Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

ECE333 - Εργαστήριο Ψηφιακών Συστημάτων

Εργαστηριακή Εργασία 4η

Περιεχόμενα

1.Τίτλος

Όνομα: Τσελεπή Ελένη

ΑΕΜ:03272

Ημερομηνία: 11/12– 22/01

Τίτλος εργασίας: Εργαστηριακή Εργασία 4η, Ψηφιακή Διαμόρφωση Σημάτων Ήχου

2.Περίληψη

Το παρακάτω κείμενο αποτελεί την εργαστηριακή αναφορά για την 4η εργαστηριακή εργασία όπου στόχος της αναφοράς είναι η σαφής περιγραφή της διαδικασίας σχεδίασης, επαλήθευσης, δοκιμής και τελικής υλοποίησης του κυκλώματος. Παρακάτω αναφέρονται αναλυτικά όλα τα μέρη υλοποίησης της εργασίας καθώς και στιγμιότυπα από τις κυματομορφές στο vivado.

3.Εισαγωγή

Ο στόχος της 4ης εργαστηριακής εργασίας ήταν η υλοποίηση ενός οδηγού **καταγραφής** και **αναπαραγωγής** ήχου χρησιμοποιώντας Ψηφιακή Διαμόρφωση. Ο οδηγός μετατρέπει ένα αναλογικό σήμα από το πεδίο του συνεχούς χρόνου σε ψηφιακό σήμα διακριτών τιμών για την καταγραφή ήχου και για την αναπαραγωγή ήχου ακολουθούμε την αντιστροφή διαδικασία . Τις μετατροπές από και προς το συνεχές πεδίο του χρόνου τις αντιμετωπίζουμε σαν "μαύρο κουτί " και διαχειριζόμαστε μόνο ψηφιακά σήματα . Αναλυτικά τα βήματα που ακολουθήθηκαν για την υλοποίηση του οδηγού τα παραθέτω παρακάτω .

4.Μέρος Α - Υλοποίηση Οδηγού Μικροφώνου και Αποσειριοποίηση Δεδομένων

MMCM

Για την υλοποίηση του οδηγού χρειαζόμαστε δύο ρολόγια (1 με συχνότητα λειτουργίας στα 200 MHz και ένα στα 2MHz για την δειγματοληψία) και για να τα παράγω χρησιμοποίησα την μονάδα MMCM . Όπως και στην πρώτη εργασία για τον προγραμματισμό της MMCM χρησιμοποίησα τους τύπους στη σελίδα 72 του τεχνικού δελτίου όπου

- FVCO=FCLKIN * M/D , 600MHZ <= FVC0 <= 1600MHZ , FCLKIN=100MHZ
- FOUT=FCLKIN * M/(D*O) ,**FOUT=200MHZ**

Για την συχνότητα 200MHz επέλεξα M:CLKFBOUT_MULT_F = 6, D:DIVCLK_DIVIDE=1 και O: CLKOUT_DIVIDE =3 (το οποίο προκύπτει από τον δεύτερο τύπο που παρέθεσα παραπάνω) . Για τα CLKFBOUT_MULT_F και DIVCLK_DIVIDE επέλεξα τις προαναφερθείσες τιμές γιατί ήθελα να κρατήσω το Fvco = 600MHz διότι σύμφωνα με το τεχνικό δελτίο "*The goal is to make D and M values as small as possible while keeping fvco as high as possible.*"

Για την συχνότητα 2MHz χρησιμοποίησα την cascade λειτουργία για να πέτυχω αυτή την μικρή συχνότητα διότι από τον δεύτερο τύπο προκύπτει 2=100*6/(1*O)=> O=300 το οποίο δεν μπορεί να πάρει αυτή την τιμή λόγω τον περιορισμών (1<CLKOUT_DIVIDE< 128) . Στην σελίδα 76 του τεχνικού δελτίου αναφέρει "*The CLKOUT6 divider (counter) can be cascaded with the CLKOUT4 divider. This provides a capability to have an output divider that is larger than 128. CLKOUT6 feeds the input of the CLKOUT4 divider.*"

MMCM Counter Cascading

The CLKOUT6 divider (counter) can be cascaded with the CLKOUT4 divider. This provides a capability to have an output divider that is larger than 128. CLKOUT6 feeds the input of the CLKOUT4 divider. There is a static phase offset between the output of the cascaded divider and all other output dividers.

Συνδυάζοντας αυτά τα 2 ρολόγια μπορούμε να πέτυχουμε την συχνότητα 2MHz . Αφού το CLKOUT6 *"feeds the input of the CLKOUT4 divider"* μιας και το Ο για το CLKOUT6 έχει επιλεγεί στην τιμή 3 για να πέτυχουμε την τιμή 300 το CLKOUT_DIVIDE(4)=100. Το τελευταίο που χρειάζεται να κάνουμε είναι να ενεργοποιήσουμε την cascade λειτουργία βάζοντας την τιμή TRUE στο CLKOUT4_CASCADE.

Επομένως το CLKOUT6 είναι η έξοδος για το ρολόι με την συχνότητα στα 200MHz και το ρολόι CLKOUT4 είναι η έξοδος για το ρολόι με την συχνότητα στα 2MHz.

BRAM

Για την μνήμη BRAM χρησιμοποίησα την μνήμη BRAM_SINGLE_MACRO (Tools->Language_Templates->Verilog->Macro->Artix-7->RAM->Single_Port) με μέγεθος στα 36Kb . Η συγκεκριμένη μνήμη είναι μια TDP μνήμη δηλαδή διαβάζει και γραφεί και από τα 2 Port (ενώ η SDP διαβάζει από το port A και γράφει στο port B) και Write /Read Width όρισα στα 32 bits . Από τα 36.000 bits μόνο τα 32.768(2^15) είναι valid ενώ τα υπόλοιπα είναι parity bits . Διαβάζοντας το manual BRAM-Manual και όπως παραθέτω και στο στιγμιότυπο παρακάτω κατάλαβα πως η διεύθυνση πρέπει να είναι 10 bit γιατί 32.768 / 32 bit width = 1024 = 2^10 .

