UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

"COMUNICAZIONE AXI TRA PROCESSORE E LOGICA FPGA NEI DISPOSITIVI XILINX"

Relatore: Prof. Daniele Vogrig

Laureando: Luca Giuriato

ANNO ACCADEMICO 2019 – 2020

Data di laurea 28/09/2023

RICONOSCIMENTI

Ringrazio il prof. Daniele Vogrig del Dipartimento di Ingegneria dell'Informazione dell'Università di Padova per avermi seguito durante il percorso di realizzazione di questa tesi di laurea.

Abstract

Negli ultimi anni Xilinx ha inserito in alcune board FPGA dei processori ARM che possono lavorare insieme alla classica logica dell'elettronica digitale delle board grazie a un bus che interconnette queste due componenti principali e tutte le componenti I/O di diversa natura presenti all' interno delle board. Grazie alle informazioni ricavate principalmente dal documento -AMBA AXI and ACE Protocol Specification- [2] fornito dal sito ufficiale ARM, questa tesi illustra le componenti del protocollo di comunicazione AMBA AXI prodotto da ARM che descrive il funzionamento di due componenti del bus di comunicazione, la prima dedicata al processo di scrittura e la seconda dedicata invece alla lettura. Successivamente vengono trattate l'adozione e l'adattamento di questo protocollo per il bus di interconnessione presente all' interno delle board della serie Zynq 7000 prodotte da Xilinx.

INDICE

1) INTRODUZIONE

Zynq-7000 è una serie di board FPGA che, oltre alla logica programmabile tipica della linea Artic 7 di Xilinx, integra anche un processore ARM multi-purpose[1]. Per interconnettere queste due componenti fondamentali tra di loro e con le varie interfacce I/O presenti nelle board è necessario l'utilizzo di un bus, come mostrato in figura 1.

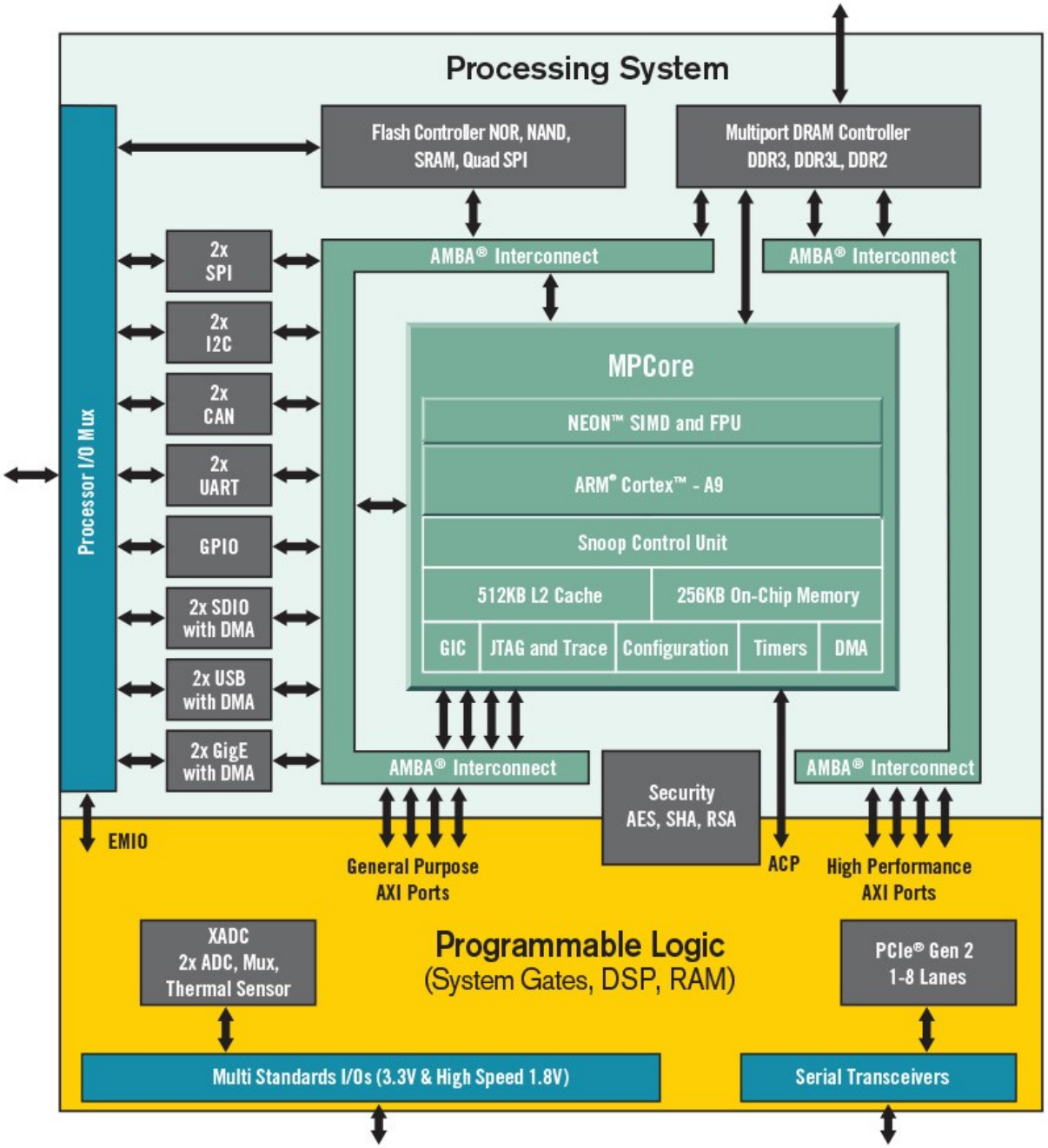

Fig.1 Architettura della board ZYNQ-7000S prodotta da Xilinx.[1]

Nella serie ZYNQ-7000, Xilinx ha scelto di adottare i protocolli AMBA AXI 4 e AMBA AXI4-Stream, ideati da ARM, per il bus di interconnessione tra la logica programmabile e il processore ARM. Il protocollo AMBA AXI 4 è un protocollo che separa i processi di scrittura

e di lettura in due sistemi di canali distinti. In entrambi i processi, nei sistemi di canali sono presenti segnali DATA, di dimensioni diverse a seconda dell'implementazione, che rappresentano l'anima della trasmissione e trasportano l'informazione. A supporto di questa componente fondamentale il protocollo prevede segnali che contengano l'indirizzo di destinazione dell'informazione e vari segnali di controllo, come dei segnali che indicano le dimensioni della transazione, il numero di trasferimenti di cui la transazione è composta, gli attributi caratteristici del tipo di memoria, segnali di feedback e molto altro. Lo scopo del protocollo, come vedremo nei prossimi capitoli, è quello di interconnettere dei componenti master a degli slave, pronti ad eseguire le richieste dei master. Il protocollo AMBA AXI4-Stream è molto simile al protocollo AMBA AXI 4, ma è caratterizzato da un solo canale di trasmissione dove il master invia informazioni a uno slave tramite un flusso di informazioni pressoché continuativo, come può essere ad esempio la trasmissione di un video, formato da pacchetti che racchiudono uno o più byte di informazione. In questo caso i segnali di supporto vengono utilizzati per delimitare i pacchetti, stabilire il loro routing e dare informazioni sulla tipologia di byte contenuti dai pacchetti. Questi protocolli vengono adottati dal bus delle board della serie ZYNQ-7000 che interconnette interfacce master e interfacce slave con canali da 32 o 64 bit.

