Så får mjukvaruexp

Gör modellen i mjukvara, bygga och testa i hårdvara

Av Stéphane Monboisset, PLDA Group

Stéphane Monboisset är marknadsansvarig på PLDA Group. QuickPlay-plattformen, beskriven här, ingår i hans ansvarsområde. Innan PLDA arbetade han på Xilinx i tio år. Dit kom han när företaget köpte uppstartsföretaget Triscenda år 2004. Stéphane har även arbetat på ST Microelectronics i Kalifornien, Frankrike och Singapore. Han har en MSEE och MBA från Paris.

Vanliga processorer räcker inte till för framtida datatjänster. Deras begränsade förmåga till parallellisering innebär att processorkraften inte blir tillräcklig utan att effektförbrukningen blir oacceptabelt hög.

Istället har tillverkare av utrustning till datacentraler länge velat dra nytta av FPGA:ns förmåga till massiv parallellism – allt för att nå den bandbredd och bearbetningsprestanda som efterfrågas inom en angiven effektbudget. Traditionellt har detta varit svår rent konstruktionsmässigt. Visserligen har konstruktionsmetoder som inkluderar verktyg för högnivåsyntes och programmeringsspråk som OpenCL, C och C++ förenklat uppgiften, men användaren har ändå varit tvungen att ha kunskap inom FPGA-konstruktion.

Det finns således ett behov av ett arbetsflöde som tillåter mjukvaruingenjörer att använda en FPGA som en mjukvarudefinierad bearbetningsplattform utan att det krävs hårdvaruexpertis. Ett sådant arbetsflöde ska kunna:

- Skapa fungerande hårdvara genom kod
- Införliva befintliga IP-block om det behövs
- Skapa allt hårdvarustöd (gränssnitt, styrning, klockor etc.)
- Stödja användning av kommersiella, ”off-the-shelf”-kort och skräddarsydda plattformar
- Eliminera hårdvarudebug
- Stödja debug av funktionella block enbart med hjälp av standardiserade mjukvaruverktyg

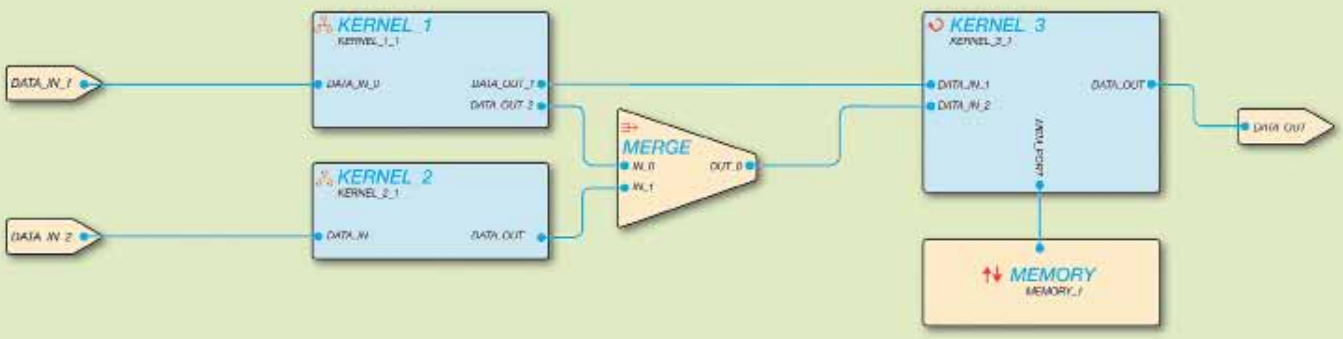
BETÄNK EN ALGORITM som består av två basfunktioner: data bearbetas i en funktion för att sedan skickas till en annan för ytterligare bearbetning. Ur ett mjukvaruperspektiv är denna realisering lika enkel som ett anrop av Funktion1 följt av ett separat anrop av Funktion2, där pekare används för att lokalisera platsen för data som ska bearbetas. Vill man implementera en sådan algoritm i en FPGA-plattform utan tillgång till rätt verktygsflöde för hårdvaruabstraktion krävs det att en mjukvaruutvecklare skapar en hårdvarukonstruktion som liknar den i figur 1 (där Kernel1 och Kernel2 är hårdvaruimplementeringar av Funktion1 och Funktion2).

”Till och med en erfaren hårdvarukonstruktör kan behöva flera veckor på sig för att skapa den allra enklaste konstruktionen för en ny FPGA”

Hårdvarukonstruktionen måste inkludera både styr- och dataplanet. Styrplanet är den del som genererar klockor och resetsignaler, hanterar uppstart av systemet, dirigerar dataplanets verksamhet samt utför alla housekeeping-funktioner. Dataplanet initierar och ansluter bearbetningselementen, Kernel1 och Kernel2, liksom de IO-gränssnitt som krävs för att läsa data och skriva bearbetat data. I exemplet i figur 1 är dessa gränssnitt Ethernet och PCI Express (PCIe).

