Elektrotehnički fakultet
Univerzitet u Banjoj Luci

UPOZNAVANJE SA AVALON-MM INTERFEJSOM

1. Uvod

Avalon je standardni interfejs protocol korišten u Altera (sada dio Intel-a) *FPGA* sistemima za komunikaciju između različitih modula ili podsistema unutar *FPGA*. Postoji čitava parodica Avalon interfejsa, i obično su pogodni za strimovanje podataka visokim brzinama, čitanje ili upisivanje registara i memorije te kontrolu tzv. *off-chip* uređaja. Sve već postojeće komponente u okviru *Altera FPGA* sistema podržavaju ove interfejse, ali također i u sve komponente koje mi kreiramo, moguće je ugraditi Avalon interfejse i tako proširiti interoperabilnost dizajna.

Kao što je već rečeno postoji čitava porodica Avalon interfejsa, ali u pogledu našeg projektnog zadatka nama je koristan Avalon-MM (*Memory-Mapped*) interfejs, pa ćemo se u nastavku pozabaviti njime.

Zadatak ovog teksta je upoznavanje sa osnovama Avalon-MM interfejsa, kojeg će kasnije kroz rad na projektnom zadatku biti potrebno implementirati. Treba istražiti šta je Avalon-MM interfejs, kako se koristi, koje su ključne osobine i napraviti uvod u njegovu kasniju implementaciju.

2. Avalon-MM interfejs

Avalon-MM je adresno bazirani čitaj/piši interfejs iz porodice Avalon interfejsa, i tipičan je za tzv. *Host/Agent* konekcije (nešto kao *Master/Slave* princip). U nedostatku dobrog prevoda u ovom tekstu će se koristiti termini iz engleskog jezika host i agent za komponente koje učestvuju u Avalon-MM konekciji. Dakle ako postoje dva hardverska modula koji treba da komuniciraju preko Avalon-MM interfejsa jedan od njih će biti host, a drugi agent. Host obično inicira transakcije, a host odgovara na njih.

Moguće je koristiti Avalon-MM interfejs za implementiranje interfejsa za čitanje i pisanje za host i agent komponente. Slijede neki od primjera komponenti koje tipično uključuju MM interfejse:

- Mikroprocesori
- Memorije
- UART
- DMA
- Tajmeri

Npr. SRAM interfejsi koji imaju transfere za čitanje i pisanje za fiksnim trajanjem imaju jednostavni Avalon-MM interfejs.

Prvi korak je dakle dizajniranje komponenta koje komuniciraju, a zatim želimo da ih povežemo preko Avalon-MM interfejsa. Ovo povezivanje uključuje povezivanje određenih signala modula, a koji su u skladu sa specifikacijam interfejsa.

2.1 Signali

Avalon-MM komponente tipično uključuju samo one signale koji su potrebni za logiku komponente koja se projektuje. Prema tome za potrebe našeg rada biće potrebno pravilno odabrati signale koji su neophodni za avalon komunikaciju i uključiti ih u našoj implementaciji logike.

Postoji skup signala koje Avalon-MM host i agent portovi dozvoljavaju. Ne postoji signal koji je uvijek neophodan, ali može se reći da je minimalan zahtjev za *read-only* interfejse u pogledu signala sadržanje *readdata* signala, dok za *write-only* interfejse su to *writedata* i *write* signali.

Sljedeća tabela se sastoji od **osnovnih** signala Avalon-MM interfejsa:

Signal	Širina	Smjer	Opis
address	1 - 64	Host → Agent	Host: address predstavlja bajt adresu.Vrijednost address signala mora da se poravna sa širinom podataka. Da bi upisali određeni bajtovi u unutar riječi podataka potrebno je koristiti byteenable signal. Agent: Podrazumijevano, međuveza (ono preko čega su spojeni i host i agent) pretvara bajt adresu u adresu riječi u agentovom adresnom prostoru. Iz perspektive agenta svaki read pristup je pristup čitanja jedne riječi podataka. Npr. address = 0 selektuje prvu riječ agenta, address = 1 drugu riječ itd
byteenable **byteenable_n**	2,4,8,16,32,64,128	Host → Agent	Omogućava jednu ili više konkretnih bajt linija tokom prenosa na interfejsima širine veće od 8 bita. Svaki bit u byteenable odgovara jednom bajtu u writedata ili readdata signalu. Tokom upisa označava koji bajtovi se upisuju. Drugi bajtovi su ignorisani od strane agenta. Tokom čitanja označava koje bajtove host čita.
read **read_n**	1	Host → Agent	Ako je postavljen označava prenos čitanja. Ako postoji ovaj signal mora postojati i readdata.
read_data	8,16, 32, 64, 128, 256, 512, 1024	Agent → Host	Potreban za interfejse koji podržavaju čitanje. Signal kao odgovor na read prenos.
response	2	Agent → Host	Opcioni signal koji nosi status odgovora na write ili read prenos. (OKAY; RESERVED; SLVERR;DECODEERROR)
write **write_n**	1	Host → Agent	Postavljen u slučaju write prenosa. Ako postoji ovaj signal mora da postoji i writedata.

write_data	8, 16, 32, 64, 128, 256, 512, 1024	Host → Agent	Podaci za write transfer. Širina mora biti ista kao širina readdata ako su oba signala prisutna. Potreban za interfejse koji podržavaju upis.

Tabela 1 – osnovni signali Avalon-MM interfejsa

Moguće je preko interfejsa imati više hostova na jednog agenta, pa se za razrješenje konflikta mogu koristiti *lock* signali.A također je moguće i da agent poštalje hostu signal kojim kaže da ne može odmah da odgovori: *waitrequest/waitrequest_n.*

Zatim signal kojim se daje potvrda ispravnosti pročitanih podataka *readdatavalid* i *readdatavalid_n,* te signal *writeresponsevalid* kojim se daje potvrda o ispravnosti podataka poslanih u *writeresponse.*

Interfejs podržava i tzv. *burst* mod, odnosno više transfera odjednom. I za kontrolu ovog moda imamo signale *burstcount,* te *beginbursttransfer*. Više informacija o tome na [1]. Za sada ove signale nećemo detaljnije razmatrati, jer iz konteksta zadatka kojeg radimo ovi signali nisu relevantni.

2.2 Svojstva (properties)

Opis ponašanja Avalon-MM interfejsa dat je kroz njegova svojstva odnosno *properties.* Tu postoji niz svojstava kao što su jedinica za adrese (tipično bajt), zatim neki parametri *burst* moda kao što su dužina *burst*-a, da li se radi uvijek *burst* maksimalne dužine. Zatim imamo maksimalan broj čekajućih transkacija za čitanje ili upis koje agent može prihvatiti. Dalje postoje neki parametri koji se tiču vremena između transkacija, postavljanja pojedinih signala i slično. U suštini dosta stvari se može podesiti i prilagoditi konkretnoj komunikaciji koja se implementira.

Treba reći da je Avalon-MM interfejs sinhroni interfejs i da radi u sinhronizaciji sa povezanim takt signalom. Signali mogu biti kombinacioni ako su pogonjeni sa izlaza registara koji su sinhroni sa takt signalom.

2.3 Transferi

Transfer ili prenos o kome smo govorili u prethodnom tekstu je zapravo operacija čitanja ili pisanja jednog ili više simbola (bajta) podataka. Transfer se dešava između Avalon-MM interfejsa i međuveze (medijum za prenos). Obično je potrebno jedan ili dva ciklusa za transfer da se završi. I host i agent učestvuju u transferu tako da host inicira transfer, a host odgovara na njega, kao što smo već i

rekli. U [1] postoji detaljnije objašnjenje nekih tipičnih transfera čitanja i upisa uz postojanja *waitrequest* signala i njegovog kontrolisanja od strane agenta, kao i vremenski dijagrami vezani uz ove prenose.

2.4 Pipelined transferi

Postoje i *pipeline* transferi koji mogu povećati throughput za sinhrone agente kojima treba nekoliko ciklusa da vrate zahtijevane podatke. Ukoliko se koristi *pipelined* transfer, prije odgovora na stari transfer, moguće je u paraleli inicirati novi transfer. Postoji adresna faza i faza podataka. Host inicira transfer u fazi podataka i prelazi u fazu podataka, a za to vrijeme može se inicirati novi transfer u fazi podataka. Detaljnije o ovome također se može naći u [1] na stranama 27 i dalje.

2.5 Burst transferi

Koristi se kada želimo da izvršimo više transfera odjednom, a ne da tretiramo svaku riječ podataka posebno. Prilikom *burst* transfera potrebno je koristiti odgovarajuće signale i podesiti parametre (*properties*) interfejsa. [1] strana 30.

2.6 Adresiranje agenta

Za početak bitno je reći adresa koju host inicira mora biti poravnata tako da bude umnožak širine riječi podataka hosta u simbolima (najčešće simbol je bajt).

Ako imamo host-agent par koji imaju različite širine podataka, onda se mora iskoristiti dinamičko podešavanje magistrale prilikom prenosa. Ako je riječ podataka kod hosta šira nego kod agenta, onda se riječi podataka hosta moraju mapirati na više lokacija u agentu. Npr. 32-bitna operacija host čitanja od 16-bitnog agenta rezultuje u dva čitanja sa sa strane agenta, tj. pročitane su dvije lokacije agenta sa dvije uzastopne adrese. Sa druge strane ako je host 'uži' od agenta, onda međuveza upravlja agentovim linijama podataka. Tokom čitanja od strane hosta, međuveza prikazuje odgovarajuće linije podataka agenta, užem hostu. A tokom upisivanja od strane hosta, međuveza automatski postavlja *byteenable* signale da upisuju podatke samo u odgovarajuće bajt linije agenta.

Agenti moraju imati širinu podatka koja je umnožak vrijednosti 8, 16, 32, 64, 128, 256, 512, 1024 bita. Sljedeća tabela prikazuje poravnanje podataka agenta za različite širine, pri operaciji pristupa od strane 32-bitnog agenta. U tabeli *OFFSET [N]* se odnosi na ofset riječi podataka unutar adresnog prostora agenta.

Host Byte Address (1)	Access	32-Bit Host Data		
		When Accessing an 8-Bit Agent Interface	When Accessing a 16-Bit Agent Interface	When Accessing a 64-Bit Agent Interface
0x00	1	OFFSET[0]$_{7..0}$	OFFSET[0]$_{15..0}$ (2)	OFFSET[0]$_{31..0}$
	2	OFFSET[1]$_{7..0}$	OFFSET[1]$_{15..0}$	—
	3	OFFSET[2]$_{7..0}$	—	—
	4	OFFSET[3]$_{7..0}$	—	—
0x04	1	OFFSET[4]$_{7..0}$	OFFSET[2]$_{15..0}$	OFFSET[0]$_{63..32}$
	2	OFFSET[5]$_{7..0}$	OFFSET[3]$_{15..0}$	—
	3	OFFSET[6]$_{7..0}$	—	—
	4	OFFSET[7]$_{7..0}$	—	—
0x08	1	OFFSET[8]$_{7..0}$	OFFSET[4]$_{15..0}$	OFFSET[1]$_{31..0}$
	2	OFFSET[9]$_{7..0}$	OFFSET[5]$_{15..0}$	—

Više informacija može se naći na https://cdrdv2-public.intel.com/667068/mnl_avalon_spec-683091-667068.pdf