Altera:
OpenCL ger snabb utveckling och enkel flytt

Deshanand Singh
Supervising Principal Engineer, Altera

Deshanand Singh är ansvarig för Alteras projekt OpenCL-to-FPGA som utvecklar högnivåverktyg för OpenCL. Innan dess arbetade hans grupp i Toronto med olika optimeringsalgoritmer till utvecklingsverktyget Quartus II. Deshanand Singh har doktorerat vid universitetet i Toronto på tekniker för att möta tidskrav vid FPGA-design. Han har över 50 patent på området.

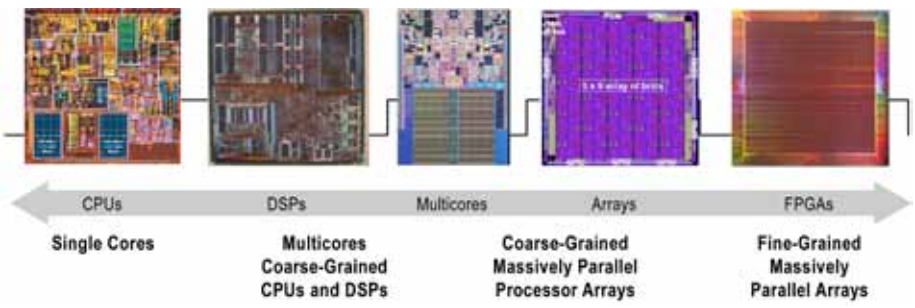
För inte så länge sedan fanns det två sätt att programmera. Det ena kan representeras av enkärniga processorer och DSP:er. De programmerades med mjukvara som bestod av en lista med instruktioner som skulle exekveras. Instruktionerna skapades så att programmeraren upplevde dem som sekventiella även om en avancerad processor kunde stuva om dem för att hitta uppgifter som kunde utföras parallellt.

Det andra sättet representeras av FPGA:erna. De har hårdvara som programmeras för att exekvera parallellt. En konstruktör som arbetar med en FPGA skapar en finmaskig applikation som bearbetar data parallellt.

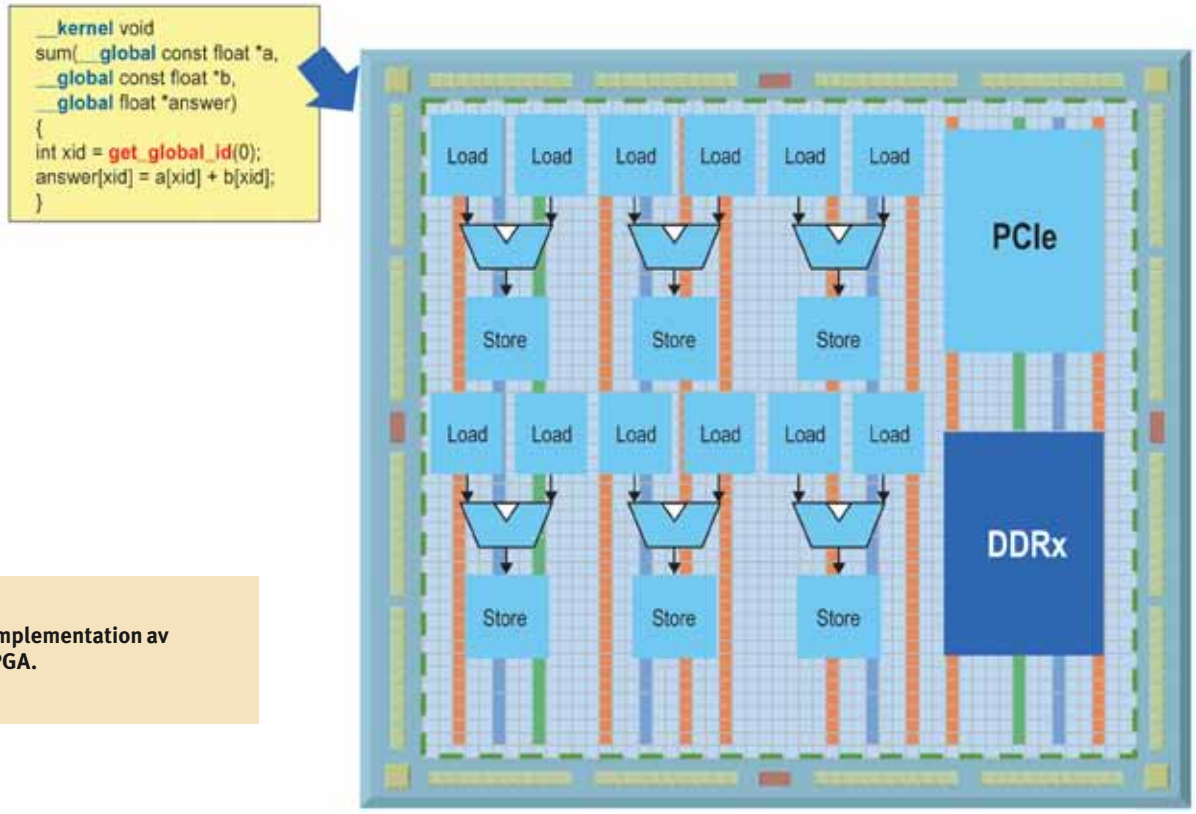
Under många år har de här två programmeringsmetoderna existerat sida vid sida men använts för helt olika tillämpningar. På sistone har teknikutvecklingen gynnat metoder som både är programmerbara och parallella.