Table 1-13: **Port Aspect Ratio for RAMB36E1 (in TDP Mode)**

Port Data Width	Port Address Width	Depth	ADDR Bus	DI Bus DO Bus	DIP Bus DOP Bus
1	15	32,768	[14:0]	[0]	NA
2	14	16,384	[14:1]	[1:0]	NA
4	13	8,192	[14:2]	[3:0]	NA
9	12	4,096	[14:3]	[7:0]	[0]
18	11	2,048	[14:4]	[15:0]	[1:0]
36	10	1,024	[14:5]	[31:0]	[3:0]
1 (Cascade)	16	65536	[15:0]	[0]	NA

Στιγμιότυπο από το Manual Bram όπου βλέπουμε το address width για DI,DO=32bit

Για να γράψουμε στην μνήμη πρέπει το σήμα Write enable (WE) να είναι ενεργό. Σύμφωνα λοιπόν με το manual το we είναι Byte-Wide Write Enable και επιτρέπει να γράφονται μόνο 8 bit για να γραφτούν και τα 32 bit επιλέγουμε το we να είναι 4 bit (4*8=32) ώστε να γράφονται και τα 32 bit στη μνήμη .

Byte-Wide Write Enable

The byte-wide write enable feature of the block RAM allows writing eight-bit (one byte) portions of incoming data. There are four independent byte-wide write enable inputs to the RAMB36E1 true dual-port RAM. There are eight independent byte-wide write enable inputs to block RAM in simple dual-port mode (RAMB36E1 in SDP mode). Table 1-7 summarizes the byte-wide write enables for the 36Kb and 18Kb block RAM. Each byte-wide write enable is associated with one byte of input data and one parity bit. All byte-wide write enable inputs must be driven in all data width configurations. This feature is useful when using block RAM to interface with a microprocessor. Byte-wide write enable is not available in the dual-clock FIFO or ECC mode. Byte-wide write enable is further described in the Additional RAMB18E1 and RAMB36E1 Primitive Design Considerations section. Figure 1-8 shows the byte-wide write-enable timing diagram for the RAMB36E1.

Στιγμιότυπο από το Manual Bram όπου εξηγείται η επιλογή του we ως 4bit

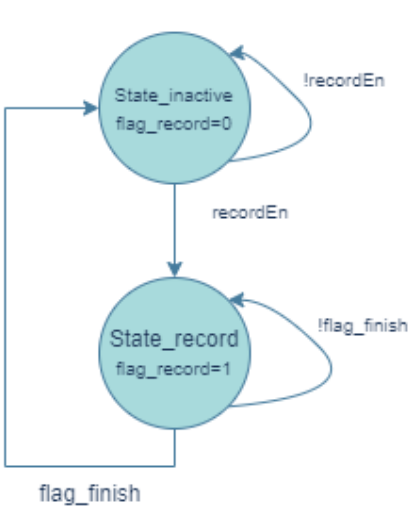
Τέλος, για την μνήμη το ρολόι έχει συχνότητα στα 200MHz και αποφάσισα το REGCE να το αφήσω σε σχόλια γιατί όπως διάβασα και στο manual είναι ένα σήμα enable για τους registers και δεν με απασχολούν για την εργασία .

Σχόλια:
Στον βοηθό που με εξέτασε εξήγησα πως ήθελα να χρησιμοποιήσω την μνήμη Verilog Primitive RAMB36E1 που προτείνεται όμως δεν μπορούσα, παρόλο που είχα διαβάσει το manual, να την κάνω να λειτουργήσει (έδειξα στον βοηθό πως δεν γινόταν ούτε read από την συγκεκριμένη μνήμη σε ένα project που είχα φτιάξει ειδικά για την μνήμη και παρόλο που το είδαμε και μαζί δεν μπορούσαμε να καταλάβουμε γιατί δεν λειτουργούσε) οπότε και επέλεξα να χρησιμοποιήσω την macro μνήμη που την είχα χρησιμοποιήσει και σε παλαιότερες εργασίες .

Οδηγός Καταγραφής Ήχου

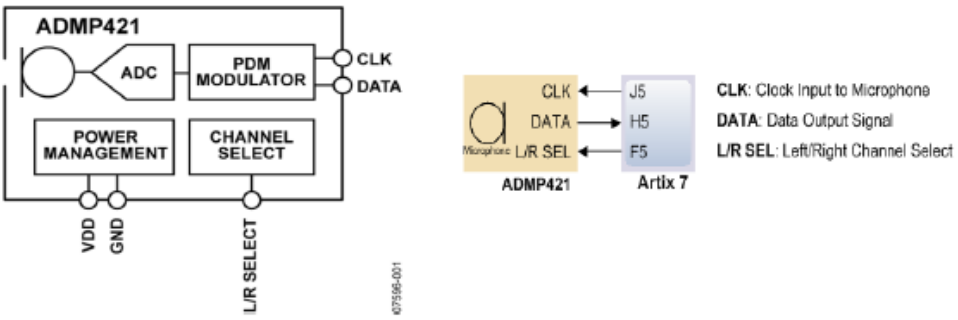
Για την καταγραφή ήχου, με το πάτημα ενός κουμπιού λαμβάνονται δεδομένα από το μικρόφωνο για ~0.015sec και αποθηκεύονται στη BRAM. Το μικρόφωνο ενεργοποιείται με συχνότητα των 2MHz και η δειγματοληψία από αυτό γίνεται μόνο στη θετική ακμή του ρολογιού. Τέλος, οι εγγραφές στη μνήμη γίνονται όλες παράλληλα με πλάτος 32 bits.

Για την καταγραφή ήχου υλοποίησα μια FSM με δυο καταστάσεις μια κατάσταση inactive όπου δεν γίνεται καταγραφή ήχου και μια κατάσταση που ενεργοποιείται με το πάτημα του κουμπιού για την καταγραφή του ήχου .



Η **FSM** για τον οδηγό καταγραφής ήχου με 2 καταστάσεις . Η συγκεκριμένη FSM είναι μια MOORE FSM καθώς η έξοδος (flag_record) καθορίζεται από την τρέχουσα κατάσταση. Αν έρθει το σήμα από το κουμπί (M17) recordEn η FSM πηγαίνει στην κατάσταση State_record για να γίνει καταγραφή του μηνύματος αλλιώς παραμένει στην κατάσταση inactive. Όταν έχει γεμίσει η μνήμη μετά από περίπου 0.015sec ένα σήμα flag_finish γίνεται ένα και η καταγραφή ήχου τελειώνει .