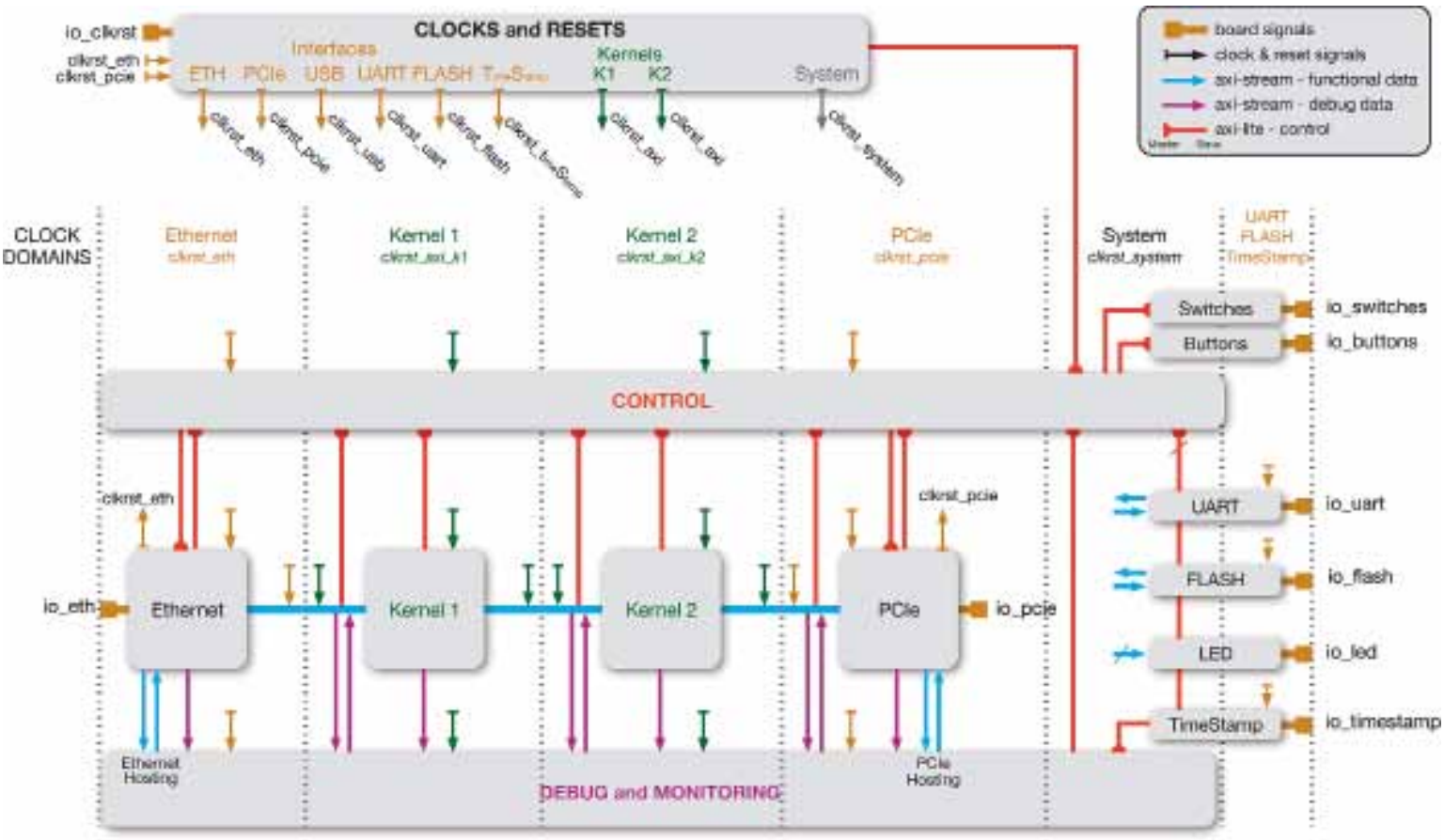
EN MJUKVARUUTVECKLARE utan specifik sakkunskap om hårdvara kan skapa Kernel1 och Kernel2 med hjälp av ett utvecklingsverktyg såsom Xilinx Vivado HLS. Verktyget kompilerar programvarufunktionerna Funktion1 och Funktion2 som de står skriva i C eller C ++ till hårdvarubeskrivningar i VHDL eller Verilog.

