Technical
University
of Crete

Αναδιατασσόμενα Ψηφιακά Συστήματα

HPY591

Milestone 1

Σπυριδάκης Χρήστος 2014030022
Βίττης Βασίλειος 2015030164

1.Εισαγωγή

Σκοπός του συγκεκριμένου Milestone είναι η επαφή με το εργαλείο Xilinx Vivado, το υπόβαθρο πίσω από την αναδιατασσόμενη λογική καθώς επίσης και με το AXI4 Stream Protocol. Πρώτο βήμα πριν την υλοποίηση του κυρίου μέρους του Project ήταν η κατανόηση των διαφόρων τύπων handshake που υπάρχουν στο protocol και χρησιμοποιούνται μεταξύ άλλων και στο DMAe με το My IP στην υλοποίηση μας.
΄Υστερα από την μελέτη μας στο AXI4 Stream protocol καταλήξαμε ότι σκοπός του είναι η επιτυχής μεταφορά και ο χρονισμός δεδομένων μεταξύ των λογικών μονάδων που το χρησιμοποιούν. Αυτό εξασφαλίζεται από παραπάνω πληροφορία που παρέχεται κατά την διαδικασία αυτή. Όπως βλέπουμε παρακάτω υπάρχουν τα σήματα TVALID και TREADY με βάση τα οποία η μεταφορά δεδομένων λειτουργεί επιτυχώς.

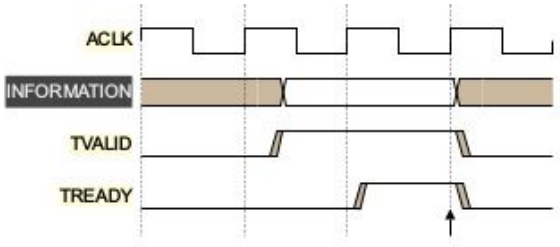


Figure 2-1 TVALID before TREADY handshake

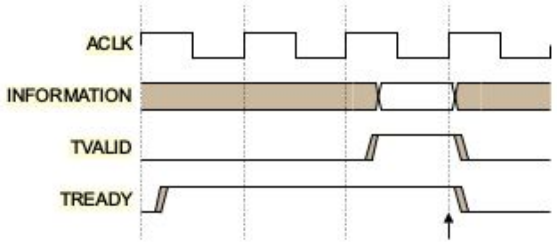


Figure 2-2 TREADY before TVALID handshake

Ο κοινός τόπος των παραπάνω τριών διαγραμμάτων είναι ότι η μεταφορά δεδομένων από τον εκάστοτε master και slave είναι επιτυχής μόνο όταν τα σήματα TVALID και TREADY ενεργοποιηθούν ταυτόχρονα. Αυτή είναι αναγκαία συνθήκη για να γνωρίζουμε ότι στον επόμενο κύκλο τα δεδομένα τα οποία λαμβάνονται στον slave από τον master είναι αποδεκτά.

Η αποθήκευση της πληροφορίας από τον DMA engine προς το Mp IP γίνεται μέσω της υλοποίησης FIFO που όπως φαίνεται και παρακάτω πληρεί κάποιες προϋποθέσεις.

2.Σχεδίαση/Υλοποίηση

Πρώτα μελετάμε τις προδιαγραφές και την θεωρητική προ - επεξεργασία της λογικής της FIFO.

