Vorlesung Architekturen und Entwurf von Rechnersystemen

Lösungsvorschlag

Prof. Andreas Koch, Yannick Lavan, Johannes Wirth, Mihaela Damian

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2022/2023
Theorieblatt 4

4.1 AXI4

In dieser Aufgabe beschäftigen Sie sich noch mal genauer mit den verschiedenen AXI4 Varianten. Sie können für die Abhängigkeiten der Handshake-Signale die AXI Spezifikation[1] zur Hilfe nehmen.

4.1.1 AXI4 vs. AXI4-Lite vs. AXI4-Stream

Erläutern Sie die wesentlichen Unterschiede der Protokolle AXI4, AXI4-Lite und AXI4-Stream. Gehen Sie bei AXI4 und AXI4-Lite auch auf die Unterschiede aus Hardware-Sicht ein.

Lösungsvorschlag:

AXI4 unterstützt memory-mapped I/O und Bursts. Weiterhin hat AXI4 zusätzliche Signale für das Ende eines Transfers, Transfer-ID, User, Quality of Service und mehr. AXI4-Lite unterstützt memory-mapped I/O aber keine Bursts. Entsprechend entfällt das WLAST Signal. ID, User, QoS und andere optionale AXI4-Signale sind nicht vorhanden. AXI4-Lite bietet somit eine minimale AXI Transfer Infrastruktur. AIX4-Stream arbeitet unidirektional, wodurch zwei AXI4-Stream Interfaces für bidirektionale Kommunikation nötig sind. Weiterhin unterstützt AXI4-Stream kein memory-mapped I/O. Durch die zusätzlichen Signale und Bursts benötigt AXI4 mehr Chipfläche als AXI4-Lite oder AXI4-Stream. Für bidirektionales memory-mapped I/O liefert AXI4 aber die bessere Performanz als AXI4-Lite, da nicht ständig neue Requests für benachbarte Daten gesendet werden müssen.

4.1.2 AXI vs. Bluespec

Wie unterscheiden sich Bluespecs RDY/EN-Handshake von AXIs Valid/Ready-Handshake?
Hinweis: Machen Sie sich klar warum (* always_enabled *) in Bluespec (* always_ready *) impliziert.

Lösungsvorschlag:

In Bluespec muss erst das entsprechende RDY-Signal einer Methode 1 sein, damit anschließend EN gesetzt werden kann. Daher impliziert auch always_enabled das Attribut always_ready. Bei AXI ist die Reihenfolge der Handshake-Signale egal. Sobald beide Signale bei einer steigenden Taktflanke 1 sind findet ein Datenaustausch (Request auf Addresskanal, Daten auf Datenkanal) statt. Weiterhin gibt es Empfehlungen, welche Signale nach Möglichkeit immer 1 sein sollen.