Andra delar av konstruktionen, sådana som inte beskrivs med algoritmer exempelvis gränssnitt, styrning, klockor och reset,



Figur 2. Ett konstruktionsexempel i QuickPlay.

erten entré till FPGA:n



Figur 1. En detaljerad hårdvaruimplementering av en två-funktionsalgoritm med traditionella FPGA-verktyg.

kan inte skapas med HLS-verktyg. Istället måste hårdvarukonstruktörer skapa dessa delar som skräddarsydda hårdvarubeskrivande funktioner eller ip. Arbetet med att sourca dessa delar och ansluta dem är ytterligare en utmaning eftersom vissa delar kanske inte är lättillgängliga eller så kan de ha olika gränssnitt liksom olika krav på klockning, uppstartsspecifikationer och så vidare.

Arbetet att implementera själva konstruktionen är också tufft. Här ingår att mappa konstruktionen att passa den valda FPGA-plattformen, skapa lämpliga begränsningar och bekräftar att dessa begränsningar är uppfyllda efter logisk syntes och implementation i hårdvaran. Till och med en erfaren hårdvarukonstruktör kan behöva flera veckor på sig för att skapa den allra enklaste konstruktionen för en ny FPGA.

FÖRETAGET PLDA GROUP – som utvecklar ip för FPGA:er – har skapat QuickPlay för att ge mjukvaruutvecklare möjlighet att implementera applikationer avsedda för processorer helt eller delvis i FPGA:er. Metodiken är mjukvarucentrerad. Konstruktören utvecklar först en modell av hårdvaran i C++. Den verifieras med vanliga debuggverktyg för C++. Därefter specificeras FPGA-plattformen och gränssnitten (PCI, Ethernet, DDR QDR osv) vartefter hårdvaran kompileras och byggs.

För att denna process ska fungera sömlöst måste hårdvaran som skapats ha exakt samma funktion som den ursprungliga mjukvarumodellen. Detta innebär att modellen måste vara deterministisk, så att den ger samma resultat som hårdvaran oberoende hur snabbt hårdvaruimplementeringen körs. Tyvärr har de flesta parallella system icke deterministisk exekvering. Flertrådad mjukvaruexekvering beror exempelvis av CPU, operativsystemet och ▶

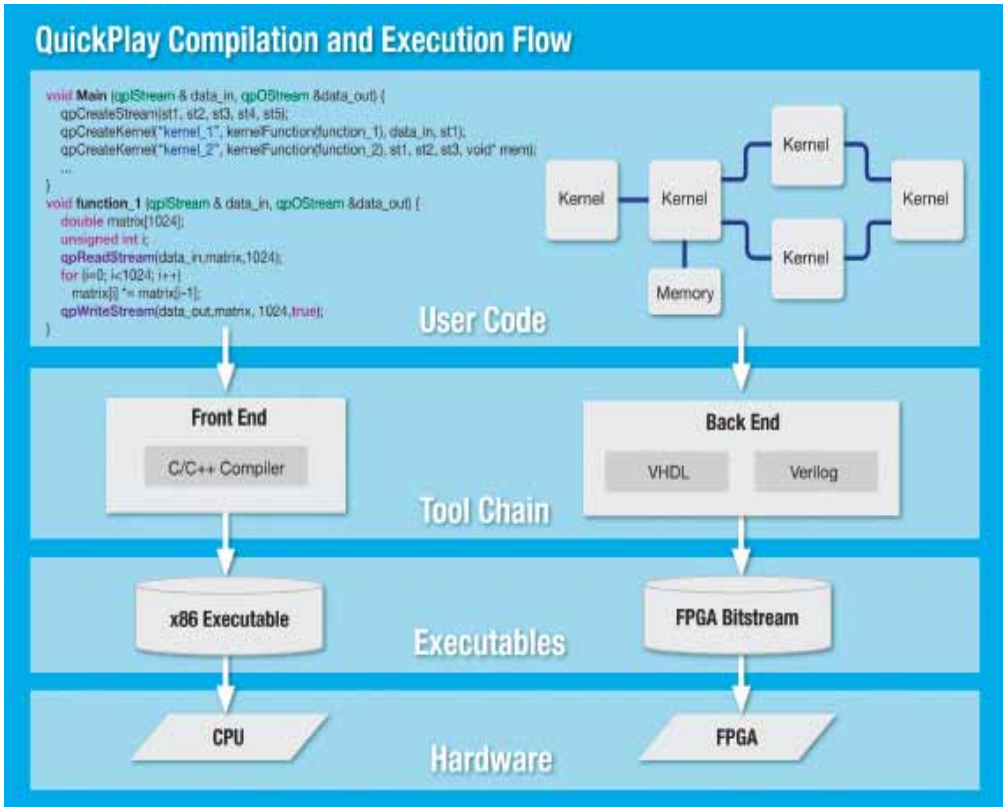
icke relaterade processer som körs på samma host. Flera körningar av samma flertrådiga program kan ge olika beteenden.

