Standardiserad arkitektur för testbänkar med VHDL

EXPERTARTIKEL

UVVM gör det enkelt att skapa strukturerade testbänkar med VHDL.

Av Espen Tallaksen, Bitvis

Espen Tallaksen är grundare av Bitvis, ett konsultbolag fokuserat på mjukvara för inbyggda system och FPGA-konstruktion. Han tog sin examen 1987 på universitetet i Glasgow och har arbetat på Philips halvledarverksamhet i Schweiz liksom på olika bolag i Norge med utveckling av asicar och FPGA:er. Espen har startat två företag, förutom Bitvis även Digitas.

Vi vet alla att arkitekturen i en FPGA-konstruktion – från toppen och hela vägen ner till mikroarkitekturen – är kritisk både för kvaliteten på FPGA:n och för utvecklingstiden. Det här gäller också för testbänken, men av någon anledning är testbänkar sällan strukturerade. Detta trots att det i de flesta projekt finns en stor potential för tidsbesparingar och förbättringar av kvaliteten. En viktig orsak är att det saknas kunnande inom verifiering. Lika viktigt är att FPGA-användarna inte efterfrågar en bra och gemensam metodik.

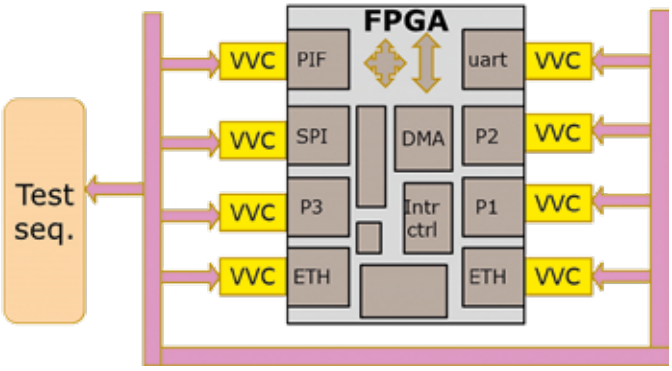
ALLT DETTA ÄNDRADES för två år sedan med lanseringen av UVVM (Universal VHDL Verification Methodology) som är ett verktyg i form av open source. UVVM ger en kraftfull och generell arkitektur för en testbänk som är enkel att använda för alla som konstruerar FPGA:er och asicar. Standardiserade verifieringskomponenter kopplas ihop precis som legobitar och en central testsekvenserare (eller flera) kan alltid applicera högnivåkommandon för att styra verifieringskomponenterna och därmed hela testbänken. Detta innebär att även personer som inte är FPGA-konstruktörer kan skriva avancerade testfall.

Därmed är det möjligt för konstruktörerna att bygga egna testnät (harness) och testfall mycket snabbare och bättre än tidigare samtidigt som de går att återanvända.

Verifieringen utgör i genomsnitt hälften av det totala arbetet i ett FPGA-projekt. Ett lika intressant faktum är att i genomsnitt hälften av verifieringstiden går åt till att avlusa. Det här betyder att det finns en enorm förbättringspotential för verifieringen generellt, men särskilt för avlusningen.

ALLA DE STORA KRETS- OCH VERKTYGLEVERANTÖRERNA pratar om detta verifieringsgap och erbjuder ”lösningar” som överbryggar det. Tyvärr är deras lösningar ofta en mycket komplex metod kallad UVM (Universal Verification Methodology) som medför att man måste lära sig ett helt nytt språk, System Verilog plus att man måste köpa väldigt dyra verktyg. De flesta FPGA-konstruktörer i Europa använder VHDL för konstruktion och verifiering, i hela världen är siffran cirka 50 procent.