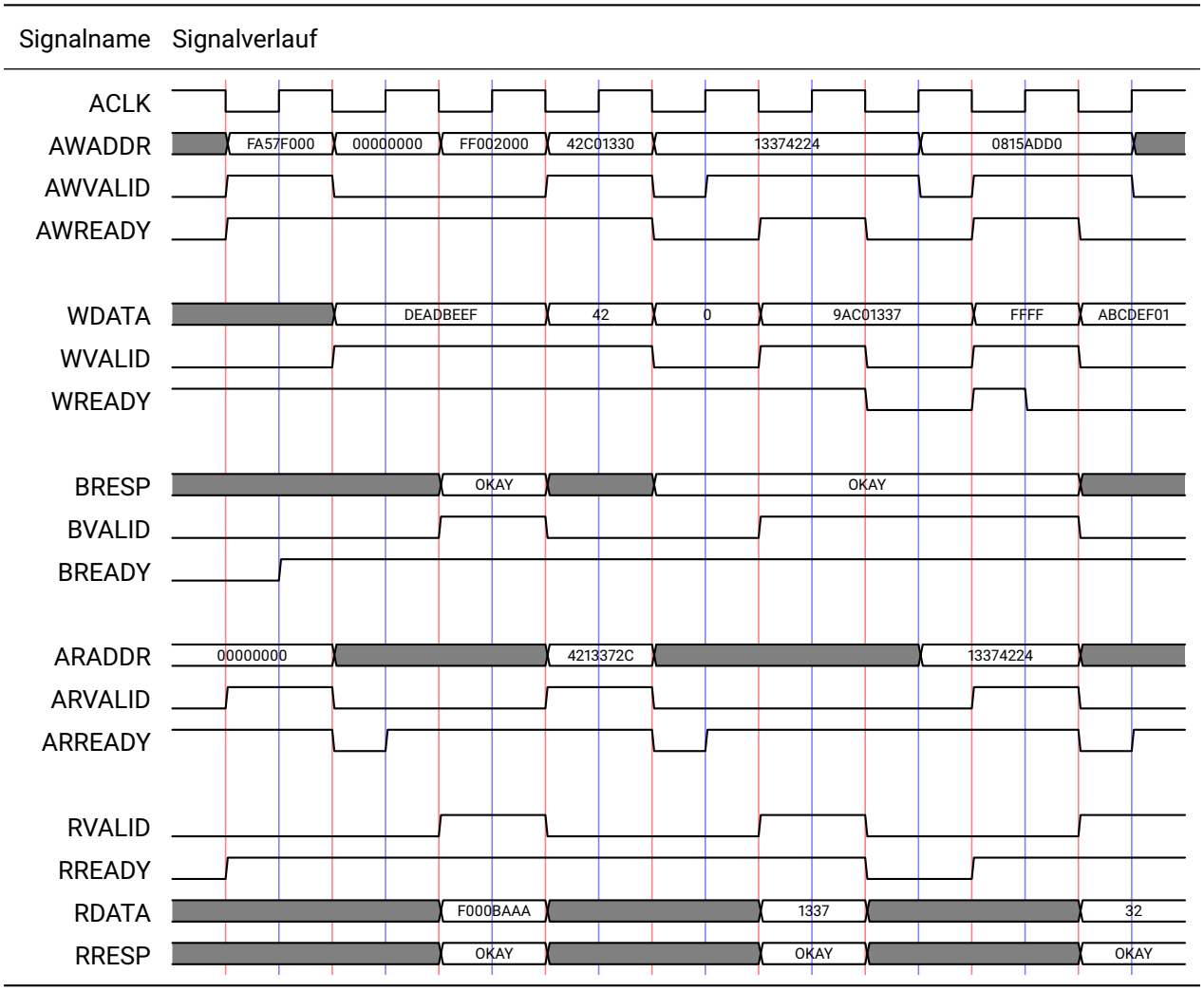

Abbildung 1: AXI4-Lite Transfers

4.1.3 Transfers

Sie können für diese Aufgabe die bereitgestellten Tabellen benutzen.
a) Nehmen Sie an, dass das Protokoll AXI4-Lite für die Transfers in Abbildung 1 verwendet wurde. Geben Sie den Zustand des Slave-Moduls nach Durchführung aller Transfers an. Die Datenworte und Adressen sind in hexadezimaler Notation gegeben.

Lösungsvorschlag:

Es finden zwei Transfers mit den Daten 0xDEADBEEF gefolgt von einem Transfer mit 0x42 statt. 0x42 wird durch das Slave-Modul selbst überschrieben, wodurch an Adresse 0x13374224 später 0x32 zurückgegeben wird. Die Handshake Signale werden jeweils nur zur steigenden Taktflanke (blau markiert) überprüft.

