

PCI-Express
Den schaffen



Umstieg

Beflügelt durch die Rückwärtskompatibilität der Hardware/Software-Schnittstelle befindet sich die Migration von PCI zu PCI-Express in vollem Gange. Um den Anschluss ans Neusystem-Geschäft nicht zu verlieren, müssen sich mittlerweile auch Hersteller von Nischenlösungen mit dieser Thematik befassen. Aufgrund begrenzter Stückzahlen spielen gerade hier die Gesamtentwicklungskosten eine große Rolle. Wie kann man aber die Kosten im Griff halten, ohne Abstriche beim Entwicklungsablauf machen zu müssen?

Charles Gardiner
Helmut Demel

PCI-Express, oder PCIe, ist der Nachfolger von PCI und bietet im Vergleich zu seinem Vorgänger eine höhere Datenübertragungsrates. Trotz seines anderen physischen Aufbaus ist PCIe softwareseitig voll kompatibel zu PCI, sodass weder Betriebssysteme und Treiber noch Anwendungsprogramme anzupassen sind. Trotzdem ist die Migration von PCI zu PCI-Express nicht kostenfrei. Natürlich könnte man eine Auflistung von Bestandteilen der Gesamtkosten über mehrere Seiten vornehmen. Aber hier sollen im Wesentlichen folgende drei Punkte betrachtet werden, die für viele Kunden auch die wesentlichen Faktoren darstellen: Einarbeitung, Bauteinkosten und Verifizierung. Bei der Einarbeitung helfen Schulung und Beratung. Hinsichtlich Bauteinkosten stellen FPGAs oftmals eine

attraktive Lösung dar. So bietet Lattice die Low-Cost-FPGAs der Familie »ECP2/M«, für die es einen PCIe-IP-Core gibt. Insbesondere die Verifizierung stellt viele Firmen jedoch vor ein größeres Problem, da hier höhere

Investitionskosten für Messgeräte oder Simulationsmodelle ins Haus stehen können. Die Erfahrung zeigt, dass die meisten Probleme bei der Implementierung bei der Kommunikation der Anwenderlogik mit

dem Core entstehen. Somit lassen sich die Verifikationskosten deutlich senken, wenn man einen Weg findet, diesen Teil am Rechner zu simulieren und zu testen. Obwohl der PCIe-Core von Lattice eigentlich nur als »obfuscated Verilog«-Netzliste verfügbar ist, ist es trotzdem möglich, eine vollwertige, auf Transaktionen basierte VHDL-Verifikationsumgebung (Transaction-Level-Modell) um den Core herum aufzubauen.

PCIe im Überblick

PCI-Express hat viele Eigenschaften eines klassischen paketvermittelnden Netzwerks. Es werden immer nur zwei Teilnehmer direkt miteinander verbunden. Ein hierarchisches System lässt sich mittels aktiven Mehrfachverzweigungen (Switches) beziehungsweise Brücken aufbauen. Das Übertragungsprotokoll ist an das bekannte ISO/OSI-Schicht-

modell angelehnt, wobei nur die physikalische, die Datensicherungs- sowie die Transportschicht spezifiziert sind. Gemeinsam mit fast allen aktuellen Systemschnittstellen im PC-Umfeld bedient sich PCIe einer seriellen Übertragung auf Basis der Differenzialtechnik mit 8B/10B-Zeichenkodierung, wobei eine Übertragungsbirrate von 2,5 GBit/s (PCIe 1.x) oder 5,0 GBit/s (PCIe 2.x) gewählt wurde. PCI-Express erlaubt die Bündelung von Datenbahnen, um die Übertragungsleistung noch mal zu erhöhen. Es sind die Konfigurationen x1, x2, x4, x8, x12, x16 und x32 spezifiziert, womit theoretisch eine Gesamtübertragungsleistung von 250 MByte/s (x1, PCIe 1.x) bis 16 GByte/s