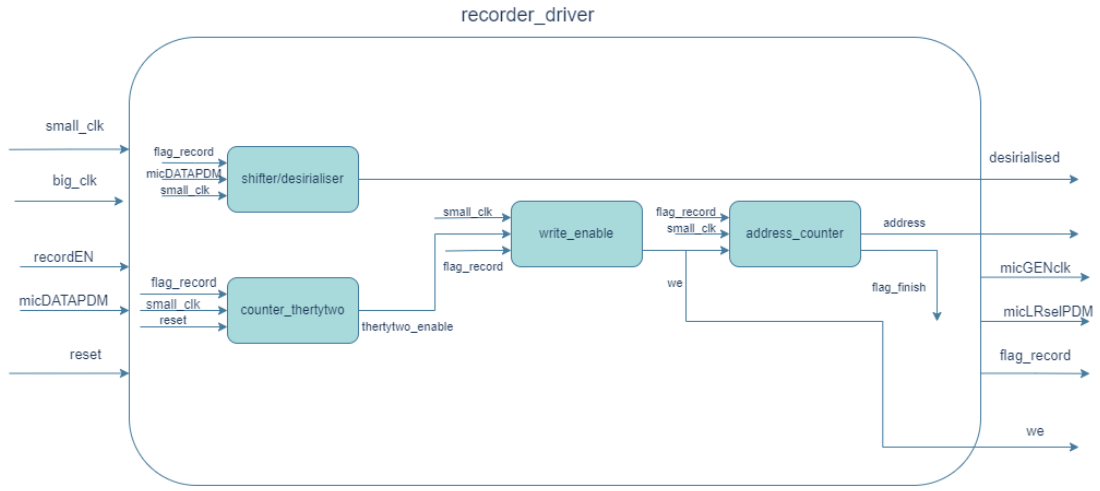
Πως λειτουργεί το μικρόφωνο ;



Σχήμα 2: Μικρόφωνο ADMP421

Το μικρόφωνο παίρνει ως είσοδο το ρολόι clk με συχνότητα 2 MHZ και ένα σήμα micLRselPDM το οποίο έχει την τιμή 0 ώστε η δειγματοληψία να γίνεται στη **θετική** ακμή του ρολογιού . Το σήμα micDataPDM είναι τα δεδομένα από το μικρόφωνο . Το μικρόφωνο επικοινωνεί με τον επεξεργαστή της πλακέτας ανταλλάσοντας μόνο 1 δεδομένο (bit) τη φορά στην ταχύτητα των 2MHz. Σε αντιδιαστολή, η Block RAM έχει πιο γρήγορο και μεγαλύτερο κανάλι επικοινωνίας για ανάγνωση (read) και εγγραφή (write) δεδομένων. ΄Αρα, η μνήμη πρέπει να διαβάζει αποσειριοποιημένα (deserialised) τα δεδομένα από το μικρόφωνο, δηλαδή τα δεδομένα από σειριακά του 1 bit πρέπει να γίνουν παράλληλα των 16 ή 32 bit, ώστε να χρησιμοποιηθεί ορθά ολόκληρη η πόρτα “εγγραφής” (write) της μνήμης. Γι αυτό τον λόγο στην FSM για την καταγραφή έχω μια μονάδα (shifter.v) για την αποσειριοποίηση των δεδομένων . Συγκεκριμένα χρησιμοποιώντας έναν right shifter για να γίνεται shifting στο least significant bit όταν έχει ξεκινήσει η καταγραφή (flag_record=1) και με το ρολόι στα 2MHZ(ώστε να βάζω στον shifter ένα bit κάθε φορά που στέλνεται από το μικρόφωνο) κάνω αποσειριοποίηση τα 32 bit του μικροφώνου. Η αποσειριοποίηση των 32 bit τελειώνει μετά από 32 κύκλους ρολογιού των 2MHz όπου σηκώνεται και ένα σήμα enable (thirtytwo_enable). Στο module *write_enable* μόλις έχουν ολοκληρωθεί 32 κύκλοι ρολογιού των 2Mhz (thirtytwo_enable) και η FSM βρίσκεται στην καταγραφή το we γίνεται ίσο με 4’b1111 ώστε να γραφεί στη μνήμη η έξοδος του deserialiser. Για την διεύθυνση της μνήμης έχω έναν counter (address_counter) ο οποίος όπως προείπα είναι 10 bit αρχικοποιείται στην τιμή ‘d1023 ώστε την πρώτη φορά που έχουν ολοκληρωθεί 32 κύκλοι η διεύθυνση λόγω overflow να γίνεται 0 και αυξάνεται κατά 1 κάθε 32 κύκλους ρολογιού των 2 Mhz. Η καταγραφή τερματίζει όταν έχουν γεμίσει όλες οι θέσεις της μνήμης όταν δηλαδή η διεύθυνση έχει την τιμή 1023 και έχουν γραφεί τα τελευταία 32 bit(Το σήμα γίνεται flag_finish=1) . Αυτό συμβαίνει μετά από χρόνο ~0.016 sec γιατί κάθε δεδομένο έχει περίοδο 500ns (2MHz) άρα 32.768bit *500ns=0.016384ns.

Όλα αυτά που περίγραψα παρουσιάζονται και στο παρακάτω διάγραμμα ροής dataflow.

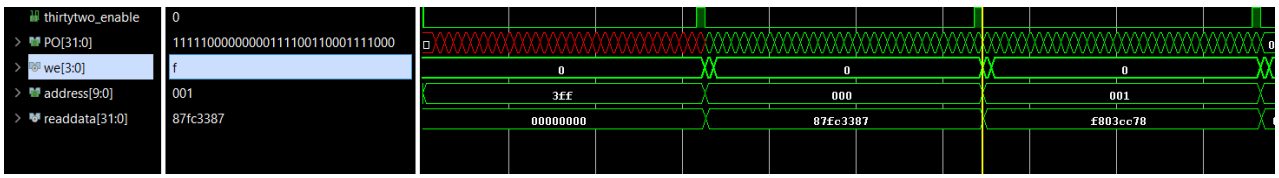


Στιγμιότυπο από το dataflow για την καταγραφή του ήχου

Τα αποτελέσματα από το testbench για το Α μέρος

Για τον έλεγχο του Α μέρους προσπάθησα στο testbench να προσομοιώσω την λειτουργία του μικροφώνου και ανά 500ns να αλλάζω το micDATAPDM . Το πρώτο δεδομένο που στέλνω είναι το 32'b 111_0000_111_00_11_0000_111111111_00001 ενώ το δεύτερο δεδομένο είναι 32'b000_1111_000_11_00_1111_000000000_11111 και ελέγχω την λειτουργία του κυκλώματος μου .