2) IL PROTOCOLLO AMBA AXI 4

2.1 Struttura del bus e delle interconnessioni

Il protocollo AMBA AXI 4 (Advanced eXtencible Interface) prevede cinque canali di trasmissione indipendenti e unidirezionali, tre dedicati alla scrittura in memoria e due dedicati alla lettura dalla memoria, come mostrato in figura 2. Come possiamo notare, i componenti che il bus interconnette sono suddivisi in due categorie, master e slave. I dispositivi master sono quei componenti che trasmettono o richiedono dati oppure chiedono l'esecuzione di svariate operazioni agli slave che invece hanno il compito di svolgere operazioni e trasmettere o ricevere dati. Per il processo di lettura il bus prevede un primo canale, da un master a uno slave, che trasmette l'indirizzo dei dati desiderati e segnali di controllo. Successivamente, i dati letti dallo slave vengono trasferiti tramite il secondo canale, da slave a master. La scrittura necessita invece di due canali che comunicano da master a slave, il primo per trasmettere indirizzo di scrittura e segnali di controllo e il secondo per trasferire i dati che lo slave scriverà in memoria. Lo slave usa il terzo canale con lo scopo di inviare un feedback al master che ha richiesto la scrittura. Il bus mette in comunicazione molteplici interfacce suddivise in master e slave interconnesse come in figura 3, rendendo possibile gestire più transazioni da master differenti verso uno stesso slave o viceversa diverse transazioni generate da uno stesso master da uno o più slave, grazie anche a cache e buffer di cui le interconnessioni e le interfacce sono dotate. [2, cap. A1]

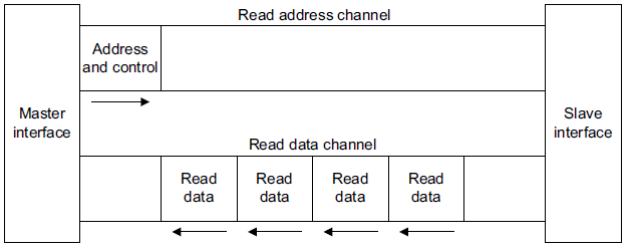

Fig.2a I due canali dedicati al processo di lettura [2].

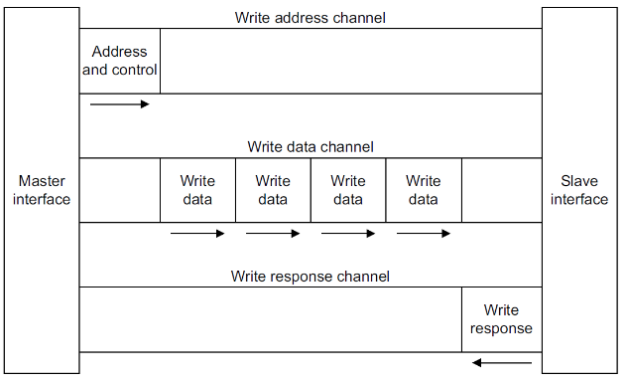

Fig.2b I tre dedicati al processo di scrittura [2].

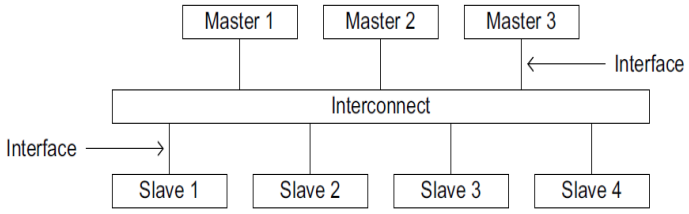

Fig.3 Struttura globale del bus, le interconnessioni collegano i master agli slave [2].

2.2 Handshake e transazioni burst

Tutti i cinque canali di trasmissione utilizzano la stessa procedura di inizio trasmissione chiamata VALID/READY hanshake. La sorgente dell'informazione genera un segnale VALID mentre la destinazione genera un segnale ready. Al primo ciclo di clock in cui entrambi i segnali sono alti, la sorgente può cominciare a inviare informazione, come mostrato in figura 4, mentre i segnali VALID e READY tornano bassi. In generale il segnale READY può eventualmente aspettare il segnale VALID per diventare alto mentre il segnale VALID deve sempre diventare alto non appena l'informazione è pronta per essere trasmessa. A seconda del canale specifico al quale i segnali VALID e READY si riferiscono, al loro nome viene aggiunto un prefisso, AW per il canale riservato all'indirizzo di scrittura, W per il canale di scrittura, B per il canale di risposta alla scrittura, AR per il canale riservato all' indirizzo di lettura e infine R per il canale di lettura.

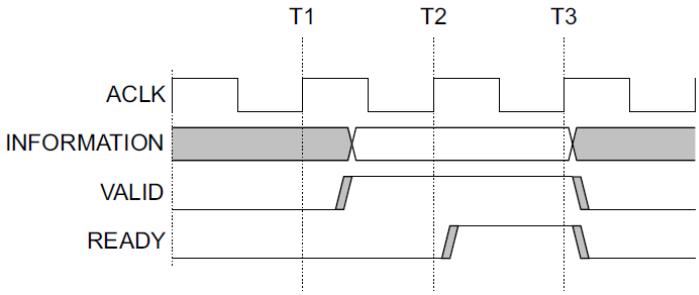

Fig. 4 Andamento dei segnali che regolano l'handshake nei cinque canali [2].

Questi prefissi vengono utilizzati in questo modo per molti altri gruppi di segnali del protocollo. In figura 5 possiamo osservare la sequenza di segnali che si susseguono nel processo di lettura. Il segnale ARVALID diventa alto quando il master è pronto e, in caso ARREADY sia basso, aspetta che anche lo SLAVE sia pronto ponendo ARREADY alto. Altrimenti, se ARREADY è già alto, il segnale RVALID viene impostato ad alto dallo SLAVE non appena può trasmettere i dati. Quando anche il master è pronto, il master pone RREADY alto e la trasmissione comincia. Una successione simile avviene anche nel processo di scrittura, come mostrato in figura 6. Quando il Master vuole effettuare una scrittura, pone AWVALID alto e, se non lo è

già, attende che anche AWREADY sia alto per inviare l'indirizzo di scrittura. WVALID diventa alto quando il master è pronto a trasmettere e aspetta

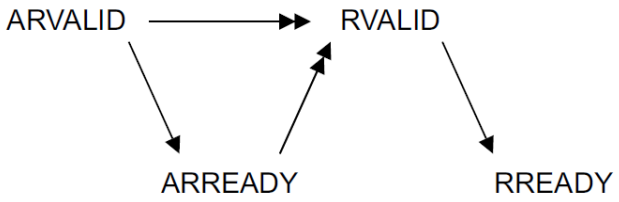

Fig.5 Successione dei segnali di handshake che regolano il processo di scrittura [2].

che WREADY sia alto per poterlo fare. Possiamo notare come WVALID non richiede che AWREADY sia già alto per essere posto al valore alto. Le transazioni nei bus AXI sono burst-based, ossia i segnali di controllo del protocollo non si usano per ogni trasferimento di cui è composta la transazione ma solo in fase iniziale. Per questo motivo il canale di scrittura deve utilizzare anche un segnale WLAST per indicare l'ultima trasmissione di dati della transazione. Oltre a necessitare di AWVALID, AWREADY, WVALID e WREADY alti, solo dopo il riscontro di un WLAST alto lo slave può assegnare il valore alto a BVALID nel canale di risposta e, dopo aver atteso che BREADY dal lato del master diventi alto, segnala il successo della transazione grazie a un segnale chiamato BRESP.

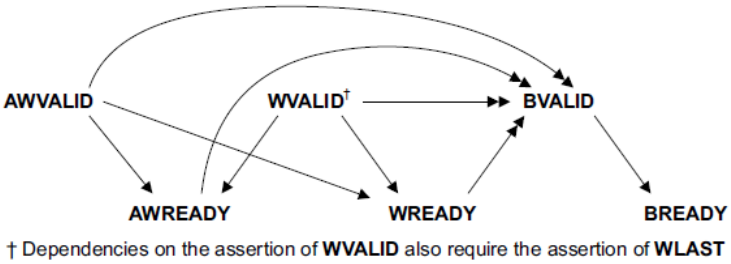

Fig.6 Successione dei segnali di handshake che regolano il processo di lettura [2].