Den andra trenden som programmerbara enheter förlitat sig på är komplex hårdvara som kan extrahera instruktioner som går att utföra parallellt ur en följd av sekventiella program. En enkärnig arkitektur tar in en ström med instruktioner och exekverar dem på en komponent som kan ha många parallella funktionsblock. En stor andel av processorns hårdvara måste vara dedicerad till att göra det här dynamiskt. Dessutom måste hårdvaran försöka kompensera för fördröjningar förorsakade av minnet.

Normalt skapar programmerarna program utan att fundera på den underliggande minneshierarkin som om minnet vore ett enda platt och lika snabbt lager. Processorn får ta hand om den fysiska verkligheten med långa fördröjningar och begränsad datatakt till externa minnen. För att kunna försörja funktionsblocken med data måste processorn hämta dessa på spekulation och lägga in dem i de interna cacheminnena så att de är snabbt åtkomliga. Efter många decennier med olika förbättringar av tekniken ger de idag bara små förbättringar.



Figur 1
Trender för programmerbara och parallella tekniker.



Figur 2
Exempel på en implementation av OpenCL på en FPGA.

MED DE DE MINSKADE förbättringar som dessa två trender ger för konventionella processorarkitekturer i minnet är det intressant att titta på möjligheter som programmerbara kretsar ger. Fokus skiftar från att automatiskt extrahera parallella uppgifter under exekveringen av koden till att identifiera parallella trådar redan vid kodningen.

Det börjar komma parallella och flerkärniga kretsar där en större del av kiselytan används för beräkningen, inte som cache eller för att hitta parallella delar i koden. Det handlar om allt från generella flerkärniga processorer med två, fyra eller åtta kärnor till grafikprocessorer med hundratals enkla kärnor som är optimerade för parallell bearbetning av data. För att få hög prestanda på dessa flerkärniga komponenter måste programmeraren koda på ett parallellt sätt. Varje kärna måste få en uppgift så att alla kärnor kan samarbeta om en gemensam uppgift. Det här är exakt vad FPGA-utvecklare gör för att skapa en systemarkitektur.

För att täcka behovet av parallella program i den flerkärniga eran skapades OpenCL (Open Computing Language) i ett försök att få en standard som fungerar på olika plattformar. OpenCL erbjuder möjligheten att beskriva parallella algoritmer, som implementeras i en FPGA, på mycket högre nivåer än som är möjligt i hårdvarubeskrivande språk som VHDL eller Verilog. Även om det finns många verktyg för den här typen av högnivåbeskrivningar så har de alla haft samma problem: de tar in sekventiell C-kod och försöker omvandlar den till parallell HDL-kod. Det svåra ligger inte i att få fram HDL- mplementationen utan att extrahera trådar som är parallella och därmed ger bra prestanda i FPGA:n.