2.1 FIFO

Timing diagram 2.1 - FIFO

Στο παραπάνω διάγραμμα χρονισμού καθορίζουμε πως θα λειτουργεί η FIFO την οποία θα υλοποιήσουμε. Η λογική της FIFO είναι ήδη γνωστή και από προηγούμενα μαθήματα. Η βασική ιδέα της είναι διακριτή ακόμα και από το ίδιο το όνομα της - first in first out. Αυτό σημαίνει ότι τα πρώτα δεδομένα που θα γραφτούν θα πρέπει να είναι τα ίδια δεδομένα που θα διαβαστούν. Η FIFO που υλοποιήσαμε όπως μας υποδεικνύει και η εκφώνηση της άσκησης είναι 64 θέσεων με 32 bits πλάτος η κάθε μια. Σχετικά με το παραπάνω διάγραμμα χρονισμού ως προδιάγραφες ορίζουμε ότι η λειτουργικότητα της FIFO ξεκινάει και τελειώνει στα σημεία που το reset σήμα είναι ανενεργό (aresetn = ‘1’). Επίσης ως προδιαγραφή επιτρέπουμε την παράλληλη εγγραφή και ανάγνωση χωρίς να υπάρχουν για το καθένα ξεχωριστά καθυστερήσεις κύκλων, άρα με το πέρας της εγγραφής δεδομένων στην FIFO είναι διαθέσιμα προς ανάγνωση τα δεδομένα (κάτι τέτοιο γίνεται αντιληπτό στο κομμάτι των κυματομορφών). Ακόμα, ορίζουμε να δίνονται τα δεδομένα προς εγγραφή σύγχρονα μαζί με το αντίστοιχο σήμα έναρξης εγγραφής (δηλαδή data_in και wen στον ίδιο κύκλο). Όπως επίσης έχουμε ορίσει ότι με το σήμα έναρξης διαβάσματος τα δεδομένα βγαίνουν από την FIFO σύγχρονα στον ίδιο κύκλο (δηλαδή data_out και ren στον ίδιο κύκλο). Επιπλέον επειδή

το περιβάλλον, στο οποίο υλοποιείται η FIFO, περιλαμβάνει και την ανάγκη λογικής σχεδίασης ελέγχου αυτής, δεν μπορούσαν να λείπουν κατά την σχεδίαση της και σήματα ένδειξης της κατάστασης της. Άρα ορίσαμε τα σήματα empty και full για να δείχνουμε ότι η FIFO είναι άδεια και γεμάτη αντίστοιχα. Η ένδειξη της γεμάτης (με αντίστοιχο τρόπο και όταν άδειας) λίστας, από την FIFO προς τον slave γίνεται ταυτόχρονα με την εγγραφή του τελευταίου δεδομένου σε αυτήν. Αυτή η πληροφορία είναι απαραίτητη για την επιτυχημένη επικοινωνία του IP μας με τον DMA engine. Τέλος, σχεδιάσαμε την FIFO ως ένα array και ορίσαμε δυο pointer, ένας για την εγγραφή (write_pointer) και ένας για την διαδικασία ανάγνωσης (read_pointer) των οποίων το εύρος τιμών είναι από το 0 μέχρι το 64 (65 θέσεων δηλαδή με μία κενή θέση σε περίπτωση που είναι γεμάτη). Ως σχεδίαση ορίσαμε ότι η FIFO μας είναι post increment, δηλαδή ο write_pointer (όμοια και ο read pointers) δείχνει στην θέση που όταν έρθουν τα δεδομένα θα γραφτούν.

Οι καταστάσεις της FIFO με βάση τις οποίες ελέγχεται η εγγραφή και η ανάγνωση της είναι η κάτωθι. Ως σημείωση προσθέτουμε ότι η μετάβαση στην κατάσταση Write/Read προϋποθέτει και την δυνατότητα αυτή, παραλείπεται δηλαδή ο έλεγχος full και empty αντίστοιχα. Ενώ όπως είπαμε και παραπάνω, η μετάβαση από την μία κατάσταση στην άλλη μπορεί να γίνει από τον ένα στον άλλο κύκλο χωρίς καθυστερήσεις.

Schematic 2.1 - FIFO’s FSMs

Συνεχίζοντας με bottom up λογική εξετάζουμε την λειτουργικότητα και ορίζουμε τις προδιαγραφές του Slave και Master module.