Στιγμιότυπο από τον έλεγχο για το Α μέρος. Όπως φαίνεται και στο στιγμιότυπο την στιγμή που στην έξοδο του deserialiser(PO) βρίσκεται ο αριθμός που έχω στείλει ανεστραμμένος λόγω του right shifting , το we=4'b1111 και το address μόλις έχει γίνει 0 .

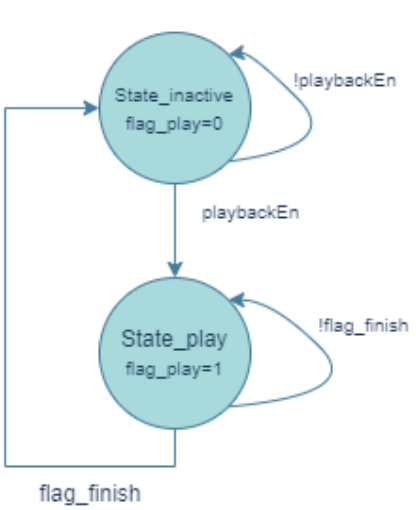


Στιγμιότυπο από την αποστολή του δεύτερου δεδομένου . Όταν στην έξοδο του deserialiser βρίσκεται το δεύτερο δεδομένο αναστραμμένο, το we έχει την τιμή 4'b1111 και το address μόλις έχει γίνει 1 . Όπως θα δείτε το σήμα readdata είναι το DO την μνήμης και όταν το address είναι στην τιμή 0 στη μνήμη βρίσκεται το δεδομένο που μόλις έγραψα (Το 87fc3387 είναι η hex μορφή του ανεστραμμένου αριθμού) και αντίστοιχα για το δεύτερο δεδομένο έτσι μπορούσα να ελέγξω ότι τα δεδομένα περνούσαν στη μνήμη .

5.Μέρος Β - Υλοποίηση Οδηγού Αναπαραγωγής Ήχου και Σειριοποίηση Δεδομένων

Για το Β μέρος της εργασίας υλοποίησα τον Οδηγό Αναπαραγωγής ΄Ηχου όπου με το πάτημα ενός καινούριου κουμπιού (M18) διαβάζει τα δεδομένα από τη BRAM ώστε να τα αποστείλει στην αντίστοιχη έξοδο οδηγώντας το Low Pass Filter Sallen-Key Butterworth για διάρκεια ~5sec επαναλαμβάνοντας το μήνυμα διάρκειας 0.015 sec.

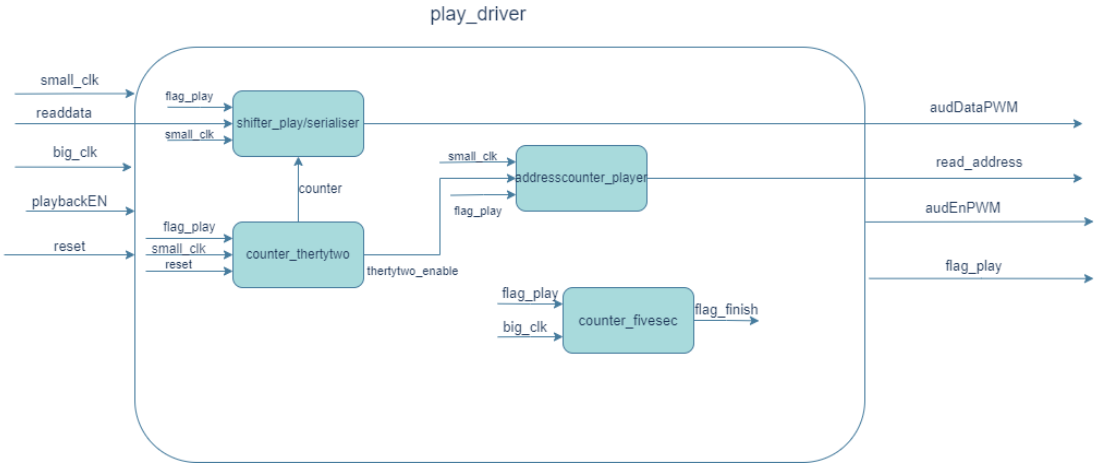
Για την αναπαραγωγή του ήχου όπως και για την εγγραφή υλοποίησα μια FSM με δυο καταστάσεις μια κατάσταση inactive όπου δεν γίνεται αναπαραγωγή ήχου και μια κατάσταση που ενεργοποιείται με το πάτημα του κουμπιού για την αναπαραγωγή του μηνύματος .



Η **FSM** για τον οδηγό αναπαραγωγής ήχου με 2 καταστάσεις . Η συγκεκριμένη FSM είναι μια MOORE FSM καθώς η έξοδος (flag_play) καθορίζεται από την τρέχουσα κατάσταση. Αν έρθει το σήμα από το κουμπί (M18) playbackEn η FSM πηγαίνει στην κατάσταση State_play για να γίνει αναπαραγωγή του μηνύματος αλλιώς παραμένει στην κατάσταση inactive. Όταν περάσουν τα 5 sec (με την χρήση ενός μετρητή) ένα σήμα flag_finish γίνεται ένα και η αναπαραγωγή τελειώνει .

