

## Milestone 4: AXI und ILA

### Lernziele

- Anwendung des integrierten Logicanalyzers
- Vertiefung des Verständnisses des AXI-Protokolls
- Analyse/Verständnis der Performance von DDR-Speichern
- Wiederholung der Softwareentwicklung mit Vitis

### Aufgabenstellung

Ein Thema dieses Milestones ist die Analyse der zeitlichen Eigenschaften des Zugriffs auf den externen Speicher. Darüber hinaus soll die Anwendung des integrierten Logicanalyzers (ILA) verdeutlicht werden, da dieser eine wichtige Möglichkeit zum Debuggen von IP darstellt und auch bei eigenen Projekten eine wertvolle Hilfe sein kann.

Für die Durchführung des Versuchs wird ein vorgegebenes IP verwendet, welches AXI-Lesezugriffe erzeugt. Dieses IP wird jeweils an die vier High-Performance AXI-Slave-Ports angeschlossen (vgl. Systemübersicht im Anhang).

Mithilfe des ILA werden die DDR-Parameter Zugriffslatenz und Bandbreite für verschiedene Betriebsmodi abgeschätzt.

Bei der Bearbeitung der Aufgabe wird kein eigenes IP erstellt. Das System, anhand dessen die Messungen durchgeführt werden sollen, kann aus einem vorgegebenen TCL-Skript erstellt werden.

### AXI-Read-Generator

Für die Erzeugung von Lesezugriffen wird das IP *axi\_read\_generator* verwendet. Es besitzt eine AXI-Schnittstelle und eine AXIL-Schnittstelle. Letztere dient der Konfiguration des IPs, so dass die Wahl unterschiedlicher Parameter per Software erfolgen kann und somit nur ein Syntheselauf erforderlich ist.

Das IP generiert lediglich Leseanforderungen. Die gelesenen Daten werden nicht ausgewertet.

Unter anderem können die folgenden Parameter sowohl im IP-Integrator als auch per Software konfiguriert werden:

- Startadresse der Burst-Leseanforderungen
- Burstlänge
- Auswahl der "Pausezeiten" zwischen der Ausgabe von Leseanforderungen (in Taktzyklen)
- Start/Stop der Leseanforderungen
- Auswahl, ob neue Lesezugriffe bereits angefordert werden, bevor vorangegangene Lesezugriffe abgeschlossen sind (Pipelining)

Die folgenden Parameter können zusätzlich im IP-Integrator konfiguriert werden:

- Datenwortbreite: 32, 64, 128 bit. (Für diesen Milestone wird eine Daten-Wortbreite von 64 bit verwendet)
- Wortbreite der Transaction-ID: 1-6 bit. (Für diesen Milestone wird eine ID-Wortbreite von 4 bit verwendet)

Die Registerbelegung des IPs ist im Anhang dargestellt.

### Durchführung

1. Erstellen Sie ein neues Vivado-Projekt und fügen Sie das Verzeichnis, in dem sich das IP *axi\_read\_generator* befindet, als IP-Repository hinzu.
2. Führen Sie das TCL-Skript *milestone4.tcl* aus (Tools -> Run Tcl Script...). Sie erhalten so das Blockdesign des Systems.