Nei canali dedicati agli indirizzi vengono anche comunicate le informazioni riguardanti le caratteristiche di una transazione burst. Attraverso i segnali di otto bit ARLEN e AWLEN, viene comunicata la lunghezza del burst tramite i due canali di indirizzo e attraverso ARSIZE e AWSIZE viene codificato il numero di byte trasportati da ogni singolo trasferimento di dati del burst, in modo tale che il numero in binario rappresenti l'esponente dell'esponenziale con base due che rappresenta il numero di byte. Ad esempio, con un AWSIZE pari a `010` si ha un burst lungo 4 byte (2 alla seconda). Esistono tre modalità di burst, ovvero FIXED, INCR e WRAP. La modalità FIXED utilizza lo stesso indirizzo per ogni trasferimento del burst, utile ad esempio per svuotare o riempire uno stack FIFO. La modalità INCR incrementa l'indirizzo a quello successivo dopo ogni trasferimento, utile per i classici accessi alla memoria. Infine, la modalità WRAP è simile alla modalità INCR ma con la possibilità di allacciarsi a un indirizzo inferiore una volta raggiunto l'indirizzo massimo prima della fine del burst. La tipologia di

burst viene indicata dai segnali ARBURST e AWBURST di tre bit. Nel canale di scrittura è presente il segnale di trasmissione dei dati largo 8, 16, 32, 64, 128, 256, 512 o 1024 bit, chiamato WDATA. Un segnale aggiuntivo WSTROBE, composto da un numero di bit pari a quello di WDATA diviso per otto, viene utilizzato per indicare quali sezioni del bus stanno trasmettendo segnali validi. Infatti, a WSTROBE[i] corrispondono i bit WDATA[8i+7,8i], validi quando WSTROBE[i] è alto. In caso i trasferimenti non occupino tutto il bus, i trasferimenti successivi occupano linee di byte differenti per le modalità di burst INCR e WRAP, come mostrato in figura 7, mentre nella modalità FIXED ogni trasferimento utilizza sempre le stesse linee di byte.

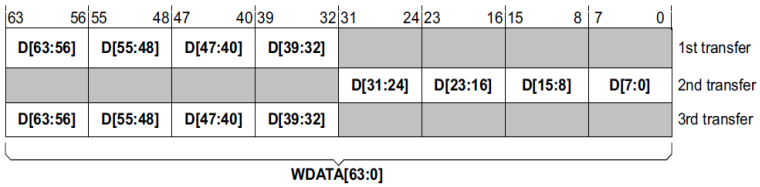

Fig. 7 Allocazione nel bus di trasferimenti di una transazione burst in modalità INCR o WRAP in caso i trasferimenti siano di dimensione inferiore alla grandezza del bus [2].

I segnali di risposta, composti da due bit, sono RRESP per il processo lettura, trasmesso nel canale di lettura da slave a master, e BRESP per la scrittura, trasmesso nel canale di risposta alla scrittura da slave a master. Con due bit si possono comunicare i quattro tipi di risposta possibili, OKAY, EXOKAY, SLVERR e DECERR. OKAY indica il successo di un normale accesso, EXOKAY indica il successo di un accesso esclusivo, tipologia di accesso discussa al paragrafo 2.5, SLVERR indica che lo slave vuole comunicare una condizione di errore al master e infine DECERR è utilizzato da un'interconnessione per segnalare l'assenza di uno slave all' indirizzo di transazione. [2, cap. A3]

2.3 Attributi di una transazione

Gli slave sono divisi in due categorie, i "*memory slave"*, capaci di gestire correttamente qualsiasi transazione, e i "*peripheral slave*", capaci di gestire solo alcuni tipi di transazione predefiniti nel loro metodo di accesso. Non potendo violare il protocollo, un "*peripheral slave"* completa anche gli altri tipi di transazione, ma non fornisce garanzia sulla correttezza del processo. I segnali AWCACHE e ARCACHE di 4 bit sono due segnali che trasportano informazioni sugli attributi delle memorie e sono presenti nei canali di indirizzo per la scrittura e per la lettura. Sintetizzando i due segnali con AxCACHE, ognuno dei 4 bit specifica un aspetto della transazione. AxCACHE[0] è il bit denominato "*bufferable*" e indica con un valore alto che la transazione può essere momentaneamente sospesa con dei buffer lungo il tragitto ed essere conclusa con dei cicli di clock aggiuntivi. AxCACHE[1] è il bit denominato "*modifiable*". Quando questo bit è basso molte caratteristiche della transazione non possono

essere modificate, come AxSIZE, AxLEN, AxBUSRT e AxADDR, impedendo ad esempio la suddivisione in più transazioni o viceversa. Viene fatta un'unica eccezione per gli accessi esclusivi, che verranno discussi nel paragrafo 2.5, nei quali AxSIZE e AxLEN possono variare a patto che non vari il numero totale di byte della transazione. Se invece AxCACHE[1] è alto, i valori citati in precedenza possono invece essere modificati e sono quindi consentite le suddivisioni o unioni di transazioni. Infine, i bit AxCACHE[2] e AxCACHE[3] sono chiamati rispettivamente "*read-allocate*" e "*write-allocate*" e indicano che la transazione potrebbe essere velocizzata, in caso almeno dei due sia alto, consultando la cache. Questi due bit hanno quindi duplice funzione, in caso il valore del bit sia alto indicano che le informazioni della transazione devono essere immagazzinate nella cache ma indicano anche che la cache potrebbe contenere ciò che la transazione richiede grazie a una precedente transazione di lettura o scrittura. AxCACHE codifica quindi tutti i possibili tipi di memoria elencati in tabella 1. Oltre agli attributi di memoria, attraverso i segnali di tre bit ARPROT e AWROT vengono codificati gli attributi di accesso. Sintetizzando i due segnali on AxPROT, AxPROT[0] alto indica un accesso previlegiato, AxPROT[1] alto indica uno stato operativo non sicuro e infine AxPROT[2] indica un accesso riguardante istruzioni se è a valore alto, mentre a valore basso indica un accesso riguardante informazioni. Oltre agli attributi di memoria, attraverso i segnali di tre bit ARPROT e AWROT vengono codificati gli attributi di accesso. Sintetizzando i due segnali on AxPROT, AxPROT[0] alto indica un accesso previlegiato, AxPROT[1] alto indica uno stato operativo non sicuro del master e infine AxPROT[2] indica un accesso riguardante istruzioni se è a valore alto mentre a valore basso indica un accesso riguardante informazioni. [2, cap. A4]

ARCACHE[3:0]	AWCACHE[3:0]	Memory type
0000	0000	Device Non-bufferable
0001	0001	Device Bufferable
0010	0010	Normal Non-cacheable Non-bufferable
0011	0011	Normal Non-cacheable Bufferable
1010	0110	Write-through No-allocate
1110 (0110)	0110	Write-through Read-allocate
1010	1110 (1010)	Write-through Write-allocate
1110	1110	Write-through Read and Write-allocate
1011	0111	Write-back No-allocate
1111 (0111)	0111	Write-back Read-allocate
1011	1111 (1011)	Write-back Write-allocate
1111	1111	Write-back Read and Write-allocate

Tab. 1 Possibili tipi di memoria codificati nel protocollo AXI. I valori non presenti sono riservati [2].

2.4 Transazioni multiple e modello di ordinamento

Il protocollo AXI prevede anche degli identificatori di transazione, AXI ID, trasmessi dai segnali AWID, ARID, BID e RID, che si riferiscono rispettivamente al canale di indirizzo di scrittura, al canale di indirizzo di lettura, al canale risposta alla scrittura e al canale lettura. L'unica eccezione si presenta nel canale di scrittura dove il segnale WID, presente nella versione 3 del protocollo AXI, è stato rimosso poiché AXI4 non prevede il "*write interleaving*", ossia che alcuni slave possano processare transazioni di scrittura con diversi ID contemporaneamente, rendendo il segnale WID ridondante e superfluo e grazie alla sua rimozione è stato ridotto il peso dei pin dell'interfaccia. In generale tutte le transazioni con lo stesso ID, qualunque sia il canale, devono essere trasmesse nell' ordine di ricezione e ricevute nell'ordine di trasmissione. Ad esempio, durante un processo di lettura, uno slave che risponde a multiple transazioni con lo stesso ARID deve inviare i dati letti al master nello stesso ordine di indirizzo con cui il master richiede le transazioni. Il segnale RID ha in questo caso un funzionamento simile a un aknowledgement dato che copia il valore di ARID a cui si riferisce. Anche la ricezione da parte del master deve avvenire in ordine. Infatti, se il master esegue transazioni con lo stesso ARID destinate a più slave differenti, le interconnessioni hanno il compito di riorganizzare le varie risposte da parte degli slave nell'ordine utilizzato precedentemente dal master. Se invece più transazioni hanno ID diverso, il protocollo non prevede un ordine preciso poiché il valore di ID è sufficiente a non generare ambiguità. Nel canale di scrittura avviene un processo analogo nel quale i segnali AWID e BID svolgono rispettivamente funzioni simili a ARID e RID. Le interconnessioni aggiungono a loro volta bit addizionali ai segnali ARID e AWID con una codifica che identifica univocamente ogni master. Questo meccanismo risulta vantaggioso perché i master non hanno più la necessità di controllare gli ID usati dagli altri master e possono lavorare in maniera più semplice e indipendente. Nel caso di una lettura, dato che alla singola porta fisica di accesso al master corrispondono molteplici porte logiche, oltre a bit che indicano il master vengono aggiunti bit anche per identificare la giusta transazione di lettura tra quelle in corso nel master specifico. Gli slave modificano a loro volta i segnali di risposta riproponendo i bit aggiuntivi nei loro segnali ID. A differenza del numero di bit di ID generati dai master, variabile a seconda dell'implementazione, le aggiunte hanno un numero fisso di bit, 4 per identificare il numero del master e 4 per identificare la porta logica a cui è destinato il processo. Di conseguenza gli slave aggiungeranno 8 bit ai segnali ID di risposta per tenere conto di queste due componenti. Grazie all' unicità degli ID creati da interfacce e interconnessioni, ogni master viene univocamente identificato dai suoi ID, permettendo l'integrazione di un modello di ordinamento nel protocollo. Il modello di ordinamento consiste in una serie di regole che specificano l'ordine in