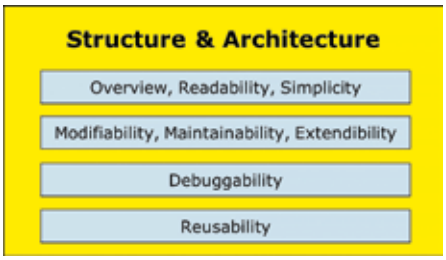
VHDL är tillräckligt kraftfullt för nästan alla FPGA-projekt. Att använda VHDL gör det möjligt för utvecklarna att fortsätta använda ett språk de redan kan för att steg för steg addera ny funktionalitet när det behövs. Det är med andra ord en mjuk och effektiv vidareutveckling av din nuvarande testbänk och ditt språk.



En typisk testbänk med dedicerade VVC:er på alla FPGA:ns gränssnitt.

Dock är finesser och funktioner långt ifrån tillräckliga. Strukturer och arkitekturer är nycklar till alla viktiga aspekter för en god FPGA-utveckling.

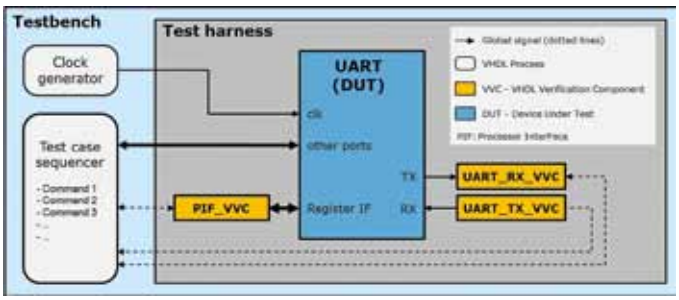
I FIGUREN VISAS visas de mest kritiska faktorerna för konstruktion och verifiering under ett projekt. Under konstruktionsfasen är vi väl medvetna om detta vilket är den viktigaste orsaken till att vi delar upp konstruktionen i delar som är mer eller mindre självständiga. Det gäller exempelvis för UART, SPI, DMA-controller och AD-omvandlare. Dessa moduler, eller VHDL-komponenter, tilldelas kommandon av mjukvaran och sedan utför de sina tilldelade



uppgifter medan mjukvaran gör något helt annat. Om något går fel eller modulen behöver hjälp signalerar den detta. Normalt görs det med en avbrottssignal, ett interrupt. Det verkligt trevliga är att FPGA-konstruktionen, som gjorts som top-level, är enkel att förstå. Vi ser modulerna och känner till deras funktioner.

Bussen som kopplar ihop modulerna är standardiserad, exempelvis AXI4-lite eller Avalon, och modulernas bussgränssnitt är standardiserade för att passa bussen. Mjukvaran som styr allt körs med högnivåkommandon.

DET ÄR UPPENBART att den struktur som är beskriven ovan är extremt effektiv och ger bra kvalitet. För testbänkar har det dock inte funnits motsvarande system eller metodik. Det är en av de viktigaste orsakerna till att testbänkar är ostrukturerade och huvudorsaken till att UVVM utvecklades.



Arkitekturen i konstruktionen återspeglas i UVVM, vilket ger en arkitektur i testbänken som verkligen är enkel att förstå och dessutom bibehåller alla fördelar från en bra och modulär struktur

- Verifieringsmoduler med en lättbegriplig funktionalitet och gränssnitt
- Ett standardiserat bussystem mellan sekvenseraren till verifieringsmodulerna
- Ett standardiserat bussgränssnitt på alla verifieringsmoduler. I UVVM kallas dessa moduler för VVC:s (VHDL Verification Components)

ARKITEKTUREN I UVVM är väldigt lik arkitekturen i en FPGA-konstruktion med den externa mjukvarusekvenseraren som nämndes ovan.

Sekvenseraren i testfallet distribuerar kommandon till VVC:erna på samma sätt som mjukvaran distribuerar kommandon till FPGA-modulerna. Sedan gör VVC:erna det de är tillsagda att göra på samma sätt som i en FPGA. Det innebär att man normalt hanterar accessen på deras fysiska gränssnitt. För FPGA-modulen kan det vara att skicka några data och det kan vara så även för VVC:n.

