Hyperregister ger fart åt FPGA:erna

Stoppa in Hyperregister på lämpliga ställen i konstruktionen så lyfter prestanda

Av Allan Davidson, Altera

Allan Davidson är ansvarig för marknadsföring och produktplanering när det gäller Alteras Stratixfamilj.
Han började på företaget 2006 och har över 20 års erfarenhet av programmerbar logik och FPGA:er.
Han har också arbetat med test och designat flera integrerade kretsar.

Prestanda för ett systems bestäms ofta av hur snabbt det kan skyffla data. I många fall, inklusive FPGA:er, är den vanligaste tekniken för att öka prestanda att göra de interna bussarna bredare och bredare. Det är inte ovanligt med 512 bitar, 1024 bitar eller ändå mer men tekniken slukar resurser i form av kiselyta och energi. Dessutom ökar komplexiteten när man ska designa komparatorer eller göra checksummor på så breda bussar.

I takt med att processgeometrierna fortsätter att minska ökar prestanda samtidigt som fördröjningarna i de interna busarna börjar dominera den totala fördröjningen i FPGA:an. Tyvärr måste arkitekturen förändras för att det ska gå att lösa problemen med de interna bussarna.

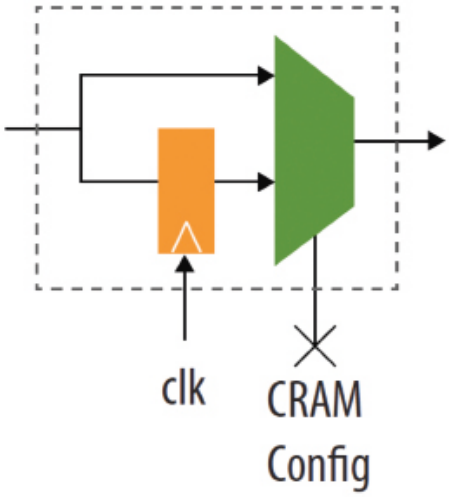
ALTERAS FPGA:ER i Stratix 10-familljen är ett exempel på en ny arkitektur kallad Hyperflex som använder ett innovativt angreppssätt för att lösa utmaningarna med de interna bussarna. Baserat på Intels 14 nm Tri-Gate-process använder arkitekturen vad som kan beskrivas som ”register överallt” som adderar ett Hyperregister som går att koppla förbi till varje segment som det går att routa till i FPGA-kärnan liksom på ingångarna till alla funktionsblock.

Figur 1 visar ett Hyperregister som det går att koppla förbi så att signalen går direkt till multiplexern eller så går den genom registret först. Multiplexern styrs med ett av FPGA:ans konfigurationsminnen (CRAM).

Figur 2 visar en liten del av logiknätet med nio adaptiva logikmoduler (ALM) och ledningarna som kopplar ihop dem. Hyperregistret är boxen som finns i de horisontella och vertikala skärningarna mellan segmenten som ska kopplas ihop.

För att maximera prestanda i en design som använder den här arkitekturen måste konstruktörerna använda en trestegsprocess baserat på retiming av register, pipelining och designoptimering. Hyperregistren gör det möjligt att använda vanliga designmetoder samtidigt som prestanda blir betydligt bättre än med konventionella Arm-arkitekturer. Tabell 1 visar möjliga förbättringar för varje steg i optimeringsprocessen. Att använda dessa register istället för de vanliga kallas Hyper-Retiming, Hyper-Pipelining och Hyper-Optimization.

Att göra om timingen för designen med hänsyn till Hyperregistren som adderats till förbindelserna kräver mycket lite arbete trots att det ger ett genomsnittligt prestandalyft på 1,4 gånger. Förbättringen är uppmätt på Alteras Stratix 10-kretsar och gäller jämfört med tidigare generationer. Genom att ta bort kritiska vägar flyttar Hyper-Retiming register från de adaptiva logikmodulerna till förbindelserna för att balansera fördröjningarna från register till register vilket medför att konstruktionen kan köra en snabbare klocka.