Denna typ av icke-determinism i hårdvaran skulle kräva felsökning av själva hårdvaran. Detta motverkar syftet med ett verktyg som riktar sig till mjukvaruutvecklare, men QuickPlays dataflödesmodell garanterar deterministisk exekvering oavsett hårdvaran. Modellen består av samtidiga funktioner, så kallade kernels. Dessa kommunicera med strömmande kanaler, vilket väl överensstämmer med hur en mjukvaruutvecklare skulle kunna skissa ett program på en whiteboard. Innehållet i varje kernel kan vara godtycklig kod i C/C++, ip från tredjepart eller till och med HDL-kod.

Om vi tar en närmare titt på konstruktionsflödet:

Steg 1: Ren mjukvarukonstruktion. FPGA-konstruktionen skapas genom att lägga till och ansluta kernels i C och specificera kommunikationskanalerna via utvecklingsmiljön på värddatorn. QuickPlay IDE erbjuder ett C/C++-bibliotek och API för att skapa kernels, strömmar, strömmande portar och minnesportar, samt för att läsa och skriva till och från strömmande portar och minnesportar.

Steg 2: Funktionell verifiering garanterar att mjukvarumodellen fungerar korrekt. Modellen kompileras i datorn och körs av testprogram som skickar data till ingångarna för att verifiera hur korrekta utgångarna är.

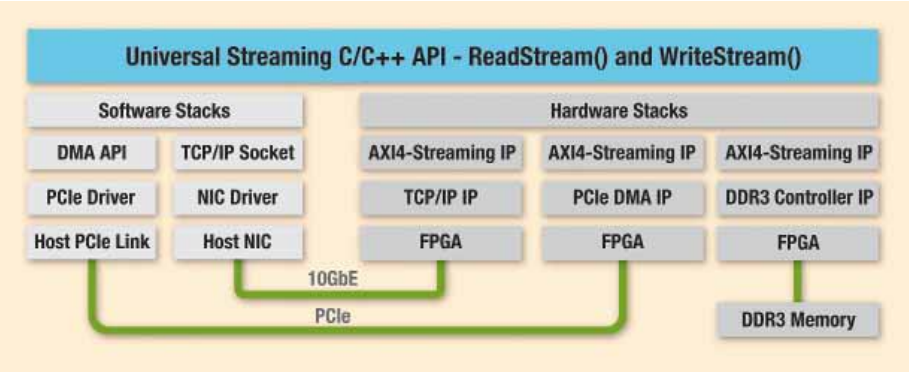


Figur 3. QuickPlays konstruktionsflöde är okomplicerat.

Steg 3: FPGA-hårdvara skapas från mjukvarumodellen. I detta skede väljs FPGA:n – men också de fysiska gränssnitten och protokollen som ska mappa konstruktionens in- och utgångar – med enkla rullgardinsmenyer.

Steg 4: Systemutförandet (system execution) liknar funktionell verifiering förutom att FPGA-konstruktionen körs på det valda FPGA-kortet medan värdapplikationen fortfarande körs i mjukvara. Verkligt data kan strömmas in och ut från FPGA-kortet. Många fler tester kan köras i detta steg än under funktionell verifiering.

Steg 5: System-debug. Felsökning på hårdvarunivå är aldrig nödvändigt, även om ett fel upptäcks efter att en funktion implemnterats i hårdvara. Detta eftersom QuickPlay garanterar funktionell likvärdighet mellan mjukvarumodellen och hårdvaruimplementationen. Alla fel i hårdvaruversionen finns också i programversionen.



Figur 4. Hård- och mjukvarustackar.

Steg 6: (tillval) Optimering. Infrastrukturen byggd av QuickPlay är mycket effektiv när det gäller prestanda och utnyttjande. Den övergripande kvaliteten på konstruktionen beror däremot på kvaliteten på de av användaren skapade kernels. Generisk C-kod kommer inte att ge den mest effektiva hårdvaruimplementeringen utan det finns tekniker och riktlinjer som mjukvaruutvecklare kan följa för att avsevärt förbättra effektiviteten i den HLS-genererade koden. Genom att använda Xilinx Vivado HLS eller genom att omkoda vissa kernels i HDL går det också att optimera ytterligare.

HÅRDVARAN SOM GENERERAS med QuickPlay är dessutom så effektiv att verktyget även blir intressant för hårdvaruingenjörer. De kan spara veckor, till och med månader, genom att låta QuickPlay ta hand om vardagliga konstruktionsuppgifter, medan de själva koncentrera sig på mervärden som de kan addera i form av kernels. ■