου μέρους. Η διεύθυνση ανάγνωσης από την μνήμη **RAM** εξόδου θα δίνεται μέσω των 8 switch της κάρτας και άρα η μνήμη σας θα πρέπει να μπορεί να χωρέσει 256 τιμές. Εφόσον η **ROM** σας περιέχει λιγότερες **x(n)** τιμές από αυτές τις 256, η RAM σας θα γεμίσει με 'num_y / num_x' φορές τις ίδες τιμές εξόδου, θα αρχίσουν δηλαδή οι τιμές της **RAM** να επαναλαμβάνονται.

Τέλος για να προσομοιώσετε τη μονάδα δοκιμής **FIR_Test** θα πρέπει να φτιάξετε ένα testbench **tb_FIR_Test** που να την κάνει instantiate μαζί με ένα proces για το ρολόι clk και στη συνέχεια το stimulus process να κάνει assert / de-assert το σήμα rst και να αυξάνει την είσοδο switches, που αντιστοιχεί στα switch της κάρτας και άρα στη διεύθυνση ανάγνωσης της **RAM,** από 0 ως 255.

Συχνότητα λειτουργίας και Επιφάνειας- Κλιμάκωση

Αφού ολοκληρώσετε και επιβεβαιώσετε την ορθή λειτουργία της υλοποίηση σας, μετρήστε τη μέγιστη συχνότητα (timing) και την ανάγκη σε FPGA resources (area) που το κάθε φίλτρο σας μπορεί να υλοποιηθεί χωρίς timing errors για **D** = 32, **W** = 8 και **N** = *4, 8, 16, 32, 64, ή 128* και καταγράψτε τα συμπεράσματα σας με σχετικά διαγράμματα και περιγραφή/συζήτηση αυτών.

Bonus (*Προαιρετικά, για έξτρα πόντους που θα ανακοινωθούν αργότερα)*

N/W Assessment: Μετρήστε τη μέγιστη συχνότητα (timing) και την ανάγκη σε FPGA resources (area) που το κάθε φίλτρο σας (A και B) μπορεί να υλοποιηθεί **χωρίς timing errors** για όλα τα δυνατά **N** αλλά αυτή τη φορά πέρα από **D** = 32, **W** = 8 δοκιμάστε και για **D** = 32 και **W** = *16*, ή 32 και επίσης για **D** = 64 και **W** = 32, ή 64. Στη συνέχεια καταγράψτε τα συμπεράσματα σας με σχετικά διαγράμματα και περιγραφή/συζήτηση αυτών. Έχετε κατά νου πως εφόσον τα σχέδια σας είναι σωστά παραμετροποιημένα ως προς **D**, **W, K, L** δε χρειάζονται καθόλου αλλαγές στον κώδικα.

“Hints and tips”

- Θα πρέπει να αναπαραστήσετε όποια εσωτερικά σήματα χρειάζεστε για τις εισόδους και εξόδους των πολλαπλασιαστών και του αθροιστή που θα χρησιμοποιήσετε**,** με τον τύπο ‘signed’ της VHDL από τη IEEE βιβλιοθήκη ‘numeric_std’. Ο συγκεκριμένος τύπος δεδομένων αναπαριστά αριθμούς 2’s complement ώστε να γίνονται σωστά οι πράξεις του πολλαπλασιασμού και της πρόσθεσης. Μπορείτε επίσης να χρησιμοποιείτε με προσοχή τον τύπο ‘integer’, θέτοντας πάντα κατάλληλο ‘range’. Τα σήματα των οντοτήτων (entities) μπορούν να είναι τύπου ‘signed’ ή ‘std_logic_vector’ και θα τα μετατρέπετε πριν ή / και αφού τα χρησιμοποιήσετε στην κάθε μονάδα για πολλαπλασιασμό ή πρόσθεση με κατάλληλες συναρτήσεις μετατροπής ‘std_logic_vector()’, ‘signed()’, ‘to_signed() και to_integer()’. Τα top-level entities θα πρέπει αναγκαστικά να έχουν σήματα τύπου ‘std_logic_vector’.

