

Inledning

Av Leif Ahlbom

Här kommer årets vårnummer av SiliNova Process med intressanta artiklar kring:

- "Specman Elite 4.0.1" d.v.s. Verisity version 4 av Specman, som släpptes den 15 mars i år.

- "e Celerator: syntetiserarbar e", som demonstrerades på förra årets DAC och som nu Verisity lanserat. Ett verktyg som gör att verifieringen kan snabbas upp rejält.

- "Vad är .NET?" Microsoft lanserade nyligen infrastrukturprodukten .NET. Vi reder ut begreppen då .NET inbegriper allt från en plattform till utvecklingsmiljö och enskilda delar.

- "Bättre arkitektur med ASIP" Med detta nummer inleder vi en artikelserie där vi slår ett slag för kunskapskonsulting där SiliNova kan erbjuda specialkompetens inom bland annat områden som arkitekturdesign, systemdesign och verifiering. Vi börjar denna artikelserie med att beskriva hur användandet av kommersiella CPU:er kan förbättras.

Trots stor turbulens och osäkerhet inom telekomsektorn går livet vidare och vi närmar oss åter branschens viktigaste mäsas, dvs "DAC 2002".

Enligt vår tradition kommer vi att även i år besöka den och ge ut ett specialnummer om detta evenemang.

Trevlig läsning,



Leif A. är inte bara en flitig ledarskribent. Han är också VD för SiliNova AB.

Bättre arkitektur med ASIP

Av Tomas Jonsson

SiliNova erbjuder specialkompetens inom systemdesign, arkitekturdesign och verifiering. Det är inte ovanligt att en del inte inser sina behov inom dessa områden. Ofta finns det potential att förbättra system med avseende på tillverkningskostnad, kapacitet och kvalitet. Även om behovet är känt fordras det kunskap om hur ett system kan förändras utan att funktion och flexibilitet, - t ex programmerbarhet, - försämras.

Kunden är expert på sitt applikationsområde med avseende på funktion. SiliNova hjälper gärna till med sin kunskap om hur systemet/arkitekturen skall se ut eller förbättras. Likaså hjälper vi till med hur systemet kan implementeras och verifieras på alla nivåer. Vi är även experter på att definiera de miljöer som krävs för att verifieringen skall gå så snabbt och effektivt som möjligt. SiliNova har stor erfarenhet och kompetens inom design och verifiering av både små och stora system, från enkla RTL-simulator/testmiljöer till komplexa miljöer innehållande emulator, RTL-simulator och HW/SW samverifieringsverktyg. I SiliNova process kommer vi fortlöpande att ge tips på hur system kan förändras och hur de kan verifieras på bästa sätt.

CPU dyr på lång sikt

Vi börjar med att i denna artikel beskriva hur användandet av kommersiella CPU:er kan förbättras. För många systembyggare känns det ofta enklast och snabbast att börja konstruera med en eller flera CPU:er. Det är sedan lätt att de "hänger kvar" trots att det på lång sikt är en dyr lösning. Detta gäller inte minst system innefattande signalbehandling där man vill att algoritmer skall vara enkla att kunna ändra på. Ofta samsas ett avancerat RTOS, styrningen av systemet och algoritmerna i en och samma CPU.

En alternativ lösning skulle kunna vara att konstruera en egen CPU. Det är dock en ganska avancerad uppgift både när det gäller implementeringen och val av utvecklingsverktyg.

ASIP

En bättre lösning är att låta RTOS och styrningen ligga i en billig kommersiell CPU, och de mer beräkningskrävande algoritmerna vara implementerade i en ASIP (Application-Specific-Instruction-Set-Processors). Om man har ett system uppbyggt med flera CPU:er och de kostar för mycket eller har för hög effektförbrukning kan valda funktioner flyttas till en ASIP. På så sätt minskas CPU-behovet eller till och med elimineras helt. Tillverkningskostnaden minskas eller hålls konstant även när nya funktioner läggs till. Ett system som är uppbyggt av hårdkodade algoritmer kan göras mer flexibel med en ASIP-lösning. Både administration av minne och prediktering av prestanda är mycket enklare att utföra än med en CPU-lösning.