(x32, PCIe 2.x) pro Richtung erreichbar wäre. Bei PCI-Express bildet ein ganzes Datenpaket eine »atomare« Übertragungseinheit. Nutzpakete bestehen aus einem drei oder vier Doppelworte großen Header und, sofern Daten mit übertragen werden, aus bis zu 1024 Doppelworten großen Nutzdaten (Payload). Ein Doppelwort besteht aus 32 Bit. Die Datensicherungsschicht hat dafür zu sorgen, dass ein Paket vom Sender zum Empfänger sicher übertragen wird. Hierzu werden zweierlei Mechanismen angewendet: das »ACK/NACK«-Protokoll und die »Flow Control«. Jedes Nutzpaket wird mit einer 12-Bit-Sequenznummer sowie einer 32 Bit großen Prüfsumme (Link Cyclic Redundancy Check, LCRC) erweitert. Die Sequenznummer dient dazu, das Paket zwischen den Link-Partnern eindeutig zu identifizieren. Die LCRC-Prüfsumme sichert die Datenübertragung zwischen den zwei Link-Partnern gegen Datenverfälschung ab. Eine Kopie des Pakets wird im so genannten Replay-Buffer zwischengespeichert. Den Replay-Buffer überwacht außerdem ein Timer. Falls der Link-Partner weder ein ACK- (acknowledge) noch ein NACK-Quittierungspaket (not acknowledge) mit derselben Sequenznummer innerhalb des Timeout-Fensters zurücksendet, wird das Paket aus dem Replay-Buffer erneut gesendet. Das Originalpaket wird ebenso wiederholt, wenn der Link-Partner mit einer NACK-Quittierung reagiert (Bild 1). Die ACK/NACK-Pakete gehören

zu den Verwaltungspaketen und haben eine Größe von 6 Byte.

Zehn Übertragungsarten

In regelmäßigen Abständen tauschen die zwei Teilnehmer eines PCIe-Links 6 Byte große Flow-Control-Pakete (FC) aus. Diese Pakete informieren über die Aufnahmefähigkeit des FC-Senders in Bezug auf weitere Datenpakete. Diese Information

Message-Anforderung mit bzw. ohne Daten und Completion-Quittierung mit bzw. ohne Daten. Die anfordernden Übertragungsarten werden wiederum in die zwei Gruppen »posted« (Speicher schreiben bzw. Messages) und »non-posted« (Speicher lesen, I/O-Zugriffe und Konfigurations-Zugriffe) geteilt. Die »Non-posted«-Zugriffe müssen vom Empfänger mit einem Quittierungspaket (Completion Packet) beantwortet werden, das je nach Anforderungstyp aus einem Header beziehungsweise von der Transportschicht des FC-Senders bezogen. Die Applikationslogik darf nur dann eine Datenübertragung initiieren, wenn der Link-Partner bereits eine ausreichende Aufnahmefähigkeit angezeigt hat, so genannte »Credits«. Die Applikationslogik einer PCIe-Express-Lösung wird in der Transportschicht untergebracht. Die Spezifikation definiert insgesamt zehn Übertragungsarten:

- Speicherlese- bzw. -schreibanforderung,
- I/O-Lese- bzw. -Schreibanforderung,
- Konfigurationslese- bzw. -schreibanforderung,
- Message-Anforderung mit bzw. ohne Daten und
- Completion-Quittierung mit bzw. ohne Daten.

Die anfordernden Übertragungsarten werden wiederum in die zwei Gruppen »posted« (Speicher schreiben bzw. Messages) und »non-posted« (Speicher lesen, I/O-Zugriffe und Konfigurations-Zugriffe) geteilt. Die »Non-posted«-Zugriffe müssen vom Empfänger mit einem Quittierungspaket (Completion Packet) beantwortet werden.