3. Aktivieren Sie den Debug-Modus für die AXI-Full-Verbindungen (nicht für die AXI-Lite-Verbindungen) für Speicherzugriffe zwischen IP und PS (vgl. Systemübersicht im Anhang). Lassen Sie die Verdrahtung des Logicanalyzers automatisch von Vivado vornehmen.  
Sie können die Speichertiefe (*Sample Data Depth*) des Logicanalyzers erhöhen (z.B. auf 2048), indem Sie den Logicanalyzer per Doppelklick konfigurieren.
4. Synthetisieren Sie das System nachdem Sie einen HDL-Wrapper erzeugt haben. Die Erzeugung des Bitfiles wird, je nach Leistungsfähigkeit des verwendeten Rechners, einige Zeit in Anspruch nehmen (10-20 Minuten).
5. Erstellen Sie in dem Projekt ein Unterverzeichnis *vitis* und exportieren Sie die Hardware in dieses Unterverzeichnis. Achten Sie hierbei darauf, dass Sie *Include Bitstream* ausgewählt haben. Starten Sie anschließend Vitis und legen Sie den Workspace im Unterverzeichnis *vitis* an.
6. Erstellen Sie in Vitis ein neues Application-Project auf Basis des Hello-World-Templates. Die für die Erstellung eines Plattform-Projects benötigte XSA-Datei finden Sie im Unterverzeichnis *vitis*.
7. Ersetzen Sie den Code in der C-Datei *hello.c* mit der vorgegebenen C-Datei. Kompilieren Sie die Software und starten Sie das Programm über den Debugger auf dem Zybo-Board.
8. Erst jetzt verbinden Sie sich mit Hilfe des Vivado Hardware-Managers mit dem Board ("Open Target"). Im Hardware-Manager wird Ihnen das ILA-Fenster angezeigt. Mit Hilfe des ILA können Sie nun die Signale auf den AXI-Verbindungen beobachten. *Hinweis: Das FPGA wird über Vitis programmiert. Das Programmieren über Vivado ist nicht erforderlich.*
9. Führen Sie die Messungen der Zugriffslatenz mit den in der nachfolgenden Tabelle aufgeführten Parametern durch. Eine Vorlage für ein Programm, mit dem Sie die Messungen durchführen können, finden Sie im Praktikumsordner. Als Latenz soll die Zeit (in Taktzyklen) gewählt werden, die zwischen der Bereitstellung einer Leseanforderung (steigende Flanke des Signals ARVALID) und dem Abschluss des Bursts (gekennzeichnet durch RLAST=1 & RVALID=1) vergeht. *Hinweis: Sie können hierfür zum Beispiel im ILA-Fenster "Marker" verwenden (Rechtsklick -> Add Marker).*

Beachten Sie, dass Vitis den Source-Code nach Änderungen nicht automatisch neu übersetzt. Wenn Sie dies möchten, müssen Sie unter Preferences "*Build (if required) before launching*" auswählen.

Die Basisadressen der Generator-IPs können Sie dem Vivado Address Editor entnehmen. Prüfen Sie, ob im vorgegebenen C-Code die korrekten Adressen verwendet werden.

*Hinweis: Es ist nicht erforderlich, die Programmierung der Programmable Logic aus Vivado vorzunehmen. Die Programmierung erfolgt zu Beginn des Debuggens aus Vitis heraus.*

*Hinweis: Wird im ILA-Fenster "Overflow" angezeigt, können Sie dies ignorieren. Es handelt sich um einen Bug in der Darstellung*

#Messung	Aktive HP-Ports	Burst Size	Pausezyklen	Pipelining
1	nur HP0	8	50	aus
2	nur HP0	4	50	aus
3	nur HP0	1	50	aus
4	nur HP0	4	5	aus
5	nur HP0	4	5	an
6	nur HP0	8	5	an
7	HP0, HP1, HP2, HP3	8	5	an

## Experimentiervorschlag für Interessierte

Die Messung mit Hilfe des Logicanalyzers ist natürlich mit einer gewissen Unsicherheit verbunden: Es ist theoretisch denkbar, dass Sie besonders positive Fälle mit kurzen Antwortzeiten messen. Um die Aussagekraft der Ergebnisse zu erhöhen, könnten Sie die minimale und maximale Latenz des Speicherzugriffs messen. Ebenso käme auch die Ermittlung der mittleren Antwortzeit in Betracht. Fügen Sie dem Generator-IP weitere Register hinzu, über die Sie diese Messwerte an die Software übergeben und mit Hilfe von `printf()` ausgeben können. Für die Ermittlung der Messwerte kann ein eigener VHDL-Prozess verwendet werden, der die relevanten AXI-Signale (ARVALID, ARREADY und RLAST) überwacht. Soll die Messung auch mit Pipelining funktionieren, wird es anspruchsvoll, da dann auch die "Transaction-IDs" (ARID und RID) berücksichtigt werden müssten.

Bedenken Sie hierbei, dass die Synchronisation des Bitstreams zwischen Vivado und Vitis nicht immer korrekt funktioniert und erst in späteren Tool-Versionen behoben wird. Als Workaround können Sie in der Debug-Konfiguration (Reiter: Target Setup, Eintrag: Bitstream File) direkt den von Vivado erzeugten Bitstream eintragen. Diesen finden Sie im Projektverzeichnis unter `<Projektname>.runs/impl_1/<Blockdesignname>_wrapper.bit`.

