

Val av Debugger

För DSP-baserade plattformar med multipla kärnor

Utveckla inbyggda system blir alltmer komplicerat. Olika arkitekturers komplexitet ökar på grund av mer anpassningar och specialiseringar på respektive områden.

Valet av CPU för olika kombinationer av arkitektur blir därför allt viktigare beroende på syftet med målsystemet. Speciellt gäller detta när man studerar DSP baserade targets sammansatta av CPU:er med multipla kärnor och målsystem med olika typer av CPU – arkitekturer.

Debuggmiljön för DSP system

Idag används DSP:er för en mängd olika typer av applikationer, allt från konsumentprodukter som mobiltelefoner, till mer komplexa distribuerade system som nätverksplattformar.

Gemensamt är att det mest kritiska *och* minst förutsägbara momentet är själva debuggningsprocessen.

Det beror på en mängd olika variabler, allt från hårdvaru- och konfigurationsproblem till felaktigheter i programvara och applikation.

Trots allt finns det några generella frågor du som utvecklare måste ha övervägt och fått klart för dig innan du startar projekt baserade på DSP:

- Hur kontrollerar och hanterar du kärnorna? Ska du välja en lösning med en debugger som hanterar alla samtidigt eller en debugger för varje kärna?

- Stödjer debuggern multicore? Och hur svårt det är att implementera och underhålla systemet med just den debuggern?
- Hur handhar du lämpligast data mellan multipla kärnor för att på ett enkelt sätt hantera och bringa klarhet i logiken?
- Är systemet generellt? Det vill säga hanterar systemet dualcore på samma sätt som n stycken kärnor och hur många kärnor kan du instruentera samtidigt via debuggern, och fortfarande ha möjlighet att reda ut och hantera varje enskild del?

Synlighet

Många faktorer påverkar hur väl du kan instrumentera ditt system när du ska debugga DSP-applikationer. Val av verktyg, arbetssätt och teknologi har en betydande och i vissa fall helt avgörande roll för tiden du behöver spendera i integration, debugg och testfasen.

En av de svåraste bitarna i utveckling för inbyggda system är att snabbt skaffa sig bra förståelse för systemets beteende och helhet.

Det är här debuggern kommer in - ett verktyg som ger oss möjlighet att synliggöra systemet som helhet och på alla nivåer, från ren maskinvarunivå till högnivåapplikationer.

Huvuduppgiften för en debugger är att hitta oönskade "finesser". Detta kan du endast uppnå om du har bra översikt på ditt system, där allt är synligt.

[Fortsättning nästa sida »](#)

Ju bättre synlighet desto snabbare kan du detektera och åtgärda buggar.

Ur detta perspektiv är instrumenteringsförmågan hos debuggern helt avgörande för det val du gör, och kan ses som ett mått på kvalitet hos debuggern.

Oberoende debuggers

För att alla kunna initialisera och starta upp alla ingående CPU:er och därmed hela systemet, måste de som enbart har fokuserat sig på DSP-sidan lära sig debuggmiljön och de verktyg som krävs för att hantera de övriga arkitekturerna.

Det gör det inte bara onödigt komplicerat och ofta instabilt, utan kräver dessutom mycket tid och fördröjer den totala utvecklingsprocessen.

Traditionen har varit och är till viss del fortfarande att man köper helhetskonceptet från en leverantör, allt från utvecklingskort, kompilator och profileringsvitt till debugger. Hur blir det då med kvaliteten? Kan en leverantör verkligen vara ledande och bäst i sin bransch på alla dessa områden?

Lösningen på problemet ligger i debuggers som är arkitekturoberoende. Med denna typ av verktyg behöver du som utvecklare bara lära dig att hantera och underhålla ett system – ett system som passar alla arkitekturer.

Multipla kärnor

Multipla kärnor eller multicore technology definieras som en teknik där du utnyttjar kraften hos flera kärnor vilka du har integrerat i en mikroprocessor – CPU.

DSP-relaterade områden eller snarade target delas upp i två modeller:

- **Multipla kärnor** i en och samma CPU (multicore) eller;
- **Flera CPU:er** av olika arkitekturer (multiprocessors) oftast dessutom chip från olika leverantörer.

När designen på målsystemet involverar multicore-teknologi kommer debuggmiljön att bli mer komplex, då debuggern inte bara måste kunna hantera olika arkitekturer av chip, utan även CPU:er med flera kärnor.

Debuggern måste dessutom klara av skillnader i buss-strukturer, olika karaktäristik av dataflöden, blandning av datamatning, synkroniseringar, kontroll på CPU-komponenter/peripherals och skillnader i minnesbanker etc.