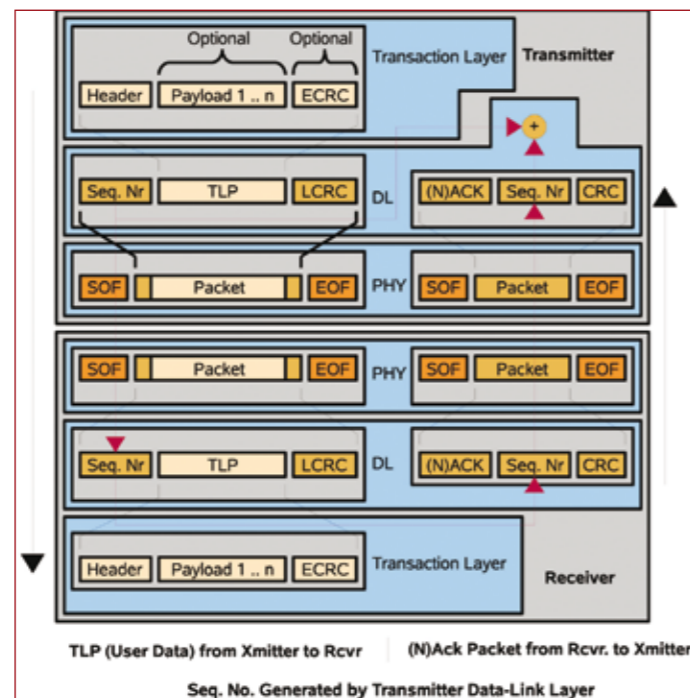


Bild 1: Übertragung eines Datenpakets mit Quittierung in der Datensicherungsschicht