cui le interfacce devono rispondere alle varie transazioni. Parte di queste regole sono già state discusse in questo paragrafo. Il modello di ordinamento si assicura che l'ordine delle transazioni richieste da un'interfaccia venga mantenuto nelle risposte corrispondenti e specifica con quale ordine le transazioni concorrenti debbano essere considerate. Infatti, il protocollo non previlegia la lettura alla scrittura o viceversa e di conseguenza un'interfaccia completa una transazione in corso prima di iniziare una transazione dell'altro tipo. Inoltre, nel caso degli accessi in memoria riguardino, parzialmente o totalmente, gli stessi indirizzi, le transazioni di questi accessi non possono essere processate contemporaneamente ma si deve rispondere nell' ordine in cui queste sono arrivate allo slave.
[2, cap. A5 e A6]

2.5 Accesso esclusivo

Per evitare che il bus sia dedicato a un master per molto tempo, è presente anche una modalità di accesso esclusivo. Questo meccanismo concettualmente simile a un sistema semaforico non intacca la latenza e neanche la banda massima raggiungibile. L'utilizzo dell'accesso esclusivo viene comunicato dai due segnali di un bit AWLOCK e ARLOCK presenti nei due canali di indirizzo e controllo mentre il successo o fallimento dell'accesso esclusivo vengono comunicati tramite BRESP e RRESP nei due canali di risposta. L'accesso esclusivo è suddiviso in due parti. Il primo passo è compiere una lettura esclusiva all'indirizzo di interesse. Lo slave interessato monitora questo indirizzo e segnala il successo della lettura esclusiva riportando con RRESP il valore EXOKAY. Gli slave che supportano l'accesso esclusivo monitorano gli indirizzi includendo un numero di unità hardware pari al numero di ID dei master che possono effettuare accessi esclusivi. Questi monitor registrano il valore di ARID e l'indirizzo della transazione di lettura. Quando il master riceve la conferma EXOKAY può cominciare la seconda parte dell'accesso esclusivo, una scrittura esclusiva. In questa scrittura, il valore di AWID copia il valore di ARID usato nella lettura precedente e tenta di scrivere allo stesso indirizzo usato in precedenza. Se l'indirizzo di destinazione non è stato modificato dopo la lettura esclusiva, lo slave completa la scrittura esclusiva e riporta ancora una volta EXOKAY tramite il segnale BRESP per segnalare il successo dell'operazione. Se invece lo slave non può completare la scrittura, riporta invece il valore OKAY per segnalare un errore. Questo avviene in due casi, ossia in caso l'indirizzo monitorato venga modificato prima dell'inizio della scrittura oppure in caso un diverso tentativo di accesso esclusivo con lo stesso ARID resetti il monitor che controllava l'indirizzo di interesse. Nel caso un master tenti un accesso esclusivo verso uno slave che non supporta questo meccanismo, lo slave ignora il segnale ARLOCK e risponde alla lettura esclusiva con OKAY al posto di EXOKAY. Il master interpreta OKAY

come un errore e non procede con la parte di scrittura. Poiché gli accessi esclusivi sono pensati per non occupare troppo il bus, vengono poste per questo tipo di accesso delle limitazioni riguardanti le dimensioni della transazione, ovvero il burst non può essere composto da più di 16 trasferimenti e la dimensione massima della transazione è 128 bytes. Bisogna anche tenere a mente che in ogni caso i monitor devono poter vedere le transazioni di un accesso esclusivo e di conseguenza è importante che si utilizzino dei corretti valori di AxCACHE. Se ad esempio la transazione di lettura si risolve grazie alla cache, il monitor non ha modo di constatare che tale transazione sia avvenuta e di conseguenza non è in grado di garantire la correttezza dell'accesso esclusivo. [2, cap. A7]

2.6 Segnali aggiuntivi

Il protocollo AXI prevede anche dei gruppi di segnali aggiuntivi che possono essere utilizzati per svolgere funzioni diverse a seconda dello scopo ricercato da chi vuole implementarli. Il gruppo di segnali (QoS) comprende i due segnali, AWQOS e ARQOS per i canali di indirizzo di lettura e scrittura, ciascuno di 4 bit. Il loro uso standard previsto dal protocollo è quello di stabilire il livello di priorità delle transazioni in caso un componente sia interessato da più transazioni contemporaneamente, a un valore alto di AWQOS o ARQOS corrisponde alta priorità. Quando interfaccia non utilizza i segnali QoS li pone al valore 0, il loro valore standard. Questa modalità di utilizzo non è obbligatoria, infatti un qualsiasi sistema metodologico di utilizzo dei segnali QoS può essere implementato a condizioni che i master siano forniti di una componente programmabile che permetta di manipolare questo tipo di segnali. Un altro gruppo di segnali aggiuntivi sono i due segnali AWREGION e ARREGION di 4 bit. La memoria può essere divisa in un massimo di 16 regioni e questi due segnali, uno per la lettura e uno per la scrittura, permettono di individuare la regione di memoria in qui sono presenti gli indirizzi interessati dalla transazione. Questo permette ad esempio di eliminare delle componenti decoder di supporto agli slave che con un solo accesso fisico provvedono molteplici accessi logici. Il funzionamento di questi segnali è definito dall' implementazione ma devono anche rispettare il protocollo. Ad esempio, se un sistema prevede meno di 16 regioni l'implementazione deve provvedere un modo per rispettare il protocollo anche per i valori di regioni non presenti, grazie a dei messaggi di errore o una riorganizzazione della memoria che occupi anche le regioni non utilizzate. Infine, il gruppo di segnali User è composto da un set di segnali che coprono ogni canale grazie a AWUSER, WUSER, BUSER, ARUSER e RUSER. I segnali user possono essere implementati con un design stabilito dall' utilizzatore e non è necessario implementarli in tutti i canali. Ad ogni modo, l'utilizzo dei segnali User è sconsigliato perché il loro comportamento non è definito dal protocollo e questo può causare

problemi di cooperazione tra due componenti nel caso i segnali vengano utilizzati in maniera incompatibile. [2, cap. A8]

2.7 Interfaccia a bassa potenza

Il protocollo prevede anche un'estensione, chiamata interfaccia a bassa potenza, che svolge due funzioni diverse a seconda del componente interessato. Se il componente non implementa uno stato di bassa potenza, questa estensione ha il solo compito di attivare o disattivare il segnale di clock del componente. Nel secondo caso, l'estensione guida l'entrata nello stato di bassa potenza o l'uscita del componente da questo stato. In entrambi i casi, il segnale CACTIVE indica al sistema di controllo del clock quando il clock deve essere attivo nel componente. Quando il clock deve essere attivo in ogni momento il segnale CACTIVE ha valore alto, mentre quando CACTIVE è basso il sistema di controllo del clock può disattivarlo. Ad ogni modo, Il segnale CACTIVE basso non impone la disabilitazione del clock poiché può restare basso permanentemente durante processi in cui il clock viene attivato e disattivato frequentemente. Inoltre, durante la richiesta di entrata o uscita dallo stato a bassa potenza, entrano in gioco due ulteriori segnali, CSYSREQ e CSYSACK. CSYSREQ viene utilizzato dal sistema per richiedere al componente il cambio di stato mentre CSYSACK viene usato dal componente per confermare la ricezione della richiesta. Per richiedere al componente di entrare nello stato di bassa potenza, il sistema pone CSYSREQ, che con un valore di default alto indica il normale funzionamento del componente, al valore basso. Se il componente può effettuare il cambio di stato pone il segnale CACTIVE basso e successivamente pone basso anche il segnale CSYSACK per confermare la ricezione della richiesta, come mostrato in figura 7.