## Abgabe

Erstellen Sie ein formloses Dokument, in dem Sie Screenshots Ihrer Messungen und eine (z.B. tabellarische) Auflistung Ihrer Messergebnisse ablegen. Ermitteln Sie auch die zugehörige Bandbreite in Bytes je Taktzyklus. Die "Pausezeiten" sollen hierbei unberücksichtigt bleiben (= nur Zeiten zwischen Adressanfrage und Ende eines Bursts messen)

Vergleichen Sie Ihre Ergebnisse der Messung 7 mit der theoretisch maximal erreichbaren DDR-Bandbreite.

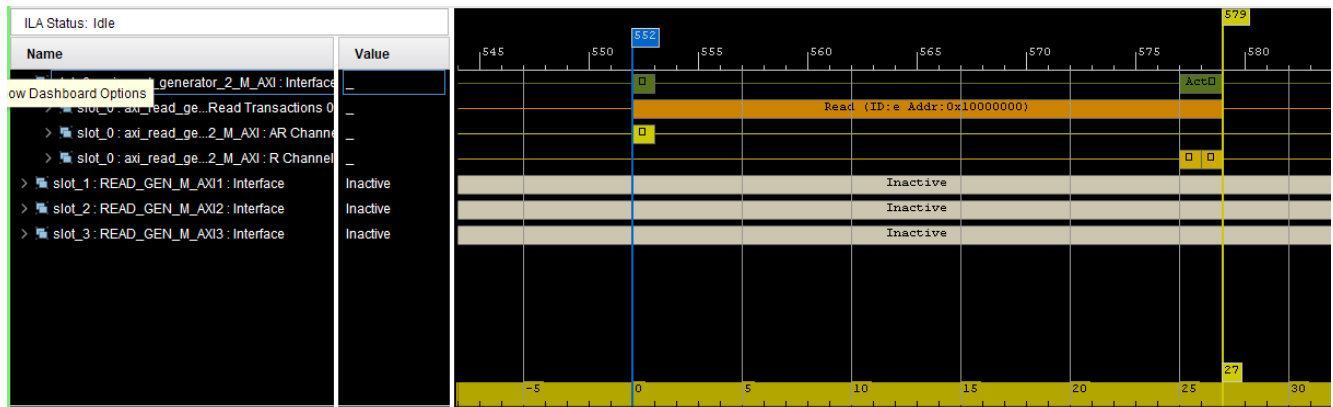
*Hinweis: Der DDR-Speicher wird mit 533 MHz betrieben und es wird eine Datenwortbreite von insgesamt 32 bit verwendet (2 DDR-Bausteine parallelgeschaltet). Die AXI-Interfaces werden mit 166.67 MHz betrieben und besitzen eine Datenwortbreite von 64 bit. Übrigens, laut Technical Reference Manual (TRM) ist die maximale Taktfrequenz der HP-Interfaces 150 MHz.*

Laden Sie das Dokument bitte im PDF-Format hoch.

*Hinweis: Die Tendenz der Ergebnisse sollte nicht überraschen (z.B. "mit Pipelining ist besser als ohne"), aber die absoluten Zahlen (auch im Vergleich untereinander) sind nicht unbedingt vorhersehbar.*

## Anhang

### Beispiel eines ILA-Screenshots für die Burstlänge 2



### Registerbelegung des IPs `axi_read_generator`

#### Control-Register (Basisadresse + 0)

31:16	15:12	11:8	7	6	5	4:0
<b>Pausezyklen</b> Anzahl der Pausezyklen zwischen Leseanforderungen	<i>reserviert</i>	<b>ARCACHE</b> Wert der ARCACHE- Leitungen des AXI- Interfaces	<b>Run</b> 0: IP inaktiv 1: IP aktiv	<b>Pipelining-Enable</b> 0: kein Pipelining 1: Pipelining aktiv	<b>Trigger</b> 0: Trigger immer 0 1: Trigger für ein Taktzyklus 1, wenn Leseadresse akzeptiert wurde	<b>Burstlänge</b> 1-16 (andere Werte reserviert)

#### Address-Register (Basisadresse + 4)

31:0
<b>Memory Address</b> Startadresse der Bursts

## Systemübersicht

Das Bild zeigt den Aufbau des verwendeten Systems. Die benötigten AXI-Interconnect-IPs befinden sich in dem Modul PS.