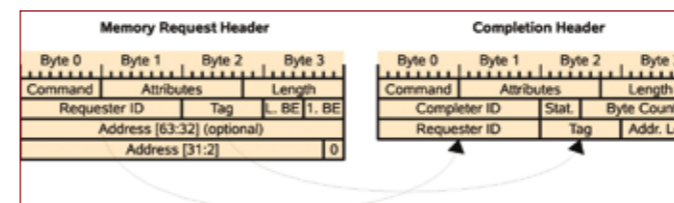


Bild 2: Header-Format mit Anforderung und Quittierung

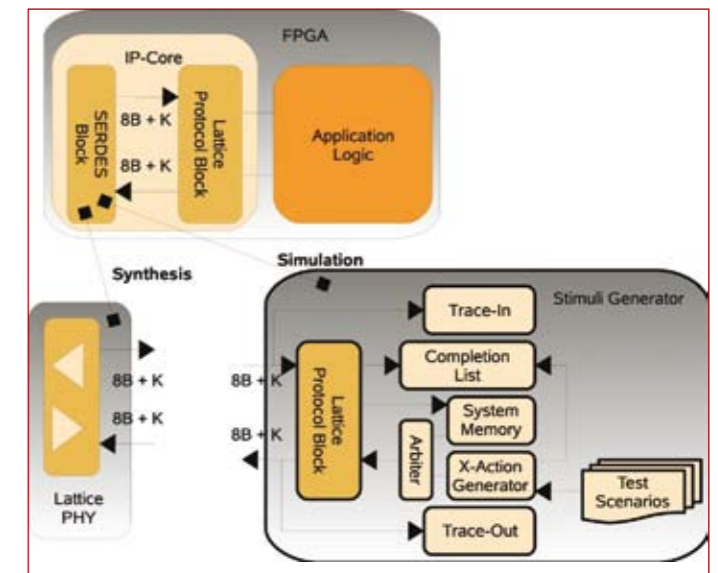


Bild 3: Austausch des SERDES-Moduls für Simulation

```

1 begin
2   -- Set up BAR
3   cfgwr0(c_csreg_bar2, X"FFFFFFF",
4     clk, tlp, sv, rv, cple);
5   -- Wait for completion response w/o payload (Cpl)
6   wait_cpl_rsp(clk, rv);
7   idle(clk, 5);
8   -- Expect 64K Window in 64-bit mem space
9   cfgrd0(c_csreg_bar2, X"FFFF000C",
10    clk, tlp, sv, rv, cple);
11  -- Wait for compl. response w/ payload (CplD)
12  wait_cpld_rsp(clk, rv);
13  cfgwr0(c_csreg_bar2, X"12340000",
14    clk, tlp, sv, rv, cple);
15  wait_cpl_rsp(clk, rv);
16  idle(clk, 5);
17  cfgrd0(c_csreg_bar2, X"1234000C",
18    clk, tlp, sv, rv, cple);
19  wait_cpld_rsp(clk, rv);
20  -- Enable Mem. Space
21  cfgwr0(c_csreg_command, X"00000002",
22    clk, tlp, sv, rv, cple);
23  -- Again, wait for completion w/o payload (Cpl)
24  wait_cpl_rsp(clk, rv);
25  idle(clk, 5);
26
27  -- Initialise Memory with background pattern.
28  -- c_membg_4kb <0..15> are predefined random data
29  -- blocks of 1K DW size, provided by the generator
30  memwr(X"12340000",
31    c_membg_4kb_i, clk, tlp, sv, rv);
32
33  -- Generate all Xfer sizes from 1 to v_loop_max
34  -- using random data. Rd / Wr in parallel
35  for i in 0 to v_loop_max loop
36    v_data_ptr := v_loop_max - i;
37    memwr(X"12340000",
38      c_membg_4kb_0(v_data_ptr to v_loop_max),
39      clk, tlp, sv, rv);
40    memrd(X"12340000",
41      c_membg_4kb_0(v_data_ptr to v_loop_max),
42      clk, tlp, sv, rv, cple);
43  end loop;
44  -- Wait until all completions have been received
45  while (cplq.is_empty = false) loop
46    idle(clk, 25);
47  end loop;
48  idle(clk, 20);
49  end ;

```

Listing 1: Testscenario mit paralleler Schreib/Leseaktivität

derung wird über die Tags sowie Requester-ID-Felder im Paket-Header koordiniert (Bild 2). Das Statusfeld im Completion-Header zeigt an, ob die Anforderung erfolgreich durchgeführt wurde oder ob Fehler aufgetreten sind. Als Fehlerzustände gel-

ten nur nicht unterstützte Anforderungen (Unsupported Request) oder Abbruch (Completer-Abort, ähnlich Target-Abort aus klassischer PCI). PCI-Express kennt keinen »Retry«-Zustand. Wenn hohe Anforderungen an die Datenintegri-

tät vorliegen, lässt sich das Datenpaket durch eine 32-Bit-ECRC aus der Transaktionsschicht absichern. Diese Prüfsumme begleitet das Paket auf dem ganzen Weg vom Datenerzeuger bis zum Endabnehmer (z.B. von einer Steckkarte durch eine/mehrere Switches oder Bridges bis zum Hauptspeicher). Die LCRC-Prüfsumme der Datensicherungsschicht dagegen sichert nur die Übertragung zwischen zwei unmittelbar benachbarten PCIe-Teilnehmern ab. Auf der PCI-Express-Schnittstelle zugewandten Seite besteht die Applikationslogik somit im Wesentlichen aus Paketgeneratoren (z.B. DMA-Controller, der SpeicherSchreib-/Lese-Anforderungen initiiert oder Completion-Paket-Generator, der angeforderte Lesedaten zurückliefert) beziehungsweise Paket-Dekodierern, um ankommende Anforderungen zu interpretieren und gegebenenfalls die enthaltenen Nutzdaten in den lokalen Speicher zu übertragen.

PCI-Express-Core

Die IP-Core-Lösung von Lattice enthält eine vollständige Implementierung der physikalischen Schicht (PHY) sowie der Datensicherungsschicht und zusätzlich

Teile der Transportschicht wie zum Beispiel die »PCI Configuration Space Register« und die ECRC-Generator beziehungsweise Prüflöge. Konfigurationszugriffe von der PCIe-Schnittstelle werden eigenständig abgewickelt. Als Schnittstelle zur Applikationslogik wird ein einfacher 16-Bit- (x1-Core) oder 64-Bit-DMA-Anschluss (x4-Core) angeboten. Strukturell besteht der IP-Core aus einem SERDES-Modul (PHY-Implementierung) und einem Protokollmodul. Beide kommunizieren miteinander über zwei Zeichenschnittstellen (Eingang/Ausgang), die aus einem 8-Bit-Datenpfad und einer Steuerzeichenanzeige (K-Leitung) bestehen. Die K-Leitung wird aktiviert, um Sonderzeichen wie zum Beispiel »Rahmenanfang« (Start of Frame, SOF) beziehungsweise »Rahmenende« (End of Frame, EOF) anzuzeigen. Obwohl es sich bei der FPGA-Familie »LatticeECP2/M« um Low-Cost-Bausteine handelt, bietet sie neben den schon erwähnten SERDES-Blöcken noch einige andere Features, um die Anwenderlogik zu implementieren. Die Familie stellt für die Logikimplementierung beim kleinsten Baustein bereits 19 000 LUT-4 (Look-up Tables) und beim größten Baustein 95 000