Bör vara mikrokodad

Det går att implementera en ASIP på ett enkelt sätt i en ASIC/FPGA. Idag kan man dessutom få med en billig kommersiell CPU på samma kisel i en sådan krets. En ASIC/FPGA-lösning kan klara av mycket avancerade funktioner, som att t ex innehålla ett RTOS, till en låg utvecklingskostnad. De tunga beräkningarna görs av den egenutvecklade ASIP:en medan administrationen sköts av CPU:n.

För att en ASIP skall vara enkel att implementera och för att man skall slippa utveckla avancerade kompilatorer etc. bör den vara mikrokodad med pipeline och minne. Bara de enklaste instruktionerna implementeras, såsom hopp, alu-operationer, load och store. Om RTOS saknas kan stöd för interrupt undvikas och schemaläggning av jobb kan göras på annat sätt. Kretsens mikrokod utvecklas i en mikrokod-assembler.

Enkel och billig verifiering

Verifieringen är enkel eftersom endast generella mekanismer implementeras i HW. Med fördel används ett genereringsverktyg med random-stimuli såsom

fortsättning på nästa sida

Specman. Algoritmerna verifieras i en HW-simulator. Detta gör slutverifieringen av algoritmerna enkel och billig innan hårdvaran släpps. Även på algoritmnivå är all HW synlig i simulatören (även interna register) vilket gör avlusning enkel.

Slutligen sammanfattas fördelarna med en ASIP:

- Billigare tillverkningskostnad.
- Enkelt att verifiera systemet både på HW och på algoritmnivå.
- Lägre effektförbrukning jämfört med en kommersiell CPU.
- Ger en programmerbar lösning jämfört med hårdkodade algoritmer i HW. Men behåller de hårdkodade algoritmernas fördelar (pris, kiselyta och effekt)

Kontakta SiliNova för att få att veta mer om vilka möjligheter en ASIP-lösning kan ge. Vi hjälper till med utredningar, kunskapsöverföring, design och verifiering. Allt för att du ska få en flygande start i ditt projekt.

www.SiliNova.se



Tomas J. är väl bevandrad i allt inom ASIC utvecklingen. Han är speciellt intresserad av CPU arkitekturer.

Specman Elite 4.0.1

Av Patrik Jarl

Nu har Specman version 4 från Verisity blivit tillgänglig för allmänheten. Den 15/3 släpptes Specman Elite 4.0.1. SiliNova fick prova på betaversionen av fyran redan i slutet av förra året.

Den nya versionen innehåller en ny datavisare, som ska tydligare presentera ditt

data. Till skillnad mot den äldre version kan man i denna se definitioner av metoder och händelsebeskrivningar. I betaversionen som jag provade kunde man lätt vandra igenom sina datastrukturer och visa dess fält, typer och värden. Jämfört med den äldre versionen som man klicka fram data i olika fönster var denna mer konsistent, nackdelen var att den var långsammare, vilket dock kan berott på att betakoden innehöll extra avlusningsinformation. När det gäller händelsebeskrivningar saknade jag lite information som t.ex. flanktyp och samplingskälla. Vidare har version 4.0.1 en ny genereringssavlusare som ska lättare hitta problem i samband med randvillkor och generering

Ny motor

Även en ny motor har införts för att evaluera temporära uttryck, som ska ge bättre prestanda än den tidigare. I den testbänk jag provade kunde betaversion inte motionera så många temporära uttryck så därför kunde jag inte mäta någon direkt prestandaökning.

Det finns dock ingen förändring i varken syntax eller semantiken av de temporära uttrycken.

När det gäller täckningsanalys har man infört stöd för att importera från en annan Verisity-produkt, SureCov. Som ni säkert vet kan man kunde man redan tidigare utöka täckningsgrupper, nu har man förbättrat detta stöd med "using also"-konstruktorn som kan användas i "cover event-typ using also [täckningsparameter]" för att förändra täckningsgruppsparametrar. På så sätt tillåts man att använda olika täckningsgruppsparametrar i olika testfall eller i olika subtyper.

Bättre täckningsanalys

Man har också infört stöd att göra täckningsanalys per instans, vilket kan väldigt intressant om man har två eller flera likadana moduler som hanterar olika flöden som t.ex. portar i en växel.

Nu finns det också stöd för att rangordna tester baserat på deras täckningsgrad så man kan identifiera deluppsättningar av tester som resulterar i bästa täckningsgrad. Med rangordningen kan man t.ex. identifiera redundanta tester.