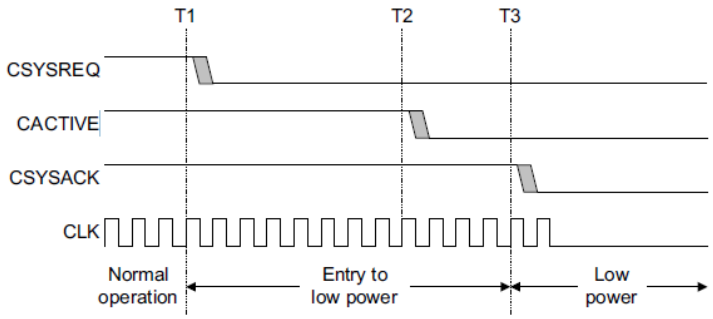

Fig. 7 Andamento dei segnali quando un componente accetta di entrare nello stato di bassa potenza [2].

In questo modo il componente entra nello stato di bassa potenza con successo. In caso il componente non potesse entrare nello stato di bassa potenza al momento della richiesta, lascia il segnale CACTIVE alto e risponde con CSYSACK basso comunicando la ricezione e il rifiuto della richiesta. CSYSREQ e CSYSACK devono però essere alti per indicare che il componente

si trova nello stato standard e di conseguenza l'ultima fase del rifiuto consiste nell'alzare i due segnali, prima CSYSREQ e successivamente CSYSACK, come mostrato in figura 8.

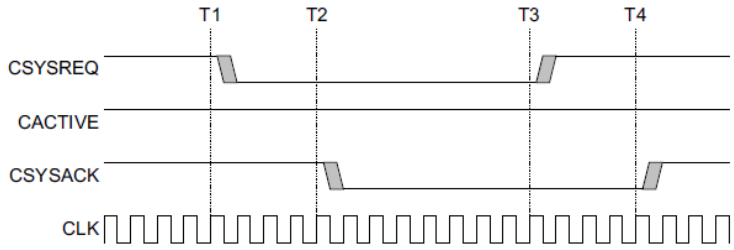

Fig. 8 Andamento dei segnali quando un componente rifiuta di entrare nello stato di bassa potenza.

L'uscita dallo stato di bassa potenza da parte del componente avviene in maniera molto simile alla seconda parte del rifiuto di entrata nello stato di bassa potenza. Infatti, il segnale CACTIVE viene portato al valore alto, successivamente il sistema alza il segnale CSYSREQ e infine il componente deve portare al valore alto CSYSACK, come mostrato in figura 9. Se invece è il sistema a richiedere l'uscita da questo stato, il primo segnale che viene alzato è CSYSREQ, seguito da CACTIVE e CSYSACK, come mostrato in figura 10. [2, cap. A9]

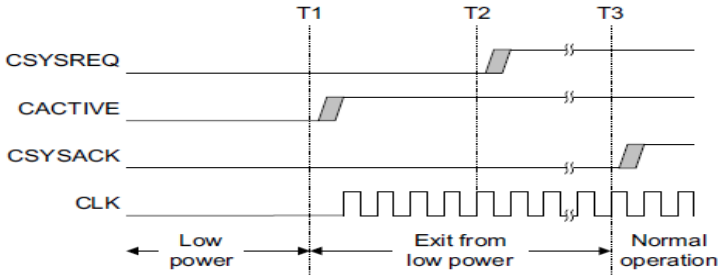

Fig. 9 Uscita dallo stato di bassa potenza da parte del componente [2].

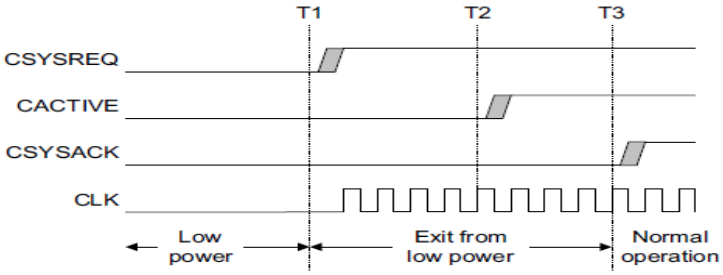

Fig. 10 Il sistema di controllo del clock richiede al componente di uscire dallo stato di bassa potenza [2].

3) IL PROTOCOLLO AMBA 4 AXI4-STREAM

3.1 Affinità con il protocollo AMBA AXI

Il protocollo AMBA AXI-Stream è un protocollo che presenta varie similitudini con il canale di scrittura del protocollo AMBA AXI, pensato però per applicazioni con un flusso di dati scambiati molto lungo o addirittura continuativo. Questo protocollo è stato ideato per una connessione point-to-point tra un componente trasmettitore e un componente ricevitore, ma risulta comunque possibile utilizzare delle adeguate interconnessioni per estendere il suo utilizzo a molteplici componenti. Il protocollo utilizza la stessa procedura di handshake vista precedentemente nel protocollo AXI attraverso i due segnali TVALID e TREADY. Dopo avere stabilito una connessione, il componente trasmettitore invia dei pacchetti, similmente a come un master nel protocollo esegue transazioni di scrittura verso la memoria di uno slave.

3.2 Tipi di byte e segnali che supportano i trasferimenti

Nel protocollo AXI-Stream vengono definiti tre tipi di byte, Data byte, Position byte e Null byte. Data byte e Position byte devono essere sempre trasmessi da sorgente a destinazione e in entrambi devono essere preservati il numero dello specifico tipo di byte nello stream e anche la sua posizione relativa rispetto agli altri Data e Position byte. La differenza tra i due tipi di byte giace nel mantenere il valore associato a quei dati. Infatti, il valore associato a un Data byte deve sempre essere trasmesso, diversamente a quanto succede al valore di un Position byte che non deve essere per forza trasmesso. Infine, il Null byte è un tipo di byte utilizzato solamente per riempire dello spazio allo scopo di rispettare dei vincoli nella trasmissione e non essendo parte dell'informazione vera e propria vengono eliminati prima di raggiungere la destinazione. I valori dei byte vengono trasportati dal segnale TDATA, segnale accompagnato dai segnali TKEEP e TSTRB.

TKEEP	TSTRB	Data Type	Description
HIGH	HIGH	Data byte	The associated byte contains valid information that must be transmitted between source and destination.
HIGH	LOW	Position byte	The associated byte indicates the relative position of the data bytes in a stream, but does not contain any relevant data values.
LOW	LOW	Null byte	The associated byte does not contain information and can be removed from the stream.
LOW	HIGH	Reserved	Must not be used.

Tab.2 Le combinazioni di TKEEP e TSTRB individuano il tipo di byte trasmesso.