LUT-4 zur Verfügung. Im Vergleich zur »LatticeECP2«-Familie wurde die Größe der Speicherblöcke um den Faktor fünf erhöht, sodass nun bis zu 1032 KBit in diskreten Speicherblöcken zur Verfügung stehen. Darüber hinaus lassen sich kleinere Speichereinheiten als Distributed-Memory realisieren. Gerade bei PCIe-Anwendungen besteht oft die Notwendigkeit, Daten zwischenspeichern. Hierzu kann in der LatticeECP2/M-Familie die DDR-Schnittstelle mit bis zu 400 MBit/s und die DDR2-Schnittstelle mit bis zu 533 MBit/s betrieben werden.

Bei vielen Anwendungen (Bildverarbeitung, Sensorapplikationen und Telekom) sind auch die fest verdrahteten DSP-Blöcke mit Multiplizierer, Addierern und Akkumulatoren eine große Hilfe, da sie die mögliche Systemfrequenz erhöhen und gleichzeitig die Auslastung des Bausteines verringern. Speziell für PCI-Express x1 und x4 stehen auch Evaluierungsboards zur Verfügung, mit denen der Anwender den Core gemeinsam mit der Anwenderlogik auch im System testen kann. Dies ist sogar auch ohne IP-Core-Lizenz möglich. Das Evaluierungskonzept von Lattice sieht vor, dass noch unlizenzierte Cores mehrere Stunden funktionstüchtig sind, bevor die FPGAs automatisch angehalten werden. Durch Auslösung des Resets kann der Anwender den internen Zähler aber sofort wieder zurück- und die Evaluierung fortsetzen.

Vor der Evaluierung oder Fehlersuche in der Hardware steht natürlich die Simulation. Ein Ansatz wäre der Einsatz eines »Bus Functional Models«, das käuflich erworben werden müsste. Um solch eine Investition zu vermeiden, gibt es ein anderes Testkonzept, bei dem der SERDES-Block durch

einen Stimuli-Generator ersetzt wird. Das im Folgenden beschriebene Konzept verwendet diesen Ansatz. Die Verbindung zwischen SERDES- und Protokoll-Block (8-Bit- + K-Leitung) wird als Abgriffspunkt für die Simulation herangezogen. Die Aktivität an dieser parallelen Schnittstelle lässt sich erheblich leichter stimulieren und überwachen (Monitore) als an den RX/TX-Differenzialleitungen. Ziel der Simulation ist es in erster Linie, die Applikationslogik zu testen.

Aufbau der Simulationsumgebung

Den Aufbau der Simulationsumgebung zeigt Bild 3. Der PCIe-Stimuli-Generator hat genau die gleiche Portbelegung und VHDL-Entity-Namen wie das SERDES-Modul im IP-Core. Der Stimuli-Generator wird aber in eine andere VHDL-Bibliothek kompiliert, als die Bibliothek, die die Projektlogik enthält. Hier beispielsweise kommt eine Bibliothek namens »pcie_stimmen_lib« zum Einsatz. Durch Einfügen der folgenden Zeilen in die Top-Level-Datei des IP-Cores wird für die Simulation automatisch der Stimuli-Generator als SERDES und für die Synthese im »ispLever«-Tool nach wie vor das SERDES-Modul von Lattice gewählt:

```

-- synopsys translate_off
Library pcie_stimmen_lib;
Use pcie_stimmen_lib.pcs_pipe_top;
-- synopsys translate_on