I själva språket *e* så har det skett en förtydning av like/when arven, t.ex. kan man addera funktionalitet av en metod i föräldren trots att samma metod har blivit förändrad i en av like-barnen.

Utökad återvinning

När det gäller minnehanteringen sägs denna vara förändrad så att oavvänt minne frigörs inom Specman-tick, under laddning och kompilering. Dock finns fortfarande begränsningen att minneåtervinningen inte är aktiv i postgenerate-fasen.

Typkonvertering

När det gäller typkonvertering har denna förbättrats för bland annat mellan nycklade och vanliga listor samt uppräkningsbara typer och boolean.

Sammanfattningsvis kan man säga att den nya version innebär relativt små förändringar, troligen är det nya datavisaren den mest uppenbara för gemene man. Personligen ser jag fram emot att hårdare utvärdera den nya evalueringsmotorn för temporära uttryck.

www.verisity.com

Vad är .NET?

Av Mikael Muresu

Det kan väl inte undgått någon att Microsoft lanserat sin infrastrukturprodukt .NET. SiliNovas Mikael Muresu reder ut begreppen.

Med .NET [dot net] kan man säga att många barn har fått samma efternamn. .NET inbegriper allt från en plattform, till utvecklingsmiljö och enskilda delar. Eller vad sägs om:

- .NET Framework
- Asp.NET
- VB.NET
- Visual Studio .NET
- Ado.NET

Det visar sig dock att trots att namnen känns igen från tidigare så är det delvis helt nya produkter med gamla förnamn. Jag tänkte här ge en snabb översikt av hur det hänger ihop och beskriva en del nya och gamla begrepp.

.NET Framework

.NET Framework är just ett ramverk som måste vara installerad på den maskin på vilken en .NET applikation är avsedd att köras eller utvecklas. Troligen kommer .NET Framework vara en naturlig del av nästa version av Microsofts operativsystem. I teorin kan applikationer skrivas med Notepad (anteckningar) och sedan kompileras med kompilatorn i .NET Framework, men i praktiken använder man Visual Studio .NET som utvecklingsmiljö.

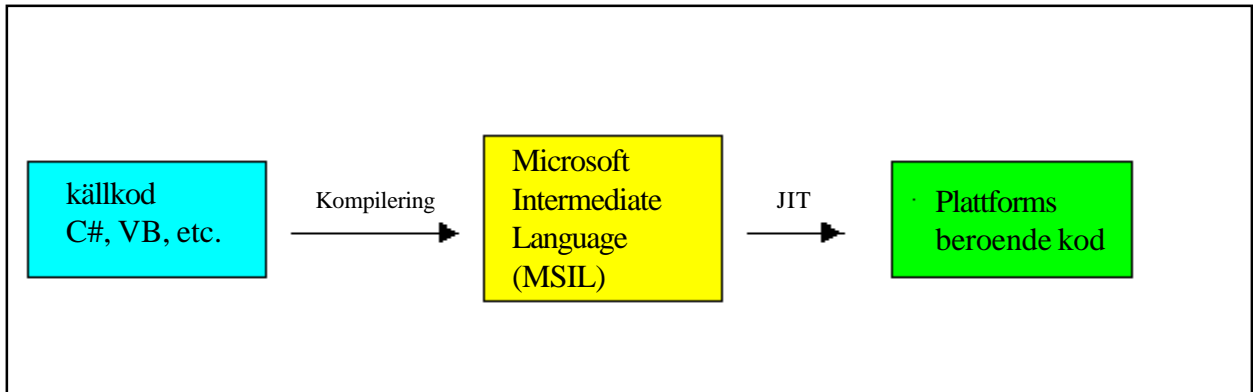
Visual Studio .NET och programutveckling (VS.NET) genererar (som tidigare) mycket med automatik, men är nu mer kraftfull och enkel att arbeta med. Microsoft har lyckats bra med att låta programutvecklaren koncentrera sig på att lösa de logiska problemen istället för bruset runt omkring.

MSIL assembly

För samtliga programmeringsspråk i .NET genereras vid kompileringstillfället något som heter assemblies, i vilka programkoden har översatts till Microsoft Intermediate Language (MSIL). När programmet körs för första gången så tar Common Language Routine (CLR) och Just In Time (JIT) och kompilerar koden till, för den avsedda maskinen, korrekt binärkod. CLR är en del av .NET Framework.

ter via XML-servers och sedan låta dem prata med varandra via en gemensam applikation.