Για την αναπαραγωγή ήχου λοιπόν, δημιούργησα έναν serialiser(shifter_play) ώστε τα 32 bit που διαβάζονται (read) από την μνήμη να γίνονται σειριακά του 1 bit καθώς όπως προείπα και στο προηγούμενο μέρος η έξοδος ήχου επικοινωνεί με τον επεξεργαστή της πλακέτας ανταλλάσοντας μόνο 1 δεδομένο τη φορά στην ταχύτητα των 2MHz. Γι αυτό τον λόγο ο serialiser λειτουργεί με ένα ρολόι στα 2 Mhz , έχει ως είσοδο το DO της μνήμης στα 32 bit και βγάζει σε κάθε κύκλο ρολογιού 1 bit . Ο serialiser «φορτώνει» σε έναν tmp πίνακα την είσοδο (το DO της μνήμης) όταν ο counter που μετράει 32 κύκλους ρολογιού των 2 Mhz είναι στο 0 και έχει ξεκινήσει η κατάσταση αναπαραγωγής ήχου. Στη συνέχεια σε κάθε κύκλο ρολογιού το αποτέλεσμα γίνεται right shifting και στο output του serialiser υπάρχει το least significant ψηφίο του tmp πίνακα (tmp[0]). Έχοντας αυτούς τους δυο deserialiser-serialiser καταφέρνουμε τα bit του audDataPWM που είναι και η έξοδος του serialiser να έχει τα δεδομένα όπως ακριβώς τα στείλαμε . Για την διεύθυνση της μνήμης έχω και σε αυτό το μέρος έναν μετρητή (addresscounter_player) ο οποίος αυξάνεται κάθε 32 κύκλους ρολογιού των 2 Mhz (thirtytwo_enable) κατά 1 ξεκινώντας από το 0 έως το 1023 . Τέλος για την αναπαραγωγή του μηνύματος έχω έναν μετρητή τον counter_fivesec ο οποίος με το ρολόι των 200Mhz μετράει 5sec (5sec/5ns=1*10^9

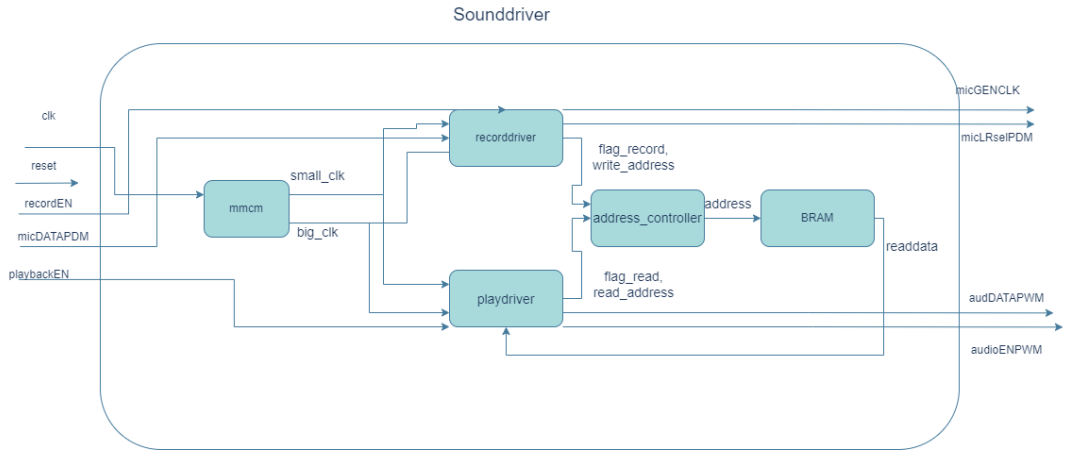
κύκλοι) όταν έχουν περάσει τα 5 sec ένα σήμα flag_finish γίνεται 1 και η αναπαραγωγή τελειώνει. Στα 5 αυτά sec η διεύθυνση της μνήμης πηγαίνει από το 0-1023 συνεχόμενα ώστε να επαναλαμβάνεται το μήνυμα . Το σήμα audEnPWM το κάνω assign με το flag_play το σήμα που δείχνει ότι είμαι στην κατάσταση αναπαραγωγής .
Όλα αυτά που περίγραψα παρουσιάζονται και στο παρακάτω διάγραμμα ροής dataflow.



Στιγμιότυπο από το dataflow για την αναπαραγωγή του ήχου .

Για να ενωθούν αυτά τα 2 μέρη της καταγραφής και της αναπαραγωγής ήχου τα ένωσα σε ένα **top level module** (sounddriver) στο οποίο υπάρχουν και δυο επιπλέον module ένα για την μνήμη BRAM που περιέγραψα στο Α μέρος και ένα module (address_controller) το οποίο ελέγχει την διεύθυνση που πηγαίνει στην μνήμη . Συγκεκριμένα, στο συγκεκριμένο module ελέγχεται με ένα απλό combinational always block αν ο οδηγός βρίσκεται στην κατάσταση καταγραφής (flag_record=1) ή στην κατάσταση αναπαραγωγής(flag_play=1) ώστε και η διεύθυνση που πηγαίνει στην μνήμη να είναι είτε από τον μετρητή οπού υπάρχει στο record module είτε από την διεύθυνση που είναι μετρητής στο playback module .

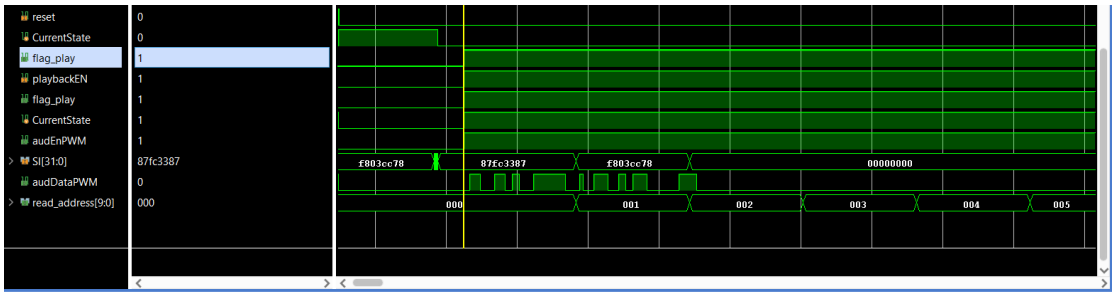
<u>Παραδοχή</u>
Ο οδηγός κάνει πρώτα εγγραφή του μηνύματος και μόλις τελειώσει ξεκινάει η αναπαραγωγή .



Στιγμιότυπο από το dataflow για το συνολικό top level module .