```

Zielsetzung des Simulationskonzepts ist es, wesentlich mehr Testfälle zu ermöglichen, als nur durch einfache Loop-Back-Tests machbar wäre. Loop-Back-Tests erlauben nur selten die Durchführung von komplexen Lastszenarien (z.B. mehrere DMA-Kanäle aktiv und

FOR CALL PAPERS

Kostengünstige Entwicklung und Fertigung komplexer

LEITERPLATTEN

am 07. Juli 2009 im Novotel Messe München

Zum siebten Mal findet am 07. Juli 2009 das DESIGN&ELEKTRONIK-Entwicklerforum »Kostengünstige Entwicklung und Fertigung komplexer Leiterplatten« statt. Der große Erfolg von Konferenz und Ausstellung zeigt, dass auf diesem Gebiet auch weiterhin großer Informationsbedarf besteht, doch rücken in weniger guten Zeiten neue Aspekte in den Vordergrund.

Wir begrüßen Beiträge aus dem gesamten Spektrum der Leiterplattenentwicklung- und Fertigung unter dem Gesichtspunkt der Kostenoptimierung, beispielsweise:

- Störungen erkennen: Wechselwirkungen zwischen Schaltungsfunktion, Boardlayout und EMV-Verhalten
- Richtige Planung: Teure Fehler im Schaltungsdesign und im Boardlayout vermeiden?
- Ausschuss verringern: Reparatur und Rework von Platinen und Baugruppen
- Wider die Verschwendung: Low-Power-Design und gutes Layout
- Iterationen vermeiden: Design for Manufacturing/Design for Test
- Gehversuche: Erstellen von Prototypen
- »Right first time« – gibt es optimale Fertigungsunterlagen?
- Passgenau: Wahl der passenden Fertigungstechnologien, Maße und Toleranzen
- Gute Platzierung: Probleme bei der Bestückung im Vorfeld vermeiden
- Dauertest: Kostengünstige fertigungsbegleitende Prüf- und Testmethoden
- Normen: Reibungslose und kostengünstige Zertifizierung der Leiterplatte
- Kooperation: Wie gestaltet man optimal die Schnittstellen im Fertigungsprozess?
- Outsourcing: Dienstleister unterstützen und begleiten den Entwicklungsprozess
- Das passende Werkzeug: EDA- und CAD-Systeme im Vergleich
- »Bei uns war das so...«: Erfahrungsberichte aus Entwicklung und Herstellung

Natürlich sind auch andere Themenvorschläge willkommen. Bitte senden Sie eine aussagekräftige Kurzfassung Ihres Vortrags in Deutsch oder Englisch (max. 2 A4-Seiten) bis zum **20. März 2009** an: Marcel Consée, Leitender Redakteur DESIGN&ELEKTRONIK, mconsee@design-elektronik.de Die Kurzfassung sollte enthalten:

- Titel des Beitrags
 - Name des Autors/der Autoren
 - Post- und Email-Adresse des Autors
 - Telefon- und Fax-Nummer
 - Inhaltsangabe des Vortrags in Kurzform
- Der endgültige, später einzusendende Beitrag für den Tagungsband, soll 10 Seiten DIN A4 nicht wesentlich überschreiten. Eine rein technische Abhandlung des Themas ist zwingend erforderlich. Marketingorientierte Vorträge werden nicht akzeptiert. Wir freuen uns auf Ihre interessanten Beiträge!

```

1 KERNEL: PCIE_DUT: update_posted at time 787108 ns
2 KERNEL: PCIE_X1: mem_wr at time 787380 ns, addr = 0x12340180, length = 7, seq-nr. = 480, tag = 5
3 KERNEL: PCIE_X1: mem_rd at time 787580 ns, addr = 0x12340000, length = 32, seq-nr. = 481, tag = 5
4 KERNEL: PCIE_X1: mem_rd at time 787692 ns, addr = 0x12340080, length = 32, seq-nr. = 482, tag = 6
5 KERNEL: PCIE_DUT: ack at time 787756 ns, seq-nr. = 480
6 KERNEL: PCIE_X1: mem_rd at time 787780 ns, addr = 0x12340100, length = 32, seq-nr. = 483, tag = 7
7 KERNEL: PCIE_DUT: ack at time 787844 ns, seq-nr. = 481
8 KERNEL: PCIE_DUT: ack at time 787956 ns, seq-nr. = 482
9 KERNEL: PCIE_DUT: ack at time 788044 ns, seq-nr. = 483
10 KERNEL: PCIE_DUT: update_non_posted at time 788108 ns
11 KERNEL: PCIE_X1: mem_rd at time 788380 ns, addr = 0x12340180, length = 7, seq-nr. = 484, tag = 8
12 KERNEL: PCIE_DUT: cpl_d at time 788172 ns, length = 16, seq-nr. = 391, tag = 5
13 KERNEL: PCIE_DUT: cpl_d at time 788524 ns, length = 16, seq-nr. = 392, tag = 5
14 KERNEL: PCIE_X1: mem_wr at time 788468 ns, addr = 0x12340000, length = 32, seq-nr. = 485, tag = 9
15 KERNEL: PCIE_DUT: cpl_d at time 788876 ns, length = 16, seq-nr. = 393, tag = 6
16 KERNEL: PCIE_DUT: cpl_d at time 789244 ns, length = 16, seq-nr. = 394, tag = 6
17 KERNEL: PCIE_DUT: ack at time 789580 ns, seq-nr. = 485
18 KERNEL: PCIE_X1: mem_wr at time 789068 ns, addr = 0x12340080, length = 32, seq-nr. = 486, tag = 9
19 KERNEL: PCIE_X1: ack at time 789692 ns, seq-nr. = 393

```

Listing 2: Log-Meldung im Konsolenfenster des Simulators

Requester-Pakete			Entsprechende Completer-Pakete		
Typ	Beschreibung	Payload	Typ	Beschreibung	Payload
Msg	Message	0			
MsgD	Message mit Payload	1 bis 1024			
MWwr	Memory Write	1 bis 1024			
MRd	Memory Read	0	CplD	Completion mit Daten	1..1024
IOWr	I/O Write	1	Cpl	Completion	0
IORd	I/O Read	0	CplD	Completion mit Daten	1
CfgWr	Config. Write	1	Cpl	Completion	0
CfgRd	Config. Read	0	CplD	Completion mit Daten	1

Tabelle 1: PCIe-Anforderungs-/Quittierungspakete

gleichzeitige externe Zugriffe über den PCIe-Link). Kern des PCI-Express-Stimulators ist natürlich der Transaktionsgenerator. Der Stimuli-Generator stellt eine Reihe von VHDL-Prozeduren bereit, die der Entwickler verwenden kann, um die einzelnen Testszenarien zu beschreiben. Es sind so gut wie alle PCIe-Zugriffe aus der Sicht eines Upstream-Ports implementiert. Ein solches Testszenario wird exemplarisch in Listing 1 gezeigt. Hier ist zu sehen, wie ein Adressbereich über ein Basis-Adressregister eingerichtet wird und anschließend zufällige Daten in den Bereich geschrieben beziehungsweise gelesen werden, um Buslast zu erzeugen. Die in Zeile 35 beginnende for-Schleife führt dazu, dass Schreibpakete vom Upstream-Port und Lese-Anwortpaketen (Completions) vom End-Point simultan aktiv sind. Es werden also die Puffer in beiden Richtungen gleichzeitig beansprucht.

Im Falle von »non-posted«-Anforderungen (Memory Read, I/O-Read/Write, Configuration-Read/Write) trägt der Transaktionsgenerator eine Kopie des erwarteten Antwortpakets in die Completion-Liste ein. Hierdurch überprüft der Stimuli-Generator folgende Punkte automatisch:

■ Treffen alle erwarteten Completion-Pakete ein?

- Wird die Reihenfolge eingehalten?
- Werden bei Completions mit Datenübermittlung (CplD) die richtigen Daten geliefert?
- Enthalten die Header-Felder (z.B. Tag, Requester-ID, Completer-ID, Length, Byte-Count, Address-Low) die richtigen Werte (Bild 2)?

Nach der Überprüfung werden alle eintreffenden Completion-Pakete automatisch wieder aus der Completion-Liste entfernt. Die im Listing 1 in der Zeile 45 beginnende while-Schleife wartet am Ende der Simulation bis alle erwartete Completion-Pakete eingetroffen sind.