Den här idé är inte ny, men med hjälp av VS.NET är det enkelt att bygga både .NET Web Services-servrar och Windows-plattformar för att sy ihop funktionalitet med XML-servrar på olika plattformar. Ett problem kan vara själva upp- och nedpackningen av XML-data som skickas,



För Windows-utvecklare: Det gamla Windows API:et med dialoger och olika typer av fönster av Windows Forms. Vill man skapa en applikation så använder man Windows Forms och sedan bestäms vilka egenskaper som just detta fönster ska ha (MDI, SDI, modal m.m.). Detta gör det även intuitivt enkelt att utveckla webblösningar eftersom motsvarighet finns i Web Forms (ASP.NET).

men används VS.NET för utveckling så är denna paketering transparent och utvecklaren behöver inte lägga ner tid på skapande av XML data. Det är en naturlig del i slutprodukten.

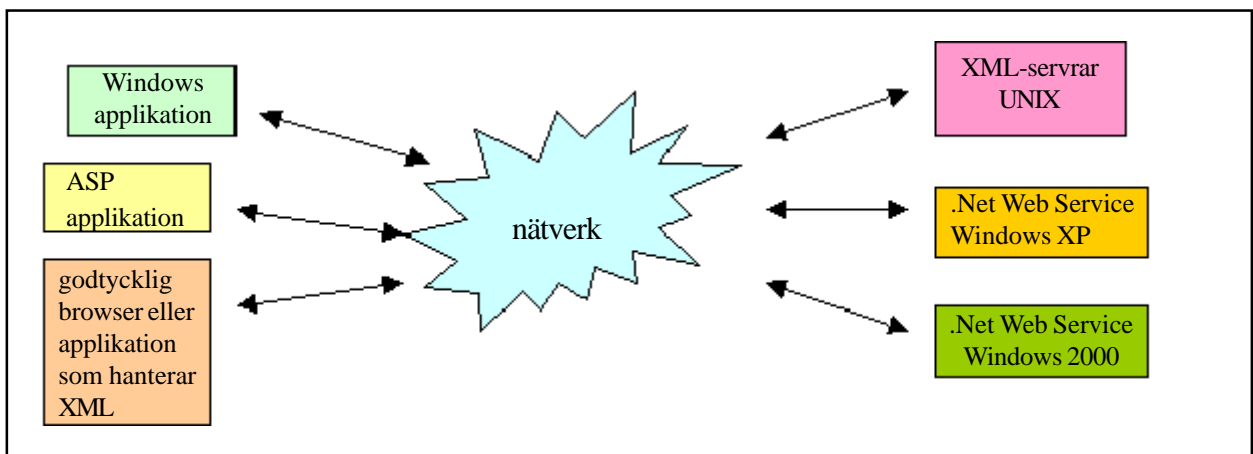
www.microsoft.com



Mikael M. är en mycket erfaren konstruktör inom systemutveckling- och objektorienterad mjukvara.

.NET Web tjänster

Microsoft har skapat ett "nytt" begrepp .NET Web Services som är ett sätt att paketera och exponera funktioner till klienter som kör på maskiner med vilket operativsystem som helst. Klienten använder XML och HTTP för att kommunicera med servern. Detta medför att integrationen av olika system kan implementeras genom att exponera tjäns-



eCelerator: syntetiserarbar e

Av Patrik Jarl

SiliNova fick se den demonstrerad på förra årets DAC, men nu har alltså Verisity lanserat eCelerator. Med detta verktyg kan verifieringen snabbas upp rejält, eftersom delar av e-testbänken kan syntetiseras och köras tillsammans med konstruktionen i en hårdvaruaccelerator eller en emulator.

Verifiering av dagens kretsar blir alltmer tidskrävande. Allt mindre linjebredder och mer komplexa teknologier ger högre kostnader att ta fram kretsar. Därför blir det mer och mer angeläget att kretsen fungerar vid produktion.

Samtidigt som allt mer satsas på verifiering, vill man inte förlänga utvecklingstiden utan snarare förkorta den och till hjälp finns verktyg som bl.a. emulatorer och hårdvaruacceleratorer. Om testobjektet kör i hög fart i en emulator måste testbänken kunna producera indata och kontrollera utdata i hög fart. Så dessa verktyg kräver mer från testbänkarna för att man ska få full utdelning.