För testbänken till UART:en i figuren kan testsekvenseraren typiskt skicka ett kommando till UART TX VVC för att en byte data ska skickas till RX-ingången på UART:en.

I UVVM finns det två sätt att nå gränssnitten hos det som ska testas (DUT, Device Under Test). Det enklaste är en vanlig BFM-procedur (Bus Functional Model) som:

```
uart_transmit(x"4C", "Apply byte on UART RX");
```

DEN SISTA DELEN fungerar som kommentar till koden och skrivs ut eller loggas när den behövs. Den här proceduren kommer att exekveras direkt och applicerar x4C till UART RX med hjälp av UART:ens protokoll.

Det betyder att om testsekvenseraren anropar proceduren kommer den att applicera UART:ens protokoll bit för bit. Det kommer att ta en stund. Under den tiden kommer testsekvenseraren att vara upptagen och kan inte göra något annat.

En mer avancerad form kallas CDM (Command Distribution Method):

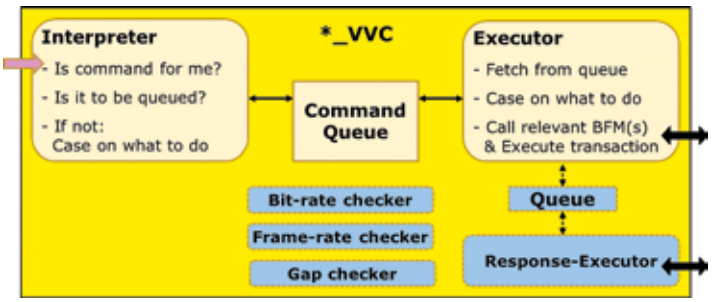
```
uart_transmit(UART_TX_VVCT,1   x"4C", "Apply byte on UART RX");
```

OM SEKVENSERAREN ANROPAR den här proceduren – allt som händer direkt är att detta kommando skickas till VVC med den måladress som anges i rött – med ett namn och instansieringsnummer av VVC:n så exekveras kommandot på testobjektet, DUT:en. Att överföra testkommandot från testsekvenseraren till VVC:n tar ingen tid vilket betyder att testsekvenseraren omedelbart är tillgänglig för att göra något annat, som att instansiera en access till ett annat gränssnitt. Det tidsödande protokollet för UART:en tas om hand av VVC:n.

Kodexemplet nedan visar hur ett litet test kan skrivas för att samtidigt testa accessen till alla gränssnitt. Exemplet är från en annan testbänk där VVC:erna för UART:ens TX och RX har slagits ihop till en och samma VVC för att enklare kunna återanvändas. För det här exemplet har vi adderat en annan parameter för att visa kanalen (RX vs TX).

```
sbi_write(SBI_VVCT,1, C_ADDR_TX_DATA, x"A0", "Send byte UART TX");
uart_expect(UART_VVCT,1,RX  x"A0", "Check byte from UART TX");
uart_transmit(UART_VVCT,1,TX  x"A1", "Apply byte on UART RX");
wait for C_FRAME_PERIOD;
sbi_check(SBI_VVCT,1, C_ADDR_RX_DATA, x"A1", "Check UART RX byte");
```

I en FPGA-konstruktion är det bra, men inte tillräckligt, att ha en bra arkitektur för den översta nivån. Det är lika viktigt att ha en bra mikroarkitektur hela vägen ner. Detta gäller såklart också för testbänken vilket i det här fallet innebär att också arkitekturen i VVC:erna måste vara enkla att förstå, modifiera och återanvända. Återigen skulle standardisering vara bra.



DEN GRUNDLÄGGANDE VVC:N består av de gula blocken i figuren. Dessa kommer att se CDM:erna ovanför och bara acceptera kommandon som är avsedda för exakt denna VVC. Den kommer då att lägga kommandot uart_transmit() på kön och låta testsekvenseraren fortsätta. Exekveraren kommer att hämta kommandot från kön, upptäcka att det är ett kommando för UART:en och sedan exekvera BFM:en på testobjektet med de bifogade parametrarna.