Der Stimuli-Generator ist in der Lage den gesamten externen Systemspeicher abzubilden. DMA-Controller in der Applikationslogik können damit beliebige Zugriffe ausführen, sowohl in einem 32-Bit-adressierbaren Speicherbereich als auch in einem 64-Bit-adressierbaren Speicherbereich. Das Speichermodell legt aber nur die tatsächlich von der Applikationslogik adressierten Bereichen an – so genanntes »Sparse Memory« –, um Simulatorressourcen zu schonen.

Wenn ein DMA-Controller in der Applikationslogik eine Leseanforderung stellt, generiert das Speichermodell das dazu passende Completion-Paket beziehungsweise -Pakete. Für die Übertragung zur FPGA-Logik muss in diesem Fall,

zusammen mit dem Transaktionsgenerator, der Zugang zur Sendeschnittstelle am PCI-Express-Core koordiniert werden, damit Transaktionen, die vom Testszenario ausgelöst werden, die Completion-Paket nicht stören.

Anforderungen an den Simulator

Damit der Entwickler den Simulationsablauf auch betrachten kann, werden die Paketströme am Eingang und am Ausgang vom PCIe-Core im Stimuli-Generator überwacht. Hierbei werden wichtige Informationen wie zum Beispiel Sequenznummer des Pakets, Adresse und Tag-ID des Pakets mit ausgegeben. Im Falle von Datenvergleichsfehlern oder fehlerhaften Paketen lässt sich somit der gesamte dazugehörige Ablauf leichter nachvollziehen.

Da der IP-Core von Lattice nur im Verilog-Format vorliegt, muss der Simulator beide Lizenzen beinhalten. Die Simulatoren der Firma Aldec (»ActiveHDL« für Windows bzw. »Riviera« für Linux/Unix/Windows) unterstützen mit nur einer Lizenz das Simulieren von gemischten Verilog- und VHDL-Projekten. Eine für das Arbeiten mit Lattice-Produkten konfigurierte Version des ActiveHDL-Simulators ist Bestandteil der Entwicklungsumgebung »ispLever«. Eine kostenlose Version von

ActiveHDL, die mit der ispLever-Starter-Software freigeschaltet werden kann, unterstützt übrigens nur eine Sprache (wählbar) und ist somit nicht geeignet.

Listing 2 zeigt einen Auszug aus dem Simulationsablauf mit dem Simulator »Riviera«. Die nach links ausgerichteten Zeilen melden Aktivitäten aus der PCIe-Transportschicht, die eingerückten Zeilen zeigen Aktivitäten aus der Datensicherungsschicht, die transparent zur Applikationslogik ausgelöst werden. Die Zeilen 3, 12 und 13 zeigen eine Speicher-Leseanforderung und die zwei dazugehörigen Completion-Pakete. Im hier simulierten System war es eine Designentscheidung, die Latenz zu priorisieren. Größere Leseanforderungen werden mit mehreren kleineren Quittierungen beantwortet. Die Zeilen 9 und 17 zeigen einen Fall, bei dem zwei Memory-Read-Anforderungen mit einem einzigen ACK-Paket quittiert werden. Ein explizites ACK-Paket für die Sequenznummer 484 wird nicht generiert. Dadurch verringert sich der Bandbreitenanteil, der nötig ist, um Verwaltungspakete aus der Datensicherungsschicht auszutauschen. Die Zeilen 1 und 10 protokollieren den Austausch von FC-Paketen (Flow Control). Hier zeigt der PCIe-Core im FPGA an, dass Pufferbereiche in der Anwenderlogik zur Aufnahme von neuen Daten wieder freigegeben worden sind. (rh)

Charles Gardiner

ist unabhängiger Anbieter von Beratungs-, Schulungs- und Entwicklungsdienstleistungen und

Helmut Demel

ist Staff Field Application Engineer bei

Lattice Semiconductor
Telefon 08 11/55 05 60
www.latticesemi.com



Halle 10
Stand 524