FPGA:n ligger på den extrema ytterkanten när det gäller parallellism, och ett misslyckande med att få fram den får större konsekvenser för prestanda än på andra komponenter.

OpenCL löser många av de här problemen genom att låta programmeraren specificera och styra parallellismen. OpenCL matchar betydligt bättre de parallella egenskaperna i en FPGA än sekventiella program i ren C-kod.

Applikationer baserade på OpenCL består av två delar. Värdprogrammet (OpenCL host) är en ren programrutin som är skriven i en standardversion av C/C++ och kan köras på vilken processor som helst. Det kan till exempel vara en mjuk processor som implementerats i FPGA:n, en hård Arm-processor eller en extern x86:a.

Vid en viss tidpunkt under exekveringen av programmet på värdprocessorn dyker det upp en funktion som tar mycket kraft och som kan tjäna på att accelereras på en parallell enhet, som en flerkärnig processor, en grafikprocessor, en FPGA eller nåt annat. Den här funktionen kallas OpenCL kernel. De här delarna skrivs i vanlig C-kod men de har med sig kunskap om parallellism och minneshierarkin. Exemplet i figur 2 utför vektoraddition på två vektorer, a och b, samtidigt som resultatet skrivs till en svarsvektor kallad ”answer”.

Parallella trådar bearbetar elementen i vektorn vilket gör att resultatet blir klart mycket snabbare när det accelereras i en komponent som har massvis med finkornig parallellism, som en FPGA. Värdprogrammet har tillgång till vanliga API:er som gör det möjligt att flytta data till FPGA:an, kör igång uppgiften och hämta resultatet.

I EN FPGA KAN uppgifterna utföras av dedicerad hårdvara med långa pipelines som av naturen är multitrådade.