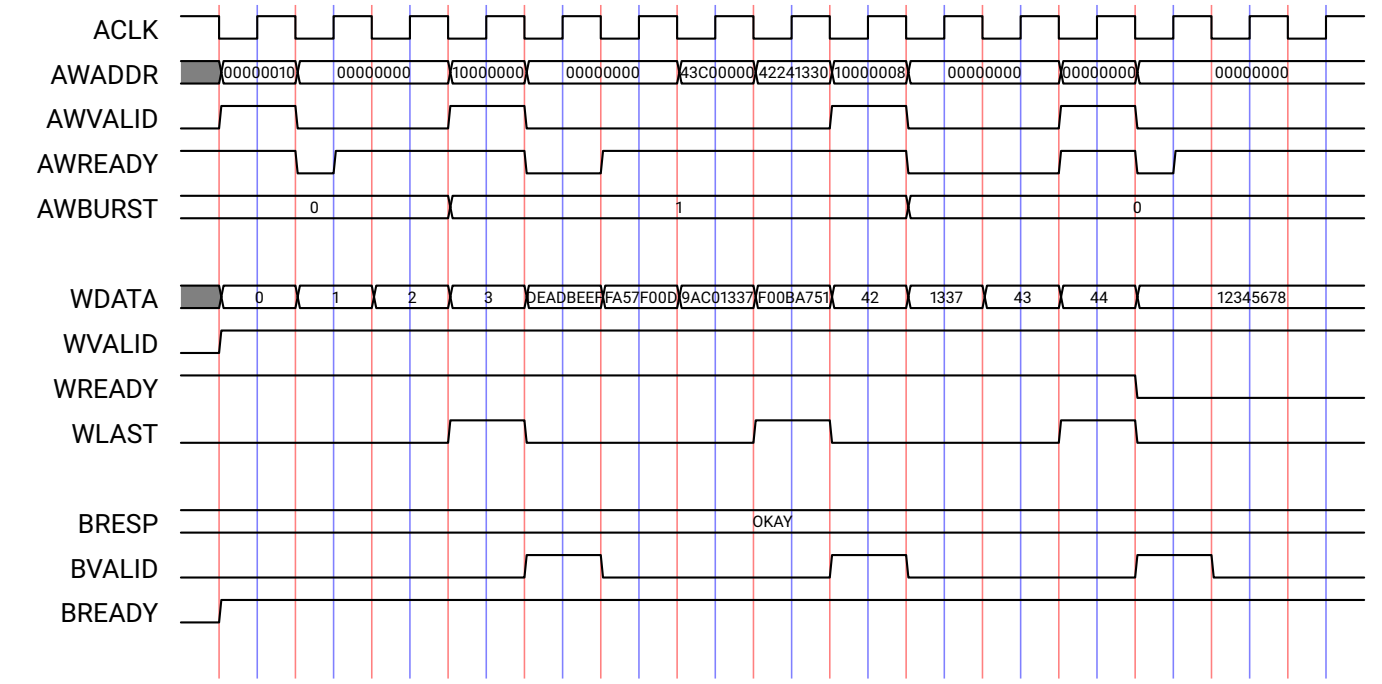

Abbildung 2: AXI4-Transfers auf dem Write-Kanal

Adresse	Wert (hexadezimal)
0x00000000	0xF000BAAA
0x0815ADD0	0x9AC01337
0x13374224	0x00000032
0x4213372C	0x00001337
0x42C01330	0xDEADBEEF
0xFA57F000	0xDEADBEEF

b) In Abbildung 2 wurde AXI4 verwendet. Für uns uninteressante Signale wurden im Diagramm ausgelassen. Geben Sie auch hier den Zustand des Slave-Speichers nach Beendigung aller Transfers an. Alle Werte sind hexadezimal angegeben. Es gilt außerdem AxLEN=3, AxSIZE=2. Nehmen Sie weiterhin an, dass immer der letzte geschriebene Wert an der entsprechenden Speicher-Adresse steht.

Lösungsvorschlag:

Im ersten Transfer wird der Burst-Modus FIXED verwendet, alle Beats schreiben also auf die gleiche Adresse. Darum steht an Adresse 0x10 der Wert 3. Die Werte an den Adressen 0x10000008 und 0x1000000C werden durch den dritten Transfer überschrieben.

Adresse	Wert (hexadezimal)
0x00000010	0x3
0x10000000	0xDEADBEEF
0x10000004	0xFA57F00D
0x10000008	0x42
0x1000000C	0x1337
0x10000010	0x43
0x10000014	0x44

4.2 Rekonfigurierbares Rechnen und TaPaSCo

4.2.1 Task Parallel System Composer (TaPaSCo)

a) Erläutern Sie die grundliegende Idee hinter TaPaSCo. Welche Probleme werden von TaPaSCo adressiert?
b) Was ist eine DSE und wofür wird sie eingesetzt?
c) Gegeben sei die aus der Vorlesung bekannte Address Map in Abbildung 3. An welche Adresse müssen Sie schreiben, wenn sie das Argument 1 Register von target_ip_00_000 setzen möchten? Sie möchten einen 32-Bit Wert aus dem Return Value Register lesen und den Interrupt bestätigen, nachdem target_ip_00_011 einen Interrupt gesetzt hat. Von welcher Adresse müssen Sie lesen, an welche Adresse müssen Sie welchen Wert schreiben?