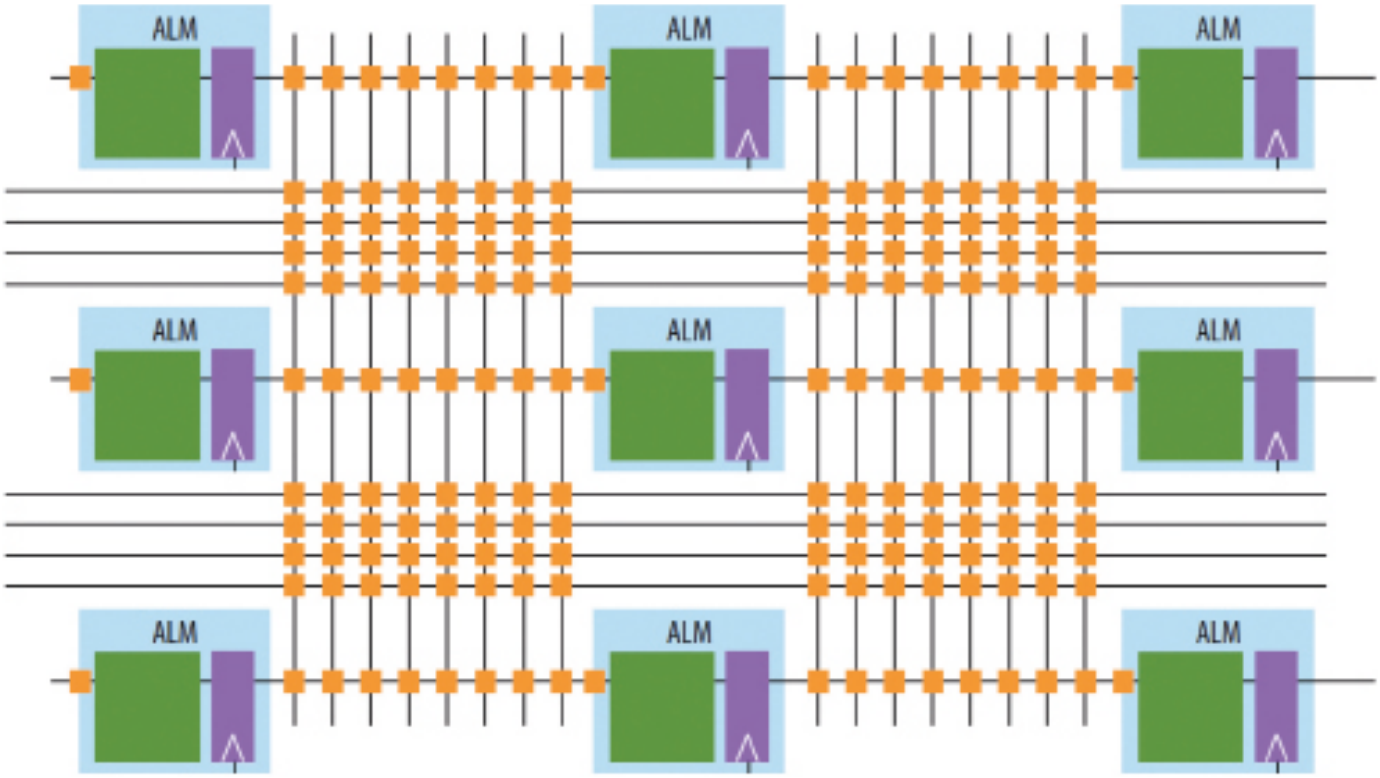
Figur 1. Ett Hyperregister som det går att koppla förbi (Bypassable).

HYPERREGISTER SITTER TÄTT i förbindelserna, de är finkorniga. Konventionell retiming kräver extra FPGA-logik och routingresurser plus att konstruktionen måste kompileras om, läggas ut och så krävs ny routing. I jämförelse kräver Hyper-Retiming inga extra FPGA-resurser och utförs efter place-and-route vilket ger ett avsevärt prestandalyft i kärnan med liten eller begränsad ansträngning.