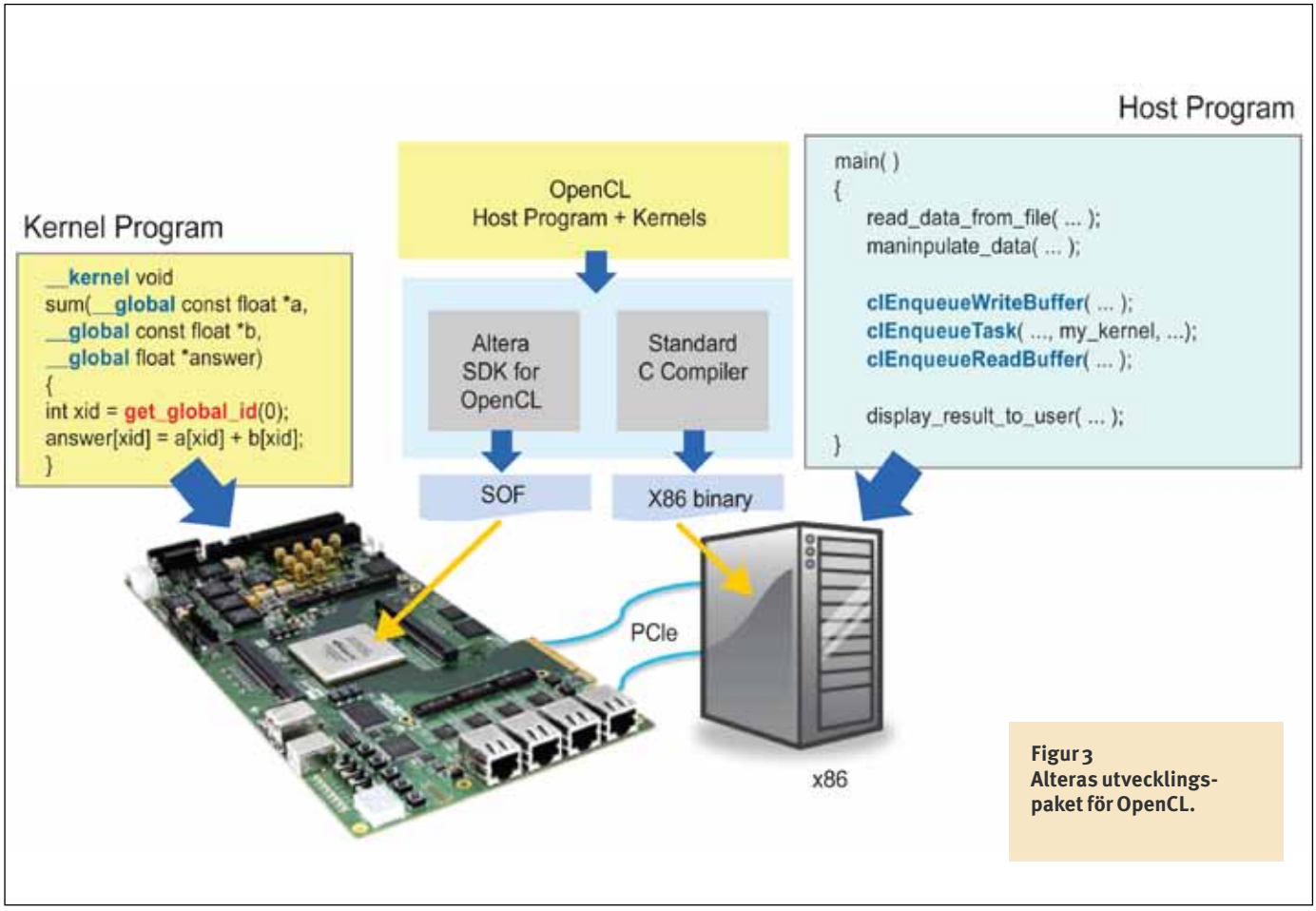
Var och en av dessa pipelines kan dupliceras många gånger för att ge ytterligare parallellism än vad om är möjligt med bara en pipeline. Att designa för FPGA:er med en OpenCL-beskrivning ger flera fördelar jämfört med traditionella metoder baserade på HDL. Utveckling för programmerbara kretsar börjar typiskt med en idé, därefter kodas en algoritm i ett högnivåspråk som C och sedan skapas instruktionsströmmen med en automatisk kompilator. Alteras utvecklingspaket för OpenCL erbjuder en utvecklingsmiljö som gör det enkelt att implementera applikationer i OpenCL i en FPGA.

DET HÄR ANGREPPSSÄTTET står i kontrast till traditionella utvecklingsmetoder som kräver att konstruktören skapar en ”cykel-för-cykel”-beskrivning av hårdvaran som sedan används för att implementera hans eller hennes algoritm. I det traditionell arbetssättet måste man skapa datavägar, tillståndsmaskiner som kontrollerar datavägarna, kopplingar till IP-kärnor på lägre nivåer via systemnivåverktyg och hantering av timingvillkor som förorsakas av krav från externa kopplingar.

Alteras utvecklingspaket för OpenCL gör alla dessa saker automatiskt åt konstruktören vilket gör att han eller hon kan fokusera på algoritmen och inte tidskrävande småsaker i hårdvaran.

När man designar på det här sättet går det enkelt att flytta konstruktionen till nya FPGA-familjer med bättre prestanda och högre kapacitet eftersom kompilatorn i OpenCL omvandlar högnivåbeskrivningen till pipelines om drar nytta av den nya FPGA:an.

Att använda OpenCL på en FPGA kan ge betydligt högre prestanda vid lägre effektförbrukning än vad som är möjligt med dagens hårdvaruarkitektur (processor, grafikprocessor, etc.). Dessutom erbjuder ett heterogent system (processor + FPGA) som programmeras med OpenCL en betydligt kortare utvecklingstid och därmed tid till marknaden jämfört med traditionell FPGA-utveckling med hårdvarubeskrivande språk som Verilog och VHDL. ■



Figur 3
Alteras utvecklingspaket för OpenCL.

Technical Papers
Jan Tångring
jan@etn.se
0734-17 13 09