Τα αποτελέσματα από το testbench για το Β μέρος

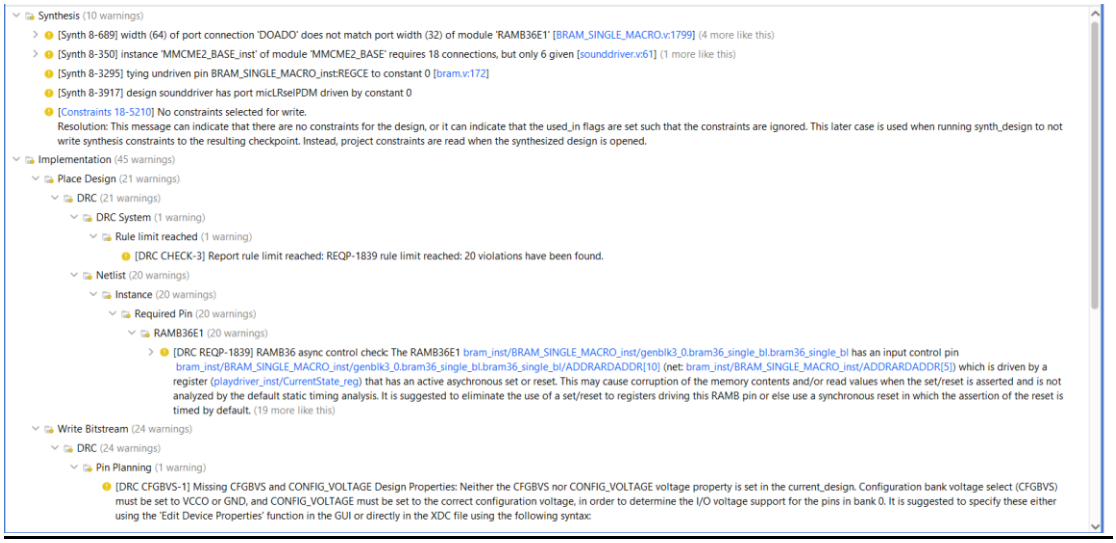
Για τον έλεγχο του Β μέρους έλεγξα αν τα δεδομένα που έστελνα στο Α μέρος θα βρίσκονται ανά 1 bit στην έξοδο audDataPWM. Επειδή ο έλεγχος είναι ενδεικτικός και για να μην εισάγω πολλές καθυστερήσεις έκανα την καταγραφή του μηνύματος να σταματάει μόλις έχουν γραφτεί τα 2 μηνύματα στη μνήμη . Έτσι στη συνέχεια έκανα το σήμα playbackEN ίσο με 1 ώστε να ξεκινήσει η αναπαραγωγή .



Στιγμιότυπο από την αναπαραγωγή του ήχου . Όπως θα δείτε το πρώτο CurrentState γίνεται 0 που είναι inactive για την εγγραφή ήχου και στη συνέχεια ξεκινάει η αναπαραγωγή όταν το playbackEn γίνεται 1 . Όταν ξεκινάει η αναπαραγωγή η διεύθυνση βρίσκεται στο 0 και το audDataPWM έχει τα bit του πρώτου μηνύματος . Μετά από 32 κύκλους ρολογιού η διεύθυνση γίνεται 1 και μετά από έναν κύκλο ρολογιού των 2 Mhz βγαίνει το δεύτερο δεδομένο στην έξοδο audDataPWM.

Το συγκεκριμένο στιγμιότυπο είναι μετά από 16ms όπου η διεύθυνση έχει περάσει τις 1024 τιμές και ξανά ξεκινάει από το 0 . Όπως θα δείτε όταν η διεύθυνση είναι πάλι 0 και 1 στο audDataPWM έχουμε τα bit που περιμέναμε όπως ακριβώς είναι και στο 1^{ο} στιγμιότυπο .

Warnings



Τα συγκεκριμένα warnings δεν με ανησυχήσαν διότι το DO από την εκφώνηση πρέπει να είναι 32 bit , το REGCE όπως εξήγησα και στο κεφάλαιο με την μνήμη επέλεξα να

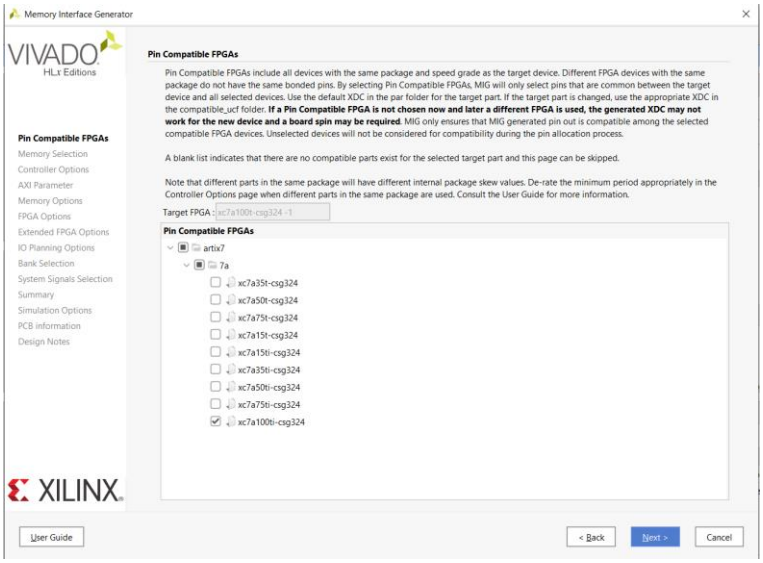
είναι σε σχόλια και τέλος τα warnings για το implementation απλά προτείνουν το reset να είναι σύγχρονο κάτι όμως που δεν γίνεται γιατί σε όλες μας τις εργασίες το reset είναι ασύγχρονο . Γι αυτό τον λόγο τα συγκεκριμένα warning δεν τα θεώρησα σημαντικά .