Lösungsvorschlag:

a) TaPaSCo dient der Komposition und Evaluation von parallelen (und homogenen) Systemarchitekturen auf Field-Programmable Gate Arrays (FPGAs). Hierbei erleichtert TaPaSCo durch seine Platform/Architektur-Unterteilung die Wiederverwertbarkeit von Designs, sodass mehr Zeit in die Lösung des eigentlichen Problems investiert werden kann, als in die Integration der Hardware auf unterschiedlichen Plattformen. Weiterhin erleichtert TaPaSCo durch den DSE-Job die Evaluation von Kompositionen.
b) DSE steht für Design Space Exploration. Hierbei kann eine Komposition bezüglich des Job-Durchsatzes evaluiert werden. Durch eine DSE kann man sowohl die maximale Frequenz eines IP-Cores, als auch die maximale Anzahl parallel platzierbarer IP-Cores feststellen. Die DSE von TaPaSCo ist sinnvoll, da sie ohne viel Aufwand für den Nutzer viele verschiedene Plattformen evaluieren kann.
c) Man muss an Adresse 0x43C00020 schreiben. Man muss von Adresse 0x43CB0010 lesen und eine 1 an Adresse 0x43CB000C schreiben.

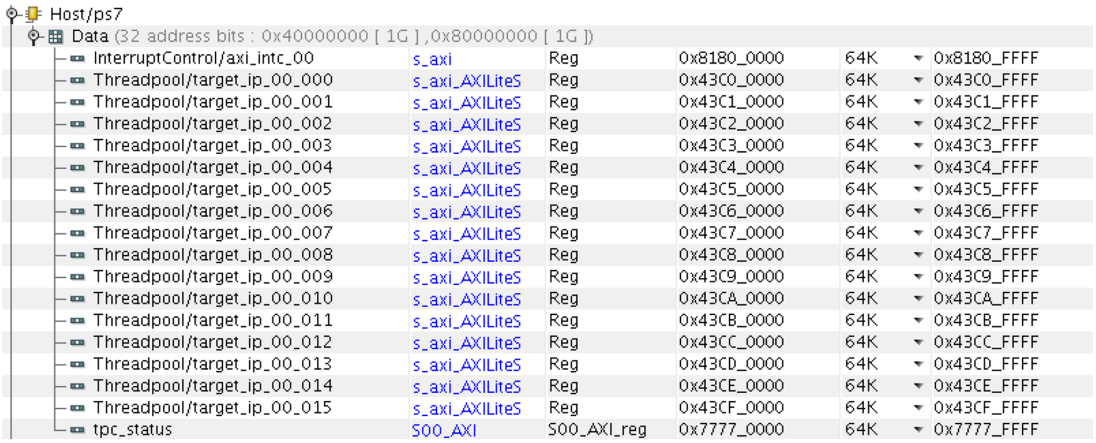

Abbildung 3: Adressmap aus der TaPaSCo Vorlesung

4.2.2 Rekonfigurierbares Rechnen

a) Was ist der Unterschied zwischen einer Lookup Table (LUT) und einem Configurable Logic Block (CLB)?
b) Setzen Sie die Funktion $Y = \bar{C}D + A\bar{B}$ in der untenstehenden LUT um, wobei $+$ für das logische ODER steht. Setzen Sie hierfür $S_1 = 0$. Bilden Sie Y auf S_0 ab.

Lösungsvorschlag:

a) Eine LUT ist Teil eines CLBs. Sie speichert nur direkte logische Zusammenhänge zwischen Eingangs- und Ausgangssignalen. Ein CLB enthält neben LUTs noch weitere Komponenten, wie zum Beispiel Multiplexer.
b) Siehe Tabelle:

A	**B**	**C**	**D**	S_1	S_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	0	0
1	1	1	1	0	0

Wichtige Abkürzungen

CLB Configurable Logic Block

DMA Direct Memory Access

DSP Digital Signal Processor

FPGA Field-Programmable Gate Array

FPU Floating-Point Unit

LUT Lookup Table

PL Programmable Logic

PS Processing System

RAM Random-Access Memory

ROM Read-Only Memory

rSoC Reconfigurable System-on-Chip

TaPaSCo Task Parallel System Composer

Literatur

[1] http://www.gstitt.ece.ufl.edu/courses/fall15/eel4720_5721/labs/refs/AXI4_specification.pdf.