När det gäller Hyper-Pipelining så arrangeras uppgifterna för ”röret” och timingen görs om med hjälp av Hyperregistren. Tekniken kräver minimalt med ansträngning och ger en genomsnittlig förbättring på 1,6

Steg	Fördelar med arkitekturen	Nödvändigt arbete	Prestandaökning (jämfört med äldre FPGA:er)
1	Hyper-Retiming	Ingen ändring eller små förändringar på RTL-nivå	1,4X
2	Hyper-Pipelining	Mer pipelining	1,6X
3	Hyper-Optimization	Beror på designen	2X eller mer

Tabell 1. En trestegsprocess för att maximera prestanda med hjälp av Hyperflexarkitekturen.



Figur 2. Hyperflexarkitekturen med register överallt.

gånger för Stratix 10 jämfört med tidigare generationer. Hyper-Pipelining eliminerar långa fördröjningar på grund av routingen genom att addera steg i form av ytterligare pipelines i förbindelserna mellan de adaptiva logikmodulerna vilket gör det möjlig att köra klockan med en högre hastighet. Återigen är Hyperregistren utspridda i förbindelserna vilket tillåter ett mycket precist val av var registret ska vara. Precis som med Hyper-Retiming använder inte heller Hyper-Pipelining någon extra FPGA-logik eller routingresurser och det görs efter place-and-route.

När datavägarna har snabbats upp med Hyper-Retiming och Hyper-Pipelining finns det konstruktioner som begränsas av styrlogiken. Det kan vara långa återkopplingsvägar och tillståndsmaskiner. För att få bättre prestanda är det nödvändigt att strukturera upp dessa logikdelar och använda funktionsmässigt ekvivalenta vägar för feed-forward eller pre-compute istället för långa återkopplingsvägar. Hyper-Optimization kräver lite mer arbete beroende på konstruktionen men resulterar i en genomsnittlig prestandavinst på två gånger eller mer för Alteras Stratix 10 jämfört med tidigare generation. I en konventionell arkitektur kallas den här processen för designoptimering. I Hyperflexarkitekturen går den under beteckningen Hyper-Optimization eftersom Hyperregistren använder de fördelar som Hyper-Retiming och Hyper-Pipelining ger jämfört med feed-forward och pre-compute.

FÖR ATT DRA FULL NYTTA av prestandaförbättringarna som Hyperflexarkitekturen och Hyperregistren ger är det viktigt att ha en optimerad verktygskedja. Altera har utvecklat en kraftfull uppsättning nya verktyg som är integrerade i Quartus II.

Det nya verktyget vägleder användaren genom optimeringsprocessen med att identifiera områden som begränsar prestanda. Verktyget identifierar också hur många pipelines som behövs och var de ska placeras samtidigt som det pekar ut kritiska vägar för styrsignalerna (exempelvis långa återkopplingsvägar).

Steget med Hyper-Retimer sker i slutet av designprocessen. Det optimerar place-and-route en andra gång med avseende på prestanda med hjälp av Hyperregistren för en optimal finkornighet av Hyper-Retiming. Det här steget gör det också möjligt för användaren att implementera Hyper-Pipelining på ett enklare sätt än vid konventionell pipelining. Kompileringsrapporten från Fast Forward identifierar vilken klockdomän som kan dra nytta av en pipeline och hur många steg som behövs. Efter det att designern modifierat RTL-koden och placerat ut det antalet steg i pipelinen som föreslagits vid kanten av varje klockdomän så placerar Hyper-Retimer automatiskt ut register i varje klockdomän på platser som optimerar prestanda. Den automatiska placeringen tillsammans med kompileringsrapporten från Fast Forward gör det enklare än någonsin att använda pipelining. ■

Figur 3. Designflödet för Hyperflexarkitekturen.