TKEEP indica con un valore alto che il byte a cui è associato deve essere trasmesso a destinazione, rendendo di fatto possibile identificare i Null Bytes caratterizzati dal segnale TKEEP a valore basso, mentre TSTRB indica un Data byte quando ha valore alto e un Position byte a valore basso. Le combinazioni di questi due segnali, riassunte in tabella 2, rendono quindi possibile distinguere i tre tipi di byte nella trasmissione. [3, cap. 2.3 e 2.4]

3.3 Pacchetti di dati

Un pacchetto di dati è un insieme di byte inviati in più trasferimenti. I pacchetti di byte del protocollo AXI-Stream rispecchiano i burst del protocollo AXI presentati nel capitolo 2. Per identificare i pacchetti si utilizzano tre segnali, ossia TID, TDEST e TLAST. TID è un segnale che trasporta il codice che identifica una particolare comunicazione, similmente a come i vari segnali AXI ID identificano le varie transazioni nel protocollo AXI. TDEST trasporta invece informazioni utili al routing dei pacchetti indicando la destinazione del pacchetto. Infine, il segnale TLAST è un segnale impiegato nella definizione degli estremi del pacchetto. Un byte di qualsiasi tipo a cui viene abbinato il segnale TLAST a valore alto viene considerato come ultimo byte di un pacchetto. Nel protocollo non vengono previsti segnali che indicano invece l'inizio di un nuovo pacchetto. Un nuovo pacchetto viene dunque identificato dai valori di TID e TDEST dopo il reset oppure dai valori di TID e TDEST del primo byte dopo la fine di un pacchetto precedente. Più pacchetti con gli stessi valori di TID e TDEST e con TLAST a valore basso possono essere uniti in un'operazione di merging se necessario. Inoltre, le interconnessioni possono aggiungere delle componenti ai segnali TID e TDEST con lo scopo di distinguere due trasmissioni identiche nei valori TID e TDEST e fornire informazioni aggiuntive utili per il routing. Un pacchetto può anche contenere trasferimenti in cui i byte sono tutti di tipo Null se necessario. Questo può essere utile per incentivare lo scorrimento di byte che sono in attesa, contenuti in buffer intermedi, per indicare la fine di un pacchetto quando non ci sono più Data o Position byte da inviare oppure per terminare una operazione per la quale lo slave di destinazione si aspetti un segnale TLAST a valore alto. Infine, per soddisfare la necessità di fornire informazioni aggiuntive durante la trasmissione viene previsto anche il segnale di banda secondaria TUSER che ha un utilizzo molto vario a seconda delle esigenze della trasmissione. Alcuni tra gli usi più comuni di questo segnale sono comunicare la locazione o il tipo di uno specifico byte, identificare segmenti di uno stesso pacchetto oppure fornire informazioni di controllo come bit di parità oppure vari flag. [3, cap. 2.5,2.6 e 2.8]

3.4 Interleaving dei trasferimenti e ordinamento

A differenza del protocollo AXI che non permetteva il “write interleaving”, il protocollo AXI-Stream permette invece l’utilizzo dell’interleaving dei trasferimenti di diverse comunicazioni. In presenza di slave che lavorano adeguatamente quando operano su pacchetti di byte interi, il segnale TLAST funge da arbitro indicando quando le comunicazioni possono essere intrecciate nell’interleaving. Gli slave possono essere progettati per lavorare senza un limite di comunicazioni comprese nell’ interleaving e in questo caso non ci sono particolari condizioni da rispettare. Invece, per gli slave il cui design prevede una capacità massima di interleaving si possono applicare diverse strategie per non violare questo parametro. Una prima strategia è quella di permettere l’interleaving a un solo master che può intrecciare più comunicazioni a piacimento. Un’altra possibilità è quella di lasciare comunicare più master contemporaneamente ma permettendo a ciascun master una sola comunicazione. Un sistema di alto livello si occuperà successivamente di operare l’interleaving tra le varie comunicazioni. Risulta comunque possibile combinare le due soluzioni a patto che un sistema di alto livello si occupi della gestione dell’interleaving, garantendo che la capacità massima dello slave non venga superata. Poiché il protocollo non prevede il riordinamento dei trasferimenti, tutti i trasferimenti devono rimanere ordinati in ogni momento della comunicazione. Questa caratteristica aumenta la prevedibilità del sistema, garantisce che l’interleaving a livello dello slave non aumenti di dimensioni, diminuisce la complessità del sistema e rende possibile un acknowledgement cumulativo. [3, cap. 4.1]

4) ADOZIONE DEI PROTOCOLLI AMBA AXI E AMBA AXI-STREAM NEGLI XILINX IP

4.1 Protocollo AXI e gli Xilinx IP

Nei tool di Xilinx è presente una vasta gamma di IP basati sul protocollo AXI 4 che permettono l'utilizzo del bus AMBA AXI nelle board [4]. Gli IP (Intellectual Property) sono dei blocchi circuitali pre-progettabili, utilizzabili nello sviluppo di un design più complesso come elementi di base. Questo rende la progettazione del design di una board FPGA molto più semplice e veloce. Non tutte le potenzialità del protocollo vengono però sfruttate dagli IP in quanto vengono adottate solo una parte delle finzioni del protocollo AXI. Inoltre, alcuni IP sfruttano invece il protocollo AXI 4-Lite, una versione semplificata di AXI 4 dove i burst i segnali AxLEN sempre impostati a 1, ossia i burst sono sempre composti da un singolo trasferimento, e i trasferimenti devono sempre occupare tutto il bus [2]. Altri IP usano invece AXI4-Stream. I protocolli AXI 4, AXI 4-Lite e AXI4-Stream cooperano anche con un Infrastructure IP, ossia un IP generico che supporta l'assemblamento del sistema trasportando e trasformando informazioni tramite interfacce AXI4 general purpose, senza però interpretare in alcun modo queste informazioni [4].

4.2 Adozione del Protocollo AMBA AXI

Gli IP basati sul protocollo AXI prevedono una serie di parametri per le transazioni descritte nel capitolo 2, ricavati da "Vivado design suite: AXI Reference Guide" [4, Cap. 4]. Negli IP i segnali READY e VALID sono totalmente supportati poiché fanno parte dei segnali essenziali per la comunicazione nel bus. Il valore dei segnali AxLEN ha un limite massimo 256 per i burst di tipo INCR mentre per i burst di tipo WRAP ha un limite massimo di 16. Questo significa che i burst di tipo INCR possono essere formati da, al massimo, 256 trasferimenti mentre i burst WRAP sono al massimo composti da 16 trasferimenti. I burst di tipo FIXED non vengono supportati dagli Xilinx IP perché si comportano in maniera non efficiente e difficilmente ottimizzabile in questo tipo di applicazioni. I segnali AxSIZE possono indicare una larghezza dei trasferimenti di 32, 64, 128, 256, 512 oppure 1024 bit, a eccezione di AXI 4-lite che invece ha a disposizione solo trasferimenti da 32 o 64 bit. È sconsigliato l'utilizzo di trasferimenti che non occupino l'intero bus perché porta a un sistema non pienamente efficiente e questo può causare l'aumento delle risorse utilizzate. Quando due IP con una larghezza dei dati diversa intendono comunicare, interviene l'Infrastructure IP garantendo la conversione dalla larghezza di origine a quella richiesta alla destinazione. Per quanto riguarda gli accessi esclusivi, i segnali a loro supporto sono generalmente presenti e le informazioni vengono trasportate dall'Infrastructure IP. Gli Xilinx IP però non supportano questo tipo di accesso e rispondono

sempre con il valore OK senza mai utilizzare EXOKAY. Questo comporta che la seconda parte degli accessi esclusivi, la scrittura, non potrà mai avvenire rendendo gli accessi esclusivi impossibili da attuare negli Xilinx IP. Per quanto riguarda i segnali di attributo, i segnali AxPROT sono impostati sempre al valore 000, indicando di default una transazione sicura e sempre riguardante dei dati. I segnali AxCACHE invece sono sempre impostati a 0011 indicando che le transazioni possono usufruire di buffer ed essere modificate per migliorare la performance del sistema, ma non si prevede l'utilizzo di cache per immagazzinare le transazioni eseguite. Passando ora a gruppi di segnali supplementari, I bit dei segnali QoS vengono trasportati dall'Infrastructure IP ma generalmente ignorati dagli Endpoint IP (Con Endpoint si intende una serie di interconnessioni interne a un design che trasporta dati dall'esterno all' interno dell'IP o viceversa [5]). Anche i segnali User vengono trattati allo stesso modo, ma sono fortemente sconsigliati perché a differenza dei segnali QoS non possono essere elaborati da dei convertitori in caso le dimensioni atomiche degli Endpoint IP siano differenti. Per quanto riguarda i segnali AxREGION, la loro funzione di decoding è difficile da predire e mantenere. L'Infrastructure IP li trasporta, però i Mater IP non li producono e questo può influenzare negativamente la connessione point-to-point con degli Slave IP. Per queste ragioni i design di Slave IP che supportano i segnali AxREGION sono fortemente sconsigliati. Per quanto riguarda lo stato a bassa potenza, le interfacce a bassa potenza non sono supportate e di conseguenza nei design delle interfacce IP i segnali CSYSREQ, CSYSACK e CACTIVE non sono presenti. I tool di Xilinx permettono inoltre di creare delle interfacce write-only o read-only. L'unidirezionalità di questa scelta rende possibili delle ottimizzazioni logiche notevoli da parte di tutti gli AXI IP, incluso L'Infrastructure IP, rendendo il sistema molto più efficiente.

4.3 Adozione del protocollo AMBA AXI-Stream

Il protocollo AXI-Stream viene adottato da Xilinx per un'ampia varietà di tipi di comunicazione diverse e di seguito verranno mostrati solo alcuni esempi ricavati da "Vivado design suite: AXI Reference Guide" [4, Cap. 4]. I Master Ip producono di norma numeri decimali a punto fisso rappresentati con N bit (la rappresentazione floating point è comunque possibile). L' interfaccia necessità però dei pacchetti di dati le cui dimensioni siano multipli di un byte per rispettare il protocollo e per questo motivo i Master IP devono anche operare del "bit stuffing". Ai numeri prodotti vengono quindi aggiunti dei bit per raggiungere una dimensione pari al prossimo multiplo di otto bit per essere successivamente considerato un pacchetto. Se i numeri sono di tipo unsigned, ossia senza un segno, il bit stuffing si opera con soli zeri mentre se i numeri sono di tipo signed, ossia caratterizzati da un segno, il bit stuffing aggiunge dei bit che rispecchino il bit di segno. A seconda della grandezza di TDATA è inoltre possibile inviare più pacchetti di

dati in parallelo. Un primo esempio di applicazione del protocollo consiste nella comunicazione di numeri complessi. In questo esempio parte reale e complessa sono composte ciascuna da 12 bit. In figura 11 possiamo vedere come un pacchetto di dati è composto da un byte per la parte reale e un byte per la parte complessa (i bit di stuffing sono stati omessi nella rappresentazione), con il segnale TLAST alto nel ciclo di clock in cui viene trasferita la parte immaginaria per indicare la fine del pacchetto.

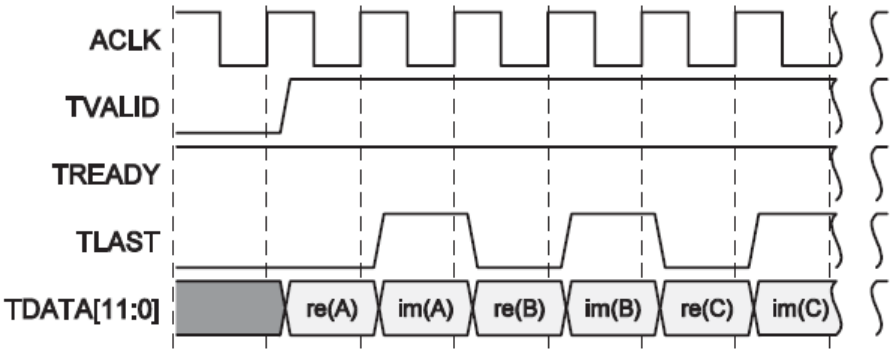

Fig.11 Numeri complessi trasmessi con AXI-Stream.

Se il bus è largo almeno 32 bit è possibile trasportare parte reale e parte immaginaria nello stesso ciclo di clock e TLAST rimane sempre alto per indicare che a ogni trasferimento corrisponde un pacchetto, come in figura 12.

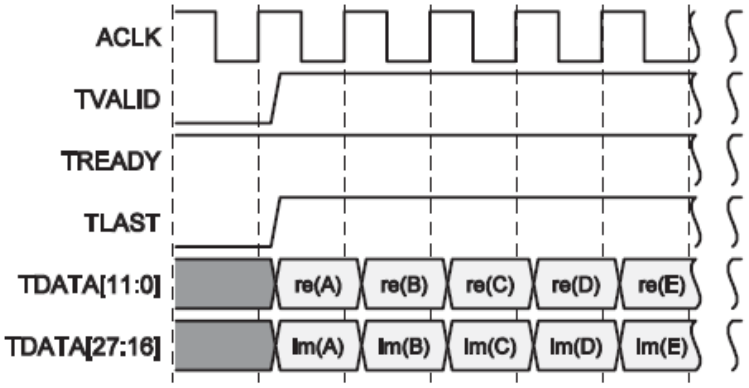

Fig.12 Trasmissione in parallelo di parte reale e immaginaria di un numero complesso

Il canale di trasmissione può essere usato per trasportare non solo dei singoli numeri complessi, ma anche dei vettori di numeri complessi. Nelle figure 13,14,15 e 16 possiamo osservare diverse soluzioni per il trasporto di pacchetti contenenti vettori composti da 4 numeri complessi con un graduale aumento di parallelismo dei byte all' aumentare della larghezza di TDATA. In questo caso un pacchetto contiene un vettore di complessi e il segnale TLAST, che viene posto al valore alto durante il ciclo di clock in cui la parte immaginaria dell'ultima componente del vettore viene trasmessa, indica la fine di un pacchetto e di conseguenza anche la fine di un vettore.

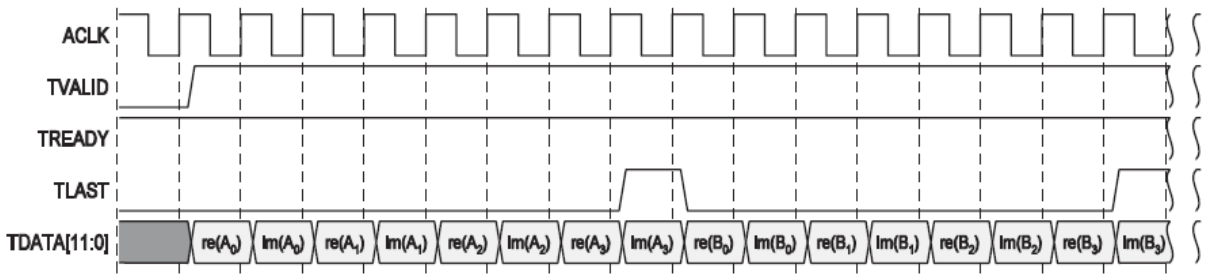

Fig.13 Bus da 16 bit, tutti i byte di un pacchetto vengono trasferiti in serie.

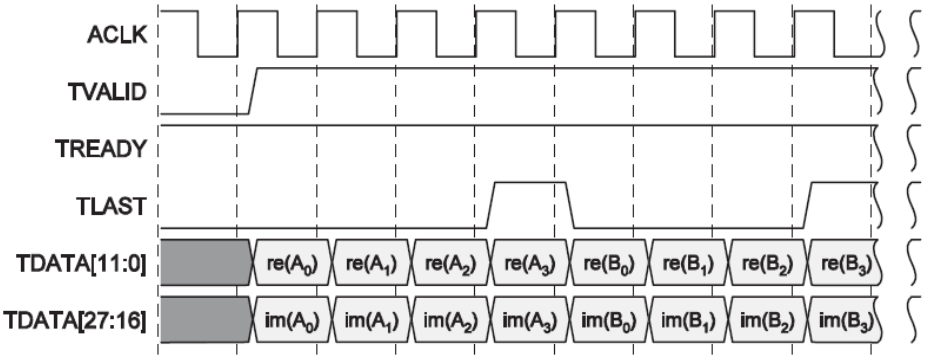

Fig.14 Bus da 32 bit, con queste dimensioni è possibile inviare ad ogni trasferimento una componente del vettore trasmettendo parte reale e parte immaginaria contemporaneamente.

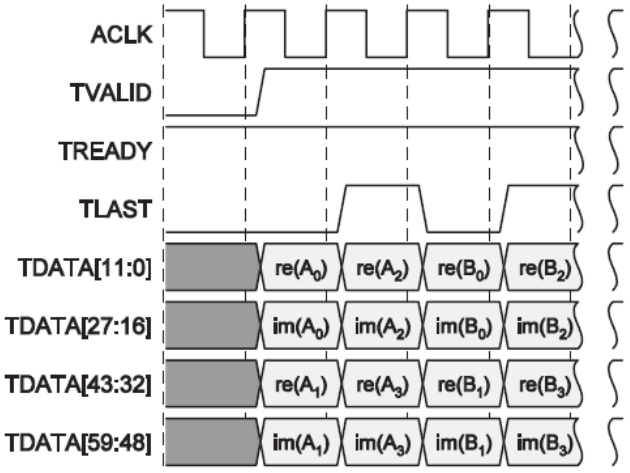

Fig.15 Bus da 64 bit, in questo caso è possibile trasmettere due numeri complessi in parallelo.

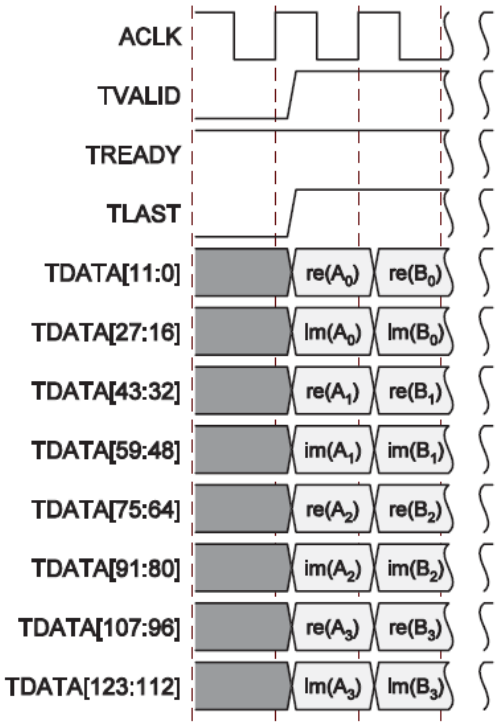

Fig.16 Bus da 128 bit, in questo caso un intero vettore viene trasmesso in un singolo ciclo di clock. Di conseguenza TLAST rimane sempre a valore alto.

Infine, AXI-Stream viene adattato anche al trasporto di informazioni video. Nel Video IP di Xilinx i segnali TKEEP e TSTRB non sono utilizzati perché in quasi tutti i casi i pacchetti contengono informazioni valide e di conseguenza vengono posti di default a valore alto. Anche TID e TDEST non vengono utilizzati perché l'IP dedica un'interfaccia distinta per ciascuna comunicazione. Di conseguenza TID e TDEST vengono impostati al loro valore di default che prevede ogni bit ha valore 0. TUSER[0] viene rinominato come segnale SOF (*Start Of Frame*) e indica l'inizio di ogni frame di cui il video trasmesso è composto indicando che durante il trasferimento in cui è alto TDATA contiene le informazioni riguardanti il primo pixel del frame. La restante parte di TUSER non viene utilizzata e di conseguenza nemmeno propagata fino allo slave. Il segnale TLAST viene rinominato come EOL (*End Of Line*). Questo segnale deve rimanere alzato per l'interita di un trasferimento che trasporta informazioni dell'ultimo pixel di una riga. In un frame ci si deve aspettare quindi EOL a valore alto un numero di volte pari al numero di righe che compongono il frame. In figura 17 è possibile vedere la struttura del canale di comunicazione e l'evoluzione dei segnali SOF e EOL.

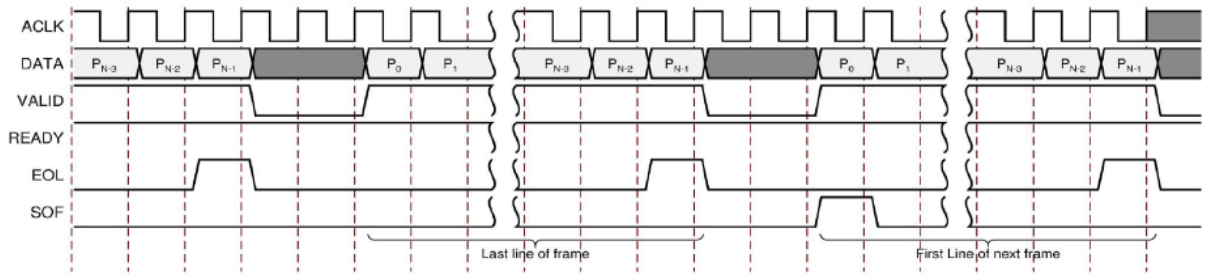

Fig. 17 Canale di comunicazione nel design di Xilinx Video IP.

5) CONCLUSIONI

Nella seguente tabella, ricavata da "*Zynq-7000 SoC (Z-7030, Z-7035,Z-7045, and Z-7100): DC and AC Switching Charateristics Data Sheet"* [6], possiamo valutare la performance della comunicazione AXI che raggiunge la frequenza massima a 250MH, la frequenza massima tra tutti i diversi tipi di interconnessioni presenti nelle board.

Symbol	Description	Min	Max	Units
$F_{EMIOGEMCLK}$	EMIO gigabit Ethernet controller maximum frequency	–	125	MHz
$F_{EMIOSDCLK}$	EMIO SD controller maximum frequency	–	25	MHz
$F_{EMIOSPICLK}$	EMIO SPI controller maximum frequency	–	25	MHz
$F_{EMIOJTAGCLK}$	EMIO JTAG controller maximum frequency	–	20	MHz
$F_{EMIOTRACECLK}$	EMIO trace controller maximum frequency	–	125	MHz
F_{FTMCLK}	Fabric trace monitor maximum frequency	–	125	MHz
$F_{EMIODMACLK}$	DMA maximum frequency	–	100	MHz
F_{AXI_MAX}	Maximum AXI interface performance	–	250	MHz

Tab.3 Frequenze delle varie interconnessioni tra logica programmabile e il processore ARM presenti nelle board della serie ZYNQ-7000.

Con un totale di due interfacce AXI master e due interfacce AXI slave generiche da 32bit, 4 interfacce AXI slave configurabili e bufferizzate da 32 o 64 bit collegate alle memorie OCM e DDR e infine un interfaccia AXI slave da 64bit collegata alla memoria CPU [6], le board della serie ZYNQ-7000 sono in grado di operare in moltissimo contesti tra cui l'assistenza la guidatore nell'ambito automotive, le telecamere per i servizi broadcast e le telecamere intelligenti, applicazioni industriali come il controllo dei macchinari o networking, applicazioni mediche di diagnosi e di immagine, stampanti multifunzione e applicazioni video e di visione notturna [7].

6) BIBLIOGRAFIA

[1] "*ZYNQ 7000 SoC*", Xilinx, inc.
https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html
Visualizzato in data 15/07/2023

[2] "*AMBA AXI and ACE Protocol Specification*", ARM Holdings Plc. (online).
https://documentation-service.arm.com/static/602a9df190ee6824a1e02b98?token=
Visualizzato in data 15/07/2023

[3] "*AMBA 4 AXI4-Stream Protocol specificacion*", ARM Holdings Plc. (online).
https://documentation-service.arm.com/static/642583d7314e245d086bc8c9?token=
Visualizzato in data 27/08/2023

[4] "*Vivado Design Suite: AXI Reference Guide",* Xilinx, inc. https://docs.xilinx.com/v/u/en-US/ug1037-vivado-axi-reference-guide Visualizzato in data 15/07/2023

[5] "*System Design"*, Opal Kelly (online) all'indirizzo https://docs.opalkelly.com/fpsdk/system-design/#:~:text=FrontPanel%20introduces%20the%20concept%20of,want%20to%20observe%20in%20FrontPanel. Visualizzato in data 10/09/2023

[6] "*Zynq-7000 SoC (Z-7030, Z-7035,Z-7045, and Z-7100): DC and AC Switching Charateristics Data Sheet*", Xilinx, inc. https://docs.xilinx.com/v/u/en-US/ds191-XC7Z030-XC7Z045-data-sheet Visualizzato in data 22/09/2023

[7] "*Zynq-7000 SoC Data Sheet: Overview*" Xilinx, inc.
https://docs.xilinx.com/v/u/en-US/ds190-Zynq- 7000-Overview
Visualizzato in data 22/09/2023