Προγραμματισμός πλακέτας

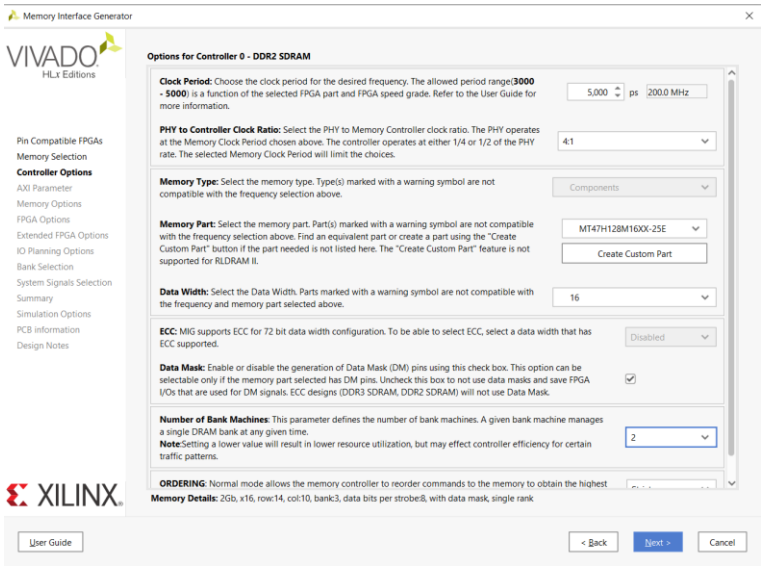
Για τον προγραμματισμό της πλακέτας χρειάστηκε μόνο ένας προγραμματισμός της πλακέτας μιας και συνέδεσα αρχικά τα ακουστικά μου στην ειδική θύρα της πλακέτας και κάνοντας αρχικά καταγραφή και μετά αναπαραγωγή πατώντας το κουμπί M18 άκουσα για 5 sec έναν ήχο (σαν βαβούρα) . Στην εξέταση μου βάλαμε τα ηχεία στην πλακέτα και ακούσαμε και πάλι τον ίδιο ήχο για διάρκεια περίπου 5 sec . Στην εξέταση να σημειώσω δεν χρησιμοποίησα τα antibounce αρχεία για τα κουμπιά που βρίσκονται στο project της εργασίας γιατί ξέχασα να τα συνδέσω με τα σήματα recordEn, playbackEn κατά τον προγραμματισμό της πλακέτας όμως τα παραθέτω γιατί θα μπορούσαν τα κουμπιά να έχουν antibounce .

6. Μέρος Γ - Υλοποίηση Ελεγκτή Μνήμης DDR2

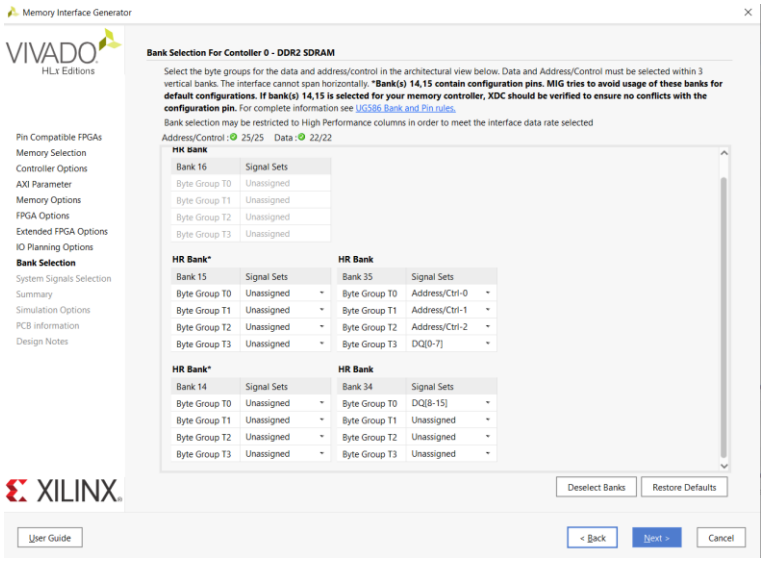
Για το Γ μέρος της εργασίας ακολούθησα τα βήματα εγκατάστασης της DDR2 που υπάρχουν στο ειδικό εγχειρίδιο DDR2 . Συγκεκριμένα, σας παραθέτω παρακάτω κάποια στιγμιότυπα από την διαδικασία εγκατάστασης .



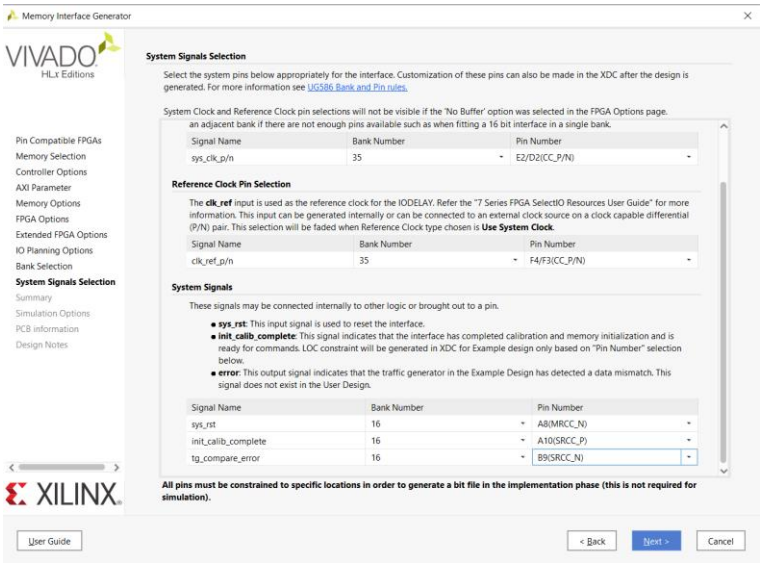
Στιγμιότυπο από την επιλογή Pins .Επέλεξα το τελευταίο γιατί η ονομασία της πλακέτας μας είναι Nexys A7-100T.



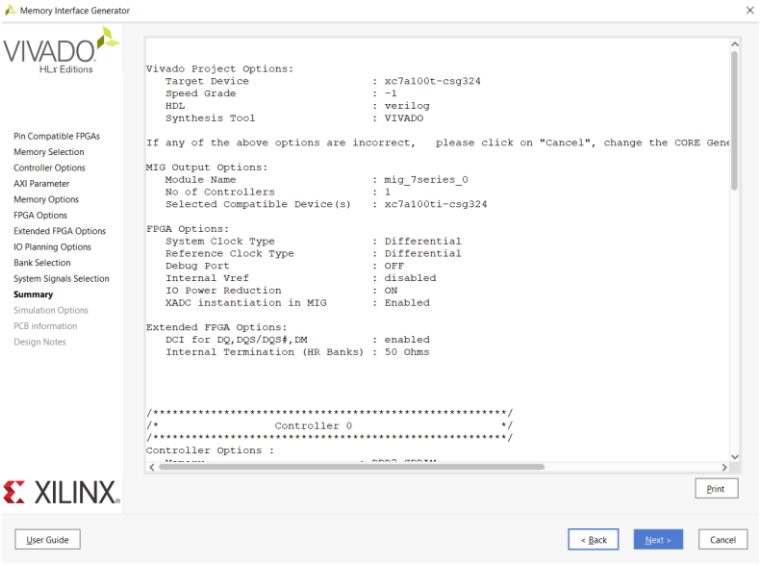
Στιγμιότυπο από την επιλογή των controllers. Όπως φαίνεται έχω επιλέξει ως περίοδο ρολογιού τα 5000ps ώστε να λειτουργεί στα 200Mhz και επίσης έχω ορίσει το Data Width στα 16bit όπως ορίζει η εκφώνηση . Τέλος , για τα Banks έχω ορίσει τον μικρότερο δυνατό αριθμό στα 2 banks.