- Θα πρέπει να δώσετε ιδιαίτερη προσοχή να μη δημιουργούνται latches κατά τη σύνθεση κάνοντας reset όλα τα στοιχεία μνήμης (flip-flops / registers) στο σχέδιο σας ή φροντίζοντας όλα τα μονοπάτια στον κώδικα συμπεριφοράς σας (behavioral process) να θέτουν τιμές σε αυτά.

- Μπορείτε αν θέλετε να φτιάξετε ξεχωριστή μονάδα - οντότητα **Demux**, όμως όπως θα διαπιστώσετε θα χρειαστεί λίγο περισσότερη προσπάθεια για να περνάτε τα δεδομένα εισόδου και εξόδου προς / από αυτές, με δεδομένο πως έχουν διαύλους που είναι παραμετρικοί και ο synthesizer του Vivado δεν υποστηρίζει ακόμα πλήρως το πρότυπο 2008 της VHDL που επιτρέπει τις δομές που απαιτούνται για να επιτευχθεί αυτό με ευκολία (σε νεότερες εκδόσεις πιθανώς να υπάρχει υποστήριξη).

- Στην αναφορά [1] αναφέρονται προτεινόμενες περιγραφές για πολλές λογικές μονάδες, ώστε αυτές να λειτουργήσουν σωστά και κατά τη διάρκεια της προσομοίωσης του σχεδίου σας αλλά και μέσα στο ίδιο το FPGA όταν το προγραμματίσετε με το σχέδιο σας. Στόχος είναι με την περιγραφή που θα προτείνετε να επιλέξει ο synthesizer τις καλύτερες δομές για την υλοποίηση του σχεδίου σας (design optimization).

- Όλες οι είσοδοι και οι έξοδοι του κυκλώματος που θα υλοποιήσετε θα πρέπει να αντιστοιχηθούν σε κατάλληλους ακροδέκτες του FPGA, ώστε το κύκλωμα να λειτουργήσει σωστά πάνω στην κάρτα. Η αντιστοίχιση αυτή γίνεται με τη χρήση ενός

αρχείου ‘xdc’. Στο αρχείο αυτό αναφέρονται όλα τα σήματα που υπάρχουν στο entity του σχεδίου που υλοποιούμε και κάθε ένα από αυτά αντιστοιχείται σε έναν ακροδέκτη του FPGA. Στο eClass στο αρχείο ‘example.xdc’ (φάκελος ‘Εργαστήριο’) μπορείτε να βρείτε ένα παράδειγμα συγγραφής αρχείου ‘xdc’ το οποίο μπορείτε να διαμορφώσετε για την υλοποίηση της παρούσας άσκησης. Τα ονόματα των σημάτων θα πρέπει να ταυτίζονται με αυτά των σημάτων που θα χρησιμοποιήσετε στο entity του σχεδίου σας. Το αρχείο ‘xdc’ θα πρέπει να το εισάγετε στο project που δημιουργήσατε στο Vivado.

- Μπορείτε να προδιαγράψετε όπως θέλετε ότι επιπλέον χρειάζεστε και δεν έχει προδιαγραφεί επακριβώς στην εκφώνηση της άσκησης. Όλες οι σχεδιαστικές επιλογές σας θα πρέπει να περιγραφούν και να δικαιολογηθούν στην έκθεσή σας.

Παραδοτέα

Το ορόσημο αυτό θα διαρκέσει 4 εργαστήρια και η ολοκλήρωση - εξέταση του θα γίνει την **Τετάρτη 26/4/2023**. Μία εβδομάδα μετά την εξέταση, δηλαδή την **Τετάρτη 3/5/2023**, θα πρέπει να παραδώσετε στο E-class συνοπτικό κείμενο σε μορφή τεχνικής έκθεσης (έγγραφα .doc μαζί με .pdf ή μόνο το .pdf) που να περιλαμβάνει τουλάχιστον τις παρακάτω ενότητες (sections) - όπου χρειάζεται για κάθε φίλτρο A, B:

- Συνοπτική παρουσίαση των μονάδων που υλοποιήσατε. Σχεδιαστικές επιλογές. Σχηματικά (schematics) ανώτερου επιπέδου που δείχνουν τα I/O ports για κάθε μονάδα και για όλο το σχέδιο συνολικά.
- Πόροι που δεσμεύονται από τη σύνθεση και την υλοποίηση για τα διαφορετικά N αλλά και D ή W (bonus). Σχολιασμός των αποτελεσμάτων της σύνθεσης και υλοποίησης μαζί με σχετικά διαγράμματα.
- Μέγιστη συχνότητα που μπορεί να επιτύχει το εργαλείο για τα διαφορετικά N αλλά και D ή W (bonus), λαμβάνοντας υπόψη την τιμή WNS (Worst Negative Slack). Θυμίζουμε πως Max F = 1 / (Clock Period - WNS). Θα πρέπει επίσης σε αυτή την ενότητα να παρουσιάσετε το κρίσιμο μονοπάτι (critical path) που αναφέρει το timing report που παράγει το εργαλείο και να εξηγήσετε (όσο μπορείτε) τι μας δείχνει αυτό και ποιές μονάδες και σήματα είναι το δημιουργούν στο σχέδιο σας. Σχολιασμός των αποτελεσμάτων μαζί με σχετικά διαγράμματα.

- Τεκμηρίωση ελέγχου ορθής λειτουργίας της κάθε μονάδας (με διαφορετικά σενάρια συμπεριλαμβανομένων και ακραίων περιπτώσεων). Ενδεικτικά διαγράμματα χρονισμού (timing diagrams) από προσομοιώσεις συμπεριφοράς σε ευανάγνωστη μορφή. Συζήτηση των αποτελεσμάτων.
- Σύνοψη ευρημάτων, παρατηρήσεις, δυσκολίες που παρουσιάστηκαν και πως τις αντιμετωπίσατε. Σε περίπτωση που δεν υλοποιούνται πλήρως οι προδιαγραφές να σχολιάζονται οι τυχόν αποκλίσεις.
- Τυχόν αναφορές σε βιβλία, εγχειρίδια κτλ που χρησιμοποιήθηκαν.

Για λόγους οικονομίας χώρου να μην συμπεριληφθεί ο κώδικας VHDL στην έκθεση.

Όλα τα παραδοτέα θα πρέπει να περιληφθούν σε συμπιεσμένο φάκελο με το όνομα και τον αριθμό μητρώου σας το οποίο θα παραδώσετε μαζί με την έκθεση. Να υπάρχει ξεχωριστός υπο-φάκελος που να περιλαμβάνει τον κώδικα VHDL επαρκώς σχολιασμένο, καθώς και τα αρχεία με τα οποία προσομοιώσατε το σχέδιό σας. Επίσης να υπάρχει η υλοποίηση που κατεβαίνει στο FPGA board (bitstream). Αν θέλετε μπορείτε να μην ανεβάσετε το project του Vivado αλλά μόνο τα αρχεία κώδικα σας, το bitstream και το report.

Αξιολόγηση

Οι εργαστηριακές ασκήσεις είναι υποχρεωτικές. Κατά την αξιολόγηση, ιδιαίτερη έμφαση θα δοθεί στη ορθότητα της υλοποίησης, στη πληρότητα του ελέγχου της υλοποίησης και της παρουσίασης των αποτελεσμάτων. Είναι πιθανόν να κληθείτε σε προφορική εξέταση επί του σχεδίου σας μετά την παράδοσή του.

Προσοχή: *Καθυστερημένες εργασίες δεν θα γίνουν δεκτές ούτε θα βαθμολογηθούν. Κάθε φοιτητής οφείλει να εργάζεται μόνος του. Η ανταλλαγή απόψεων επί της σχεδίασης επιτρέπεται, όμως απαγορεύεται ρητά η ανταλλαγή σχεδίων, κώδικα και εκθέσεων. Αν υπάρξει υπόνοια αντιγραφής όλοι οι εμπλεκόμενοι θα μηδενιστούν στο εργαστήριο.*

Χρήσιμα Αρχεία

Στη σελίδα eclass του μαθήματος θα βρείτε χρήσιμα αρχεία και συνδέσμους για την εργασία σας στον κατάλογο **Υποστηρικτικό υλικό**

[1] Vivado Synthesis User Guide PDF (E-Class -> Υλικό μαθήματος -> Υποστηρικτικό Υλικό)