Återigen – arkitekturoberoende debuggers är om inte det enda valet, så i alla fall det bästa i jämförelse med arkitektur-tillverkarspecifika.

Inte bara för att uppsättningen av debuggmiljön förklaras och debuggprocessen underlättas med en enhetlig miljö, utan också för att kunna debugga dessa typer av komplexa målsystem.

När du använder Lauterbach för att debugga system med multicore och/eller multiprocessor-applikationer startar du bara en programvaruinstans för varje CPU eller kärna.

Genom denna instans kontrollerar du sedan respektive CPU och kärna, oavsett arkitektur (Fig. 1).

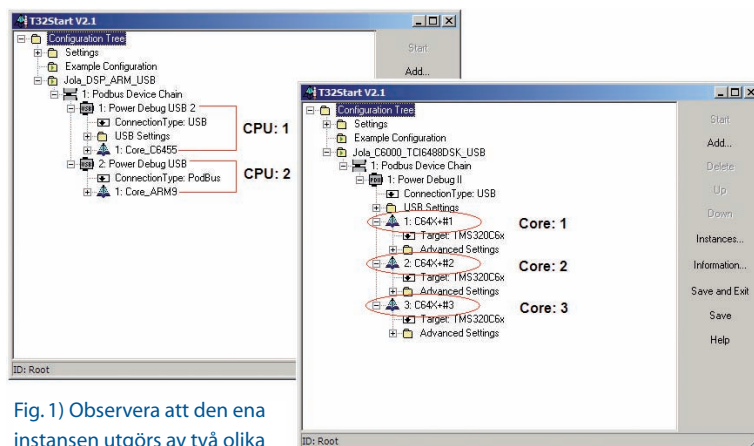


Fig. 1) Observera att den ena instansen utgörs av två olika CPU-arkitekturer (DSP - ARM) medan den andra hanterar en DSP med tre kärnor.

Konfigurationer för multicore och multiprocessor target

Uppsättning av system med multicore och multiprocessor kan generellt ske på två sätt:

- **Single device solution.** I ett system som detta har man en debuggmodul, fysik debug box som kontrollerar flera kärnor.
- **Multi device solution.** I detta system används flera debuggmoduler – en för varje kärna.

Oavsett uppsättning kommer varje kärna att ha en egen instans av Lauterbach programvara på host och det är genom denna instans du kontrollerar och debuggar kärnan.

Fortsättning sid. 6 »

Single device solution

I exemplet nedan (Fig. 2) ser du ett target utrustat med tre DSP-kärnor som hanteras från debuggerns sida. I detta fall använder vi oss av en debuggmodul, som sedan hanterar respektive kärna var för sig.

När du använder Single device solution accessas de till respektive kärna eller CPU som behöver göras, vilket sker direkt från JTAG-hanteraren inne i debuggmodulen.

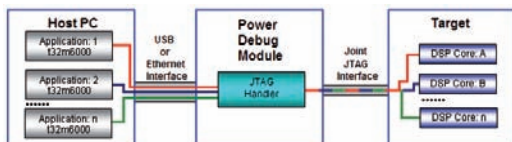


Fig. 2) Ingående target är i detta fall en DSP C6488 från TI med tre kärnor.

Multi device solution

Ett vanligt sätt att hantera skilda typer av CPU:er från olika arkitekturer är att koppla en power debugger till vardera CPU och sedan utnyttja gränssnittet Joint JTAG för kommunikation med respektive CPU.

Debuggmodulerna använder sedan ett standardiserat podbus-gränssnitt för intern kommunikation och synkronisering. Denna lösning har tillgång och hantering av Joint JTAG-gränssnittet för varje Lauterbach-instans internt.

Alltså, varje instans reserverar JTAG-porten för användning och förfogar över denna helt och hållet till dess att den sätter debugg-modulen i "Tri-State" läge (Wired-OR). Se exempel Fig. 3.

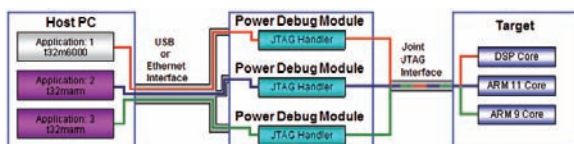


Fig. 3) Här har vi kopplat in tre debuggers som hanterar ett target utrustat med en DSP, ARM11 och en ARM9.

Multi device-lösningar kan användas för att nyttja kombination av samma, respektive separat användning av debuggmoduler till target med flera olika kärnor.

Det kräver en kombination av både Joint JTAG-gränssnitt och ett JTAG-gränssnitt (se Fig. 4).