Alla finesser tillgängliga

Tanken med eCelerator är att en del av testbänken ska kunna syntetiseras och placeras på någon typ av hårdvaruplattform såsom emulatorer eller accelerator. Detta kommer att ge en kraftig prestandaökning jämfört med vanligt mjukvarusimulering. eCelerator ska kunna ge full access till Specmans olika finesser. Den ska ge ett buffrat gränssnitt mellan mjukvarudelen och den accelererade testbänksdelen. Mjukvarudelen av testbänken exekverar e-koden på traditionellt sätt på en arbetsstation.

Frågan man kanske ställer sig: Ger detta någon tidsvinst? Min testbänk signalerar

ju med testobjektet varje klockcykel!

Verisity har analyserat detta problem och kommit fram till en lösning: Ett buffrat transaktionsbaserat gränssnitt, som man använder för att kontrollera per test om Specman ska skicka en eller tusen transaktioner till testobjektet.

10-50 ggr prestandaökning

eCelerator använder en innovativ syntes metod för att omvandla e-kod så den går att använda för hårdvaruacceleration. Genom att flytta de beräkningsintensiva delarna till hårdvara kan man få en signifikant prestandaökning i verifieringen. Man räknar med 10 till 50 ggr. Eftersom en del av testbänken är kvar på din arbetsstation kan man fortfarande komma åt de Specman stödda funktionerna. eCelerator accepterar en relativt stor uppsättning av verifieringsspråket e och låter dig skapa tillståndsmaskiner, protokollinspektörer och monitorer som är syntetiserbara och kan placeras köras på t ex en hårdvaruaccelerator.

Det transaktionsbaserade gränssnittet tillåter också full tillgänglighet till den syntetiserade testbänksdelen. Genom att den är buffrad minskar behovet av accesser varje klock-cykel och möjliggör att hårdvaruenheter får köra i full fart utan att hela tiden bli avbruten och förlora prestanda.

Bästa av två världar

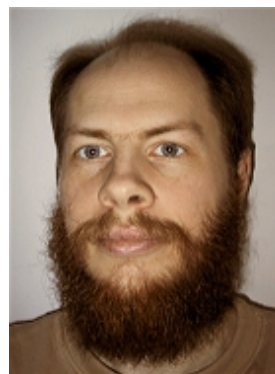
En annan fördel är att samma testbänk går att använda för ren mjukvarubaserad simulering som den hårdvaruaccelererede med fullständig konsistens. Detta möjliggör att man tidigare kan komma igång med emulering i verifikationsprocessen.

Man har på så sätt fått de bästa av två världar, generera stimuli med hjälp av randvillkor, med funktionell täckningsanalys och monitorer tillsammans med hårdvaruaccelerators prestanda.

www.verisity.com

För er som är intresserade av att syntetisera e-kod.

<http://www-ti.informatik.uni-tuebingen.de/~oase/Publications/oohsvlSSS2001.pdf>



Patrik J. är inte bara en flitig artikelskribent. Han är också kursledare för de Specmankurser som hålls hos ISS i Stockholmen. Mellan kurstillfällena jobbar han som konstruktör inom systemutveckling, med allt från mjukvara till hårdvara.



SiliNova Process

Redaktör: Magnus Hedlund

Ansvarig utgivare: Thomas Magnusson

Layout: Patrik Jarl

Adress:

SiliNova AB

Ekbacksvägen 28

SE - 168 69 Bromma

08 - 555 36 270 (telefon)

08 - 555 36 108 (fax)

silinova@silinova.se

<http://www.silinova.se>

Copyright © SiliNova AB

SiliNova är ett oberoende konsultföretag med fokus på ASIC.

Med över 70 års samlad erfarenhet av ASIC och FPGA är SiliNova ett av de ledande företagen inom branschen. Realtidssystem och objektorienterad programmering är andra områden inom vilka SiliNova har bred kunskap. Inom dessa områden erbjuder SiliNova tjänster för utveckling av kompletta system, antingen som uppdrag eller i samarbete med kunden.

SiliNova flyttar kontinuerligt fram sin position genom individuella utbildningsprogram och selektiv rekrytering.

Med vår erfarenhet i kombination med utbildning har vi som mål att samtliga uppdrag skall utföras effektivt, med hög kvalitet, till kundens fulla belåtenhet.