Den starkt strukturerade arkitekturen i VVC:erna har ytterligare en fördel. Det är enkelt att addera mer funktionalitet i strukturen. Det kan vara mer avancerade gränssnitt eller att bara ta med återanvänd funktionalitet.

Ett protokoll som Avalon MM resulterar ofta i kaotiska testbänkar eftersom det kan komma flera läskommandon innan det första svaret dyker upp. Med VVC-arkitekturen är det löjligt enkelt och mycket strukturerat. Exekveraren hämtar läskommandot från kön och vet att det är en delad access. Den anropar då BFM-proceduren och samtidigt skickar den läskommandot till nästa kö. Svarsexekveraren hämtar det och sen anropar den BFM-proceduren och väntar på respons. På det här sättet kan ett godtyckligt antal läsförfrågningar göras oberoende av svaren. Allt är snyggt strukturerat.

I UVVM finns ett script som genererar VVC:er för nya gränssnitt. Koden som skapas är en mall och det markeras klart och tydligt var användaren måste addera sina egna BFM:er och andra modifieringar som behöver göras. BFM:er måste tas fram för alla testbänkar. När man gjort det några gånger tar det inte mer än ungefär 30 minuter att skapa en VVC från grunden.

”Den starkt strukturerade arkitekturen i VVC:erna har ytterligare en fördel. Det är enkelt att addera mer funktionalitet i strukturen”

ATT UNDER LÅNG TID standardisera FPGA-konstruktionerna genom ett bra koncept för arkitekturen, med register för styrning och status, ett bussystem, autonoma moduler och kommandon för högnivåprogramvara, har resulterat i en effektiv designmetodik för FPGA:er men också effektiv mjukvaruutveckling i och med att mjukvaruutvecklarna inte behöver förstå detaljerna i modulerna som de styr. Det här kan synas uppenbart, och det är det. Så varför har det inte hänt med testbänkarna, där vi har exakt samma scenario med en sekvenserare som styr olika gränssnitt?

UVVM ERBJUDER JUST DETTA samtidigt som det också standardiserar de viktigaste delarna:

- Standardmodul (VVC) för gränssnitt till styrning och status
- Standardprotokoll från sekvenserare till VVCs
- Standardkommandon för sekvenseraren (för gemensamma uppgifter)
- Standardiserad intern arkitektur för VVC:n
- Standardkommando för kösystemet
- Standardhantering av flertrådade gränssnitt (exempelvis delade transaktioner)
- Standardmetoder för synkronisering mellan VVC:er (eller andra processer)
- Standardiserade vänt- och timeout-signaler
- Standardmeddelanden och larmhantering
- Standardkommandon för multicast och broadcast
- Standardsupport för avlusning (debugging)

INTERNATIONELLA SPELARE med ett starkt intresse för VHDL och verifiering har anammat UVVM. Aldec kör redan ett antal webinar om UVVM och kommer snart att inkludera UVVM i sina simulatorer.

Doulos är störst på utbildning på FPGA:er och asicar. De rekommenderar nu UVVM för testbänkar. Ett sista bevis att UVVM verkligen har momentum är att ESA, den Europeiska Rymdstyrelsen sponsrar utvidgningar av UVVM med över 200 000 euro och kommer att rekommendera sina leverantörer att använda UVVM för FPGA:er.

UVVM kommer med gratis VVC:er och UBM:er för exempelvis UART, I2C, SPI, SBI, AXI4-lite, AXI4-stream, GPIO och Avalon MM vilket gör det enkelt att komma igång. Under andra kvartalet i år kommer vi att släppa en mekanism för att dela VVC:er mellan VHDL-användare. Det gäller både som open source men också för kommersiella VVC:er. ■