Στιγμιότυπο από την επιλογή των Banks . Συγκεκριμένα αφού έχω επιλέξει προηγουμένως 2 Banks επέλεξα το Bank 35 και 34 .



Στιγμιότυπο από την επιλογή των Pins Number για τα banks .



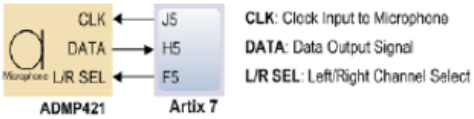
Τελικό Summary των επίλογων το οποίο παραθέτω μαζί με την εργασία .

Σχόλια

Να σημειωθεί πως τα βήματα που δεν βρίσκονται σε στιγμιότυπο είναι επειδή δεν έκανα κάποια αλλαγή από τις προεπιλεγμένες τιμές . Με το συγκεκριμένο μέρος της εργασίας δεν μπορούσα να προχωρήσω γιατί παρόλο που ακολουθούσα όλα τα βήματα που σας εξήγησα παρά πάνω στο τέλος της διαδικασίας και ενώ πατούσα την επιλογή generate δεν μου άνοιγε νέο παράθυρο ώστε να συνεχίσω με την εγκατάσταση της DDR2 κάτι που υπέδειξα και στον βοηθό του εργαστήριου κατά την εξέταση μου .

Επιλογή Constraints

Για την επιλογή των constraints για το .xdc αρχείο διάβασα το manual της Nexys nexys-a7 και το αρχείο Master XDC File για την πλακέτα μας την Nexys A7-100T. Συγκεκριμένα, για την εγγραφή ήχου επέλεξα τα pins που φαίνονται στο στιγμιότυπο βάζοντάς στο pin J5 το micGenCLK , στο H5 το micDataPDM και τέλος στο F5 το micLRselPDM.



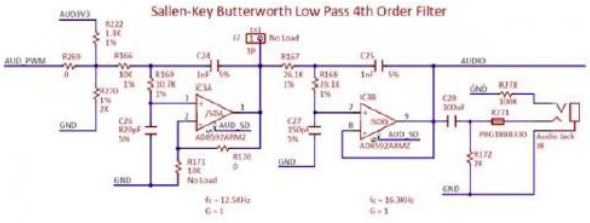
Στιγμιότυπο από το manual για την εγγραφή ήχου

```
##Buttons
#set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { BTNC }]; #IO_L9P_T1_DQS_14 Sch=btnc
#set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { BTNU }]; #IO_L4N_T0_D05_14 Sch=btnu
#set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports { BTNL }]; #IO_L12P_T1_MRCC_14 Sch=btnl
#set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports { BTNR }]; #IO_L10N_T1_D15_14 Sch=btnr
#set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports { BTND }]; #IO_L9N_T1_DQS_D13_14 Sch=btnd
```

Στιγμιότυπο από το Master XDC File το οποίο παρουσιάζει τις επιλογές στα buttons το οποίο με βοήθησε να επιλέξω τα pins για τα κουμπιά reset (N17), recordEn(M17), playbackEn(M18).

Για την επιλογή των pins για την αναπαραγωγή του ήχου συμβουλεύτηκα το manual αλλά και το Master XDC File.

The on-board audio jack (J8) is driven by a Sallen-Key Butterworth Low-pass 4th Order Filter that provides mono audio output. The circuit of the low-pass filter is shown in Figure 15.1. The input of the filter (AUD_PWM) is connected to the FPGA pin A11. A digital input will typically be a pulse-width modulated (PWM) or pulse density modulated (PDM) open-drain signal produced by the FPGA. The signal needs to be driven low for logic '0' and left in high-impedance for logic '1'. An on-board pull-up resistor to a clean analog 3.3V rail will establish the proper voltage for logic '1'. The low-pass filter on the input will act as a reconstruction filter to convert the pulse-width modulated digital signal into an analog voltage on the audio jack output.



Στιγμιότυπο από το manual της Nexys A7-100T για την αναπαραγωγή ήχου.

```
##PWM Audio Amplifier
#set_property -dict { PACKAGE_PIN A11   IOSTANDARD LVCMOS33 } [get_ports { AUD_PWM }]; #IO_L4N_T0_15 Sch=aud_pwm
#set_property -dict { PACKAGE_PIN D12   IOSTANDARD LVCMOS33 } [get_ports { AUD_SD }]; #IO_L6P_T0_15 Sch=aud_sd
```

Στιγμιότυπο από το XDC File για την επιλογή των pins για την αναπαραγωγή ήχου.

Τα παραπάνω στιγμιότυπα που παρέθεσα με βοήθησαν να συνδέσω τα σήματα audDataPWM στο pin A11 και το audEnPWM στο pin D12 .

7. Συμπεράσματα

Η τέταρτη εργαστηριακή εργασία ήταν μια απαιτητική εργασία καθώς χρειάστηκαν ώρες ενασχόλησης με τα ειδικά εγχειρίδια (manual) τόσο για την λειτουργία της μονάδας MMCM όσο και για την λειτουργία του μικροφώνου και της DDR . Με την συγκεκριμένη εργασία έμαθα να ψάχνω τις πληροφορίες που χρειάζομαι στα manual αλλά έμαθα και πως λειτουργεί το μικρόφωνο και πως γίνεται αναπαραγωγή ήχου με την πλακέτα Nexys A7-100T.