2.2 Slave

Timing diagram 2.2 - FIFO's Slave

Σχετικά με την δομή του παραπάνω διαγράμματος χρονισμού, αυτό χωρίζεται στο πάνω μέρος στο οποίο παρατηρούμε την διεπαφή του Slave module με το DMA (είναι ίδιο με το reference project καθώς χρησιμοποιούμε το AXI4 Stream protocol το οποίο χρειάζεται τα συγκεκριμένα σήματα) και στο κάτω μέρος, την διεπαφή του Slave Module με την FIFO. Επίσης, την χρονική διάρκεια που δεν έχει ήδη πραγματοποιηθεί κάποιο handshake έχουμε θεωρήσει τα δεδομένα ως σκουπίδια μαρκάροντας τα με πορτοκαλί.

Ως χρονικό παράθυρο ενδιαφέροντος θεωρούμε την χρονική διάρκεια που το reset είναι ανενεργό (aresetn='1') και το tvalid είναι ενεργό (tvalid='1'). Έτσι, έστω ότι τα δεδομένα που στείλει ο DMA Engine είναι ορθά, το slave module του MY IP θα επιτρέψει την μετάδοση πληροφορίας που θέλουν να έρθουν προς αυτόν μόνο όταν το σήμα tready του είναι ενεργό, το οποίο εξαρτάται με το αν η FIFO είναι γεμάτη. Άρα, έχοντας την ταυτόχρονη ενεργοποίηση των δύο προαναφερθέντων σημάτων μπορεί ο slave στον ίδιο κύκλο να μεταβιβάσει τα δεδομένα που λαμβάνει στην FIFO, και να ενεργοποιήσει την εγγραφή. Η διαδικασία εγγραφής δεδομένων από τον DMA στο MY IP γίνεται έως ότου είτε η FIFO να γεμίσει, full = '1' και άρα το slave θα παράγει σήμα tready ='0', είτε το tvalid του DMA να πάρει την τιμή '0'. Τέλος, ως παρατήρηση θέλουμε να προσθέσουμε ότι παρόλο που τα δεδομένα που έρχονται από τον DMA είναι valid, η υλοποίηση της FIFO τα θεωρεί ως σκουπίδια γιατί το σήμα full της είναι ενεργό που συνεπάγεται ότι το tready του slave είναι ανενεργό στο διάστημα αυτό, όπως επίσης και το wen.

2.3 Master

Timing diagram 2.3 - FIFO's Master

Με την ίδια λογική και το παραπάνω διάγραμμα χρονισμού χωρίζεται στο πάνω μέρος, στο οποίο παρατηρούμε την διεπαφή του Master module του IP μας με το DMAe και στο κάτω μέρος, την διεπαφή του Master Module με την FIFO. Επίσης, την χρονική διάρκεια που δεν έχουμε ενεργό valid έχουμε θέσει τα δεδομένα με σκουπίδια μαρκάροντας τα με πορτοκαλί.

Ως πρώτο χρονικό παράθυρο ενδιαφέροντος θεωρούμε την χρονική διάρκεια που το reset είναι ανενεργό (aresetn='1') και το tready είναι ενεργό (tready='1'). Έτσι, για να πραγματοποιηθεί ορθό handshake μεταξύ του MY IP και του DMA πρέπει τα σήματα tvalid και tready να είναι ταυτόχρονα ενεργοποιημένα με αποτέλεσμα να ενεργοποιηθεί η διαδικασία ανάγνωσης (ren = '1'). Η διαδικασία ανάγνωσης τερματίζει είτε όταν η FIFO έχει αδειάσει (empty = '1') που αναγκάζει το master module να θέσει το σήμα tvalid σε '0΄ είτε όταν το DMA Engine δεν είναι σε θέση να δεχτεί άλλα δεδομένα (tready = '0'). Ενδιαφέροντα σημεία στο παραπάνω διάγραμμα χρονισμού είναι τα σημεία ένωσης (a στο b , c στο d, e στο f) που υποδεικνύουν την άμεση σχέση μεταξύ του σήματος empty και tvalid καθώς όταν το πρώτο είναι ανενεργό (empty = '0') τοτε το δεύτερο είναι ενεργό (tvalid ='1') και αντίστροφα.

Με βάση τα παραπάνω οι διεπαφές των επιμέρους modules της MY IP λογικής γίνονται εμφανείς στους παρακάτω πίνακες. (όπου *C.P = Connection Point)

C.P.	Name	Length	Type	Info
	fifo_aresetn	1	Input	Global reset signal
SLAVE	fifo_w_aclk	1	Input	Write clock signal
	fifo_full	1	Output	FIFO is full
	fifo_wen	1	Input	Push to FIFO
	fifo_data_in	32	Input	Data to push in FIFO
MASTER	fifo_r_aclk	1	Input	Read clock signal
	fifo_empty	1	Output	FIFO is empty
	fifo_ren	1	Input	Pop from FIFO
	fifo_data_out	32	Output	Data that extracted from FIFO

Table 2.1 - FIFO

Note: Η FIFO που υλοποιήσαμε υποστηρίζει δύο διαφορετικά ρολόγια. Ένα για την εγγραφή και ένα για την ανάγνωση ώστε αν χρειαστεί να χρησιμοποιηθεί ως buffer όπως αναφέρθηκε στο μάθημα. Δοκιμάστηκε σε περιβάλλον όπου το ένα ρολόι να είναι ακριβώς διπλάσιο από το άλλο. Ο σχετικός έλεγχος υπάρχει στο folder του project αλλά δεν γίνεται περαιτέρω ανάλυση καθώς τουλάχιστον μέχρι αυτήν την στιγμή χρησιμοποιούνται κοινά ρολόγια.

C.P.	Name	Length	Type	Info

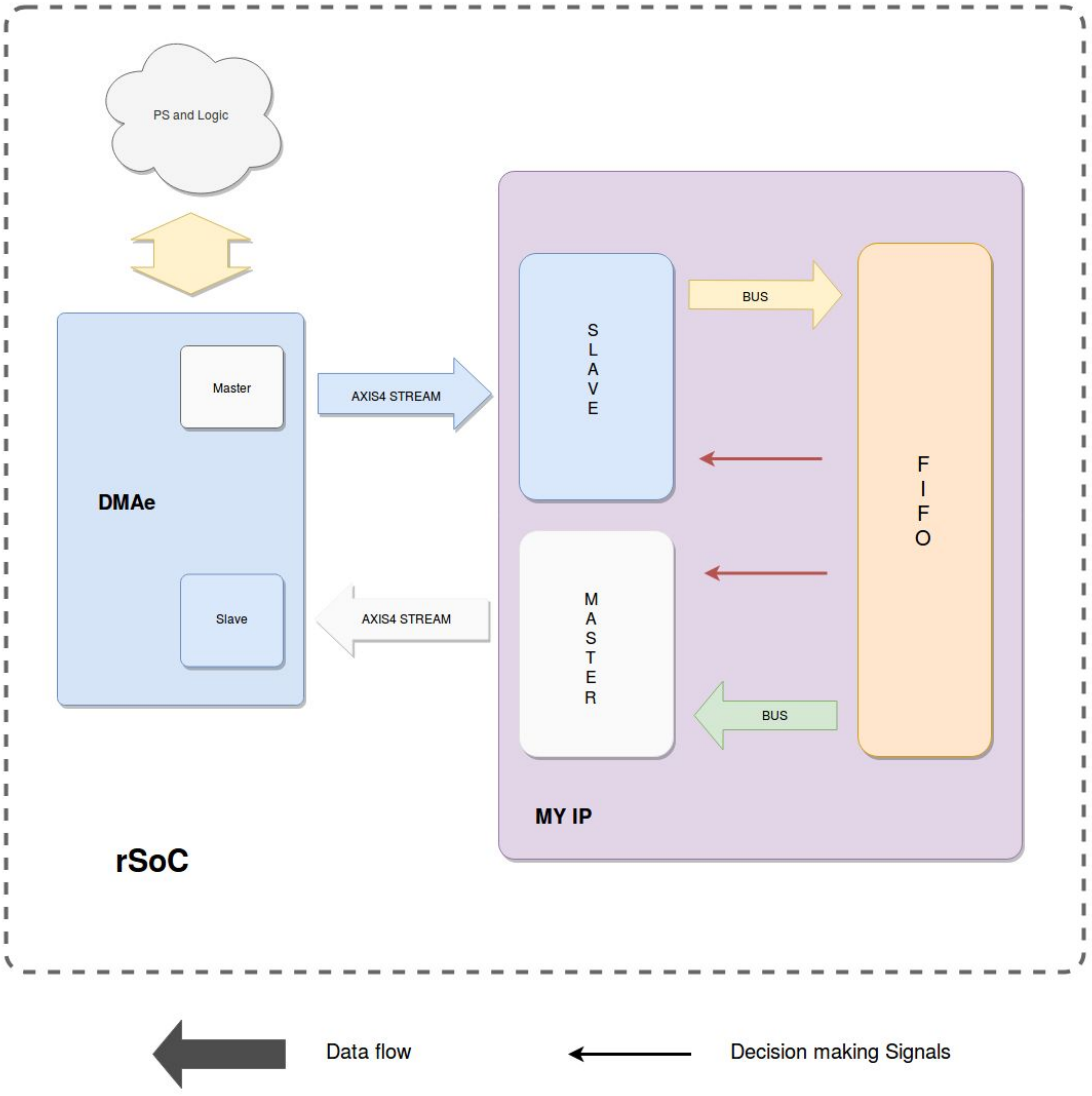
AXI4 STREAM	s00_axis_aclk	1	Input	Global clock signal
	s00_axis_aresetn	1	Input	Global reset signal
	s00_axis_tready	1	Output	Slave can accept a transfer
	s00_axis_tdata	32	Input	Primary payload
	s00_axis_tstrb	2	Input	Data type vs position type
	s00_axis_tlast	1	Input	The boundary of a packet
	s00_axis_tvalid	1	Input	Master drives a valid transfer
FIFO	fifo_w_aclk	1	Output	Write clock signal
	fifo_w_arstn	1	Output	Global reset signal
	fifo_full	1	Input	FIFO is full
	fifo_wen	1	Output	Push to FIFO
	fifo_data_in	32	Output	Data to push into FIFO

Table 2.2 - myip_v1_1_S00_AXIS

C.P.	Name	Length	Type	Info
AXI4 STREAM	m00_axis_aclk	1	Input	Global clock signal
	m00_axis_aresetn	1	Input	Global reset signal
	m00_axis_tvalid	1	Output	Master drives a valid transfer
	m00_axis_tdata	32	Output	Primary payload
	m00_axis_tstrb	2	Output	Data type vs position type
	m00_axis_tlast	1	Output	The boundary of a packet
	m00_axis_tready	1	Input	Slave can accept a transfer
FIFO	fifo_r_aclk	1	Output	Read clock signal
	fifo_w_arstn	1	Output	Global reset signal
	fifo_empty	1	Input	FIFO is empty
	fifo_ren	1	Output	Pop from FIFO
	fifo_data_out	32	Input	Data that extracted from FIFO

Table 2.3 - myip_v1_1_M00_AXIS

Με βάση τους παραπάνω πίνακες και αφού έχουμε ορίσει τις διεπαφές τόσο του MY IP module με τον DMA όσο και των εσωτερικών modules μεταξύ τους (Slave με FIFO , FIFO με Master) παρουσιάζουμε αμέσως παρακάτω την abstract σχηματική απεικόνιση του data flow μεταξύ του MY IP και DMA, όπως επίσης και μία που είναι πιο κοντά στις πραγματικές διασυνδέσεις.



Schematic 2.2

Schematic 2.3

3. Testing

Πρώτο πράγμα που χρειάστηκε να κάνουμε είναι να δοκιμάσουμε την λειτουργικότητα της FIFO ώστε να επαληθεύσουμε ότι ικανοποιεί τις προδιαγραφές τις οποίες έχουμε ορίσει παραπάνω. Στην συνέχεια και με δεδομένο την σωστή της χρήση, δηλαδή ότι μπορεί να υποστηρίξει τις λειτουργίες που του ορίζουν το control του master και του slave δοκιμάσαμε την λειτουργικότητα του MY IP συνολικά. Ενώ για το τέλος ελέγξαμε την ομαλή και σύμφωνα με τις προδιαγραφές συνεργασία του IP μας με το Top Module. Σε αυτό το σημείο πρέπει να αναφερθεί ότι έχουμε πραγματοποιήσει περισσότερα test, από αυτά που για λόγους απλότητας θα αναλυθούν στην αναφορά, όπου στο καθένα επιβεβαιώνουμε την σωστή λειτουργία και κάποιας συγκεκριμένης περίπτωσης λειτουργίας που μας ενδιαφέρει. Στο folder του project υπάρχουν τα test που αναφέρονται παραπάνω καθώς και σχετικό inline documentation με το τι θέλουμε να ελέγξουμε στο καθένα. Παρόλα αυτά τα test που αναφέρονται παρακάτω προσπαθήσαμε να είναι συνοπτικά, κατανοητά και να περιλαμβάνουν όσο το δυνατόν μπορούμε περισσότερα σημεία ενδιαφέροντος. Τέλος καλό είναι να σημειωθεί ότι τα screenshots των κυματομορφών που χρησιμοποιούνται παρακάτω υπάρχουν στο folder “**./doc/sim**”.

3.1 FIFO

Η γενική ιδέα γύρω από τον έλεγχο ορθής λειτουργίας της FIFO κυρίως κυμαίνεται στο να δούμε κατα πόσο μπορούν να γίνουν αυτόνομες εγγραφές/αναγνώσεις καθώς και η παράλληλη υποστήριξη αυτών, ακόμα τι γίνεται σε περίπτωση που προσπαθούμε να γράψουμε παραπάνω δεδομένα από όσα μας ορίζουν οι προδιαγραφές ότι μπορεί να έχει εκείνη την στιγμή. Ελέξγαμε τι γίνεται αν προσπαθήσουμε να διαβάσουμε δεδομένα όταν η FIFO είναι άδεια και τέλος ότι η FIFO μπορεί να λειτουργεί για ανεξάρτητο αριθμό δεδομένων εισόδου δηλαδή να μην σταματάει η λειτουργία της έπειτα από ένα γέμισμα και ένα άδειασμα της αλλά να συνεχίζει να λειτουργεί.

Το υποθετικό πρόγραμμα σε pseudo-code το οποίο θα αναλύσουμε κι τα expected outputs, βάση της σχεδίασης, τα οποία περιμένουμε είναι το παρακάτω:

CCs	**Pseudo - Code**	**Expected Output**
1	Reset	read_pointer = write_pointer = 62*
2-3	Nop	-
4	Push 1	Write 1 in 62 && write_pointer = 63
5	Push 2	Write 2 in 63 && write_pointer = 64
6	Push 3 && Pop 1	Write 3 in 64 && write_pointer = 0 Read 1 from 62 && read_pointer = 63
7	Push 4 && Pop 2	Write 4 in 0 && write_pointer = 1 Read 2 from 63 && read_pointer = 64
8	Push 5 && Pop 3	Write 5 in 1 && write_pointer = 2 Read 3 from 64 && read_pointer = 0
9	Pop 4	Read 4 from 0 && read_pointer = 1
10	Pop 5	Read 5 from 1 && read_pointer = 2
11	Pop 6	Read Action Failed, FIFO is empty

Table 3.1 - FIFO testing code

Στην σχεδίαση αναφέραμε ότι η FIFO έχει δύο pointers, έναν για την εγγραφη και έναν για την ανάγνωση, οι οποίοι είναι post increment. Συνεπώς μπορούμε να φανταστούμε ως “κυκλικό” πίνακα την FIFO. *Για να βεβαιωθούμε ότι όταν χρειάζεται επανέρχονται στην θέση 0, η FIFO έχει σχεδιαστεί ώστε η αρχική τιμή τους να παίρνει την τιμή 62. Έτσι κάνοντας

λίγες εγγραφές είναι δυνατή η αποσφαλμάτωση αυτής της λειτουργίας πολύ γρήγορα. Επίσης δοκιμάζουμε ότι είναι δυνατόν να γίνει τόσο ανεξάρτητα αλλά και παράλληλα εγγραφές και αναγνώσεις και μάλιστα ακόμα και η μία μετά την άλλη χωρίς κενούς χρόνους. Ενώ στον τελευταίο κύκλο δοκιμάζουμε τι θα γίνει αν προσπαθήσουμε να διαβάσουμε από μία κενή FIFO, σε αυτήν την περίπτωση δεν αλλάζουν οι τιμές των Pointers και πάμε σε κατάσταση Read η οποία αποτυγχάνει (γίνεται σχετικός έλεγχος πριν το διάβασμα και πάμε σε κατάσταση αποτυχίας, αυτό δεν πρόκειται να συμβαίνει από το υψηλότερο επίπεδο καθώς θα συνυπολογίζονται τα σήματα full και empty). Στο παρακάτω simulation μπορούν να παρατηρηθούν όλα αυτά τα οποία θα έπρεπε σχεδιαστικά να γίνονται πράγμα που μας επαληθεύει την λειτουργία της.

Simulation 3.1 - myip_v1_1_FIFO_TEST

3.2 MY IP

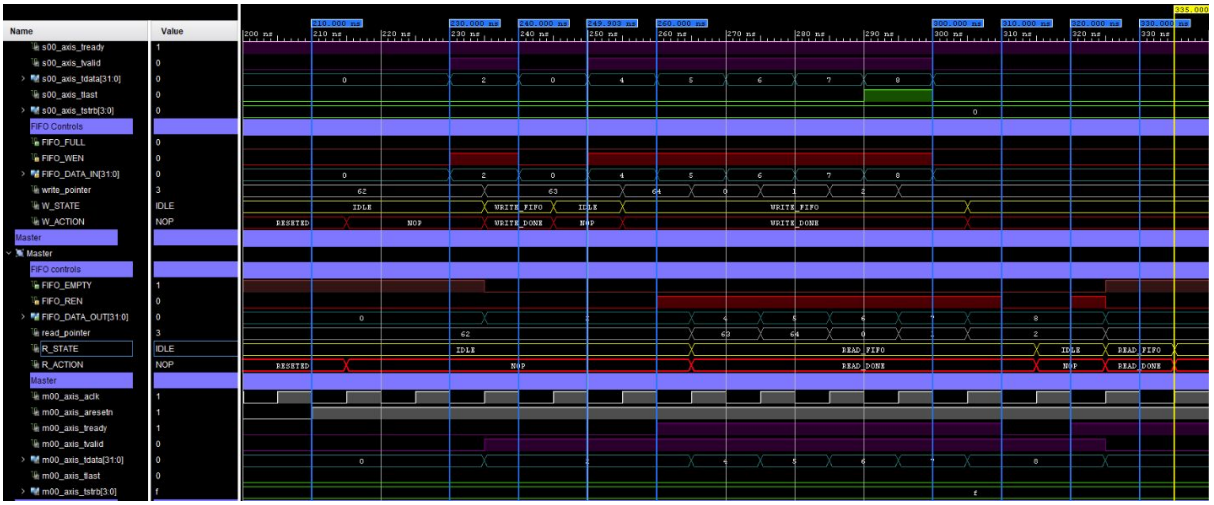
Όσον αφορά την επαλήθευση ορθής λειτουργίας του IP μας, επικεντρωθήκαμε στην επιτυχή μεταφορά πληροφορίας από τον “έξω” κόσμο σε αυτό και το ανάποδο, από την στιγμή που ξέραμε ότι ήδη λειτουργούσε η FIFO, χωρίς όμως να σημαίνε αυτό ότι δεν ελέγξαμε ότι εγγράφονται σωστά τα δεδομένα. Παρακάτω παρατίθεται το υποθετικό σενάριο το οποίο θα αναλύσουμε.

CCs	**Pseudo - Code**	**Expected Output**
1	Reset	Reset State
2	Nop	-
3	Nop	-
4	Data to write	WEN and DATA to FIFO
5	Nop	-
6	Data to write	WEN and DATA to FIFO
7	Read and Write	WEN and DATA to FIFO && REN and read DATA
8	Read and Write	WEN and DATA to FIFO && REN and read DATA
9	Read and Write	WEN and DATA to FIFO && REN and read DATA
10	Read and Write last	last WEN and DATA to FIFO && REN and read DATA
11	Read	REN and read DATA
12	Nop	-
13	Read	REN and read DATA
14	DMA Read ready without Handshake (FIFO is empty)	

Table 3.2 - MY IP testing code

Με την παραπάνω διαδικασία μπορούμε να εξασφαλίσουμε ότι χρησιμοποιώντας την λογική του AXI4 Stream protocol μπορούμε τόσο να γράψουμε και να διαβάσουμε ξεχωριστά. Όπως επίσης ότι μπορούν να γίνουν και οι δύο ενέργειες ταυτόχρονα. Άλλο ένα σημαντικό κομμάτι είναι ότι οι παραπάνω ενέργειες μπορούν να γίνουν είτε με κενούς χρόνους ενδιάμεσα της μίας από την άλλη είτε η μία μετά την άλλη να γίνονται διαδοχικά. Ενώ για κατακλείδα, αξίζει να σημειωθεί ότι ελέγχουμε, ότι οι τιμές των σημάτων tready του slave και

tvalid του master είναι ανάλογες με τις ανάγκες μας, ώστε να είμαστε σίγουροι ότι υπάρχει handshake μόνο όταν χρειάζεται. Στο παρακάτω simulation επαληθεύονται όλα όσα αναφέραμε παραπάνω.



Simulation 3.2 - myip_v1_1_TEST

3.3 TOP MODULE

Simulation 3.3 - Top Module

Στην παραπάνω κυματομορφή παρατηρούμε την συνολική και πιο γενική λειτουργικότητα όλου του συστήματος του IP μας. Οι εντολές που εκτελούνται σε αυτό το χρονικό παράθυρο είναι ένα streaming 8 κύκλων από δεκαεξαδικούς αριθμούς ξεκινώντας από το 0x10 έως το

0x80 με βήμα 0x10. Ως πρώτη παρατήρηση συμπεριφοράς της λογικής μας σε συνάρτηση με τον DMA engine είναι ότι με την ολοκλήρωση της πρώτης εγγραφής στην FIFO μας (s00_axis_tdata = 0x10) το σήμα m00_axis_tvalid ενεργοποιείται στο ‘1’ γεγονός που αποδεικνύει ότι τα δεδομένα που γράφτηκαν στον προηγούμενο κύκλο είναι έτοιμα προς ανάγνωση. Έτσι, όπως και παρατηρούμε όσο το m00_axis_tready είναι ΄1΄ (δηλαδή ο DMAe είναι έτοιμος να δεχθεί δεδομένα) το ip μας στέλνει την πληροφορία και ο DMA την παίρνει. Η διαδικασία αυτή τελειώνει όταν ο DMAe διαβάσει όλα τα δεδομένα της FIFO με αποτέλεσμα αυτή να αδειασει και να κάνει ανενεργό το σήμα των ορθών δεδομένων της (m00_axis_tvalid = ‘0’). Επίσης, δεύτερη παρατήρηση είναι ότι επειδή η FIFO μας δεν έχει γεμίσει, το s00_axis_tready παραμένει σταθερό καθόλη την διάρκεια του εν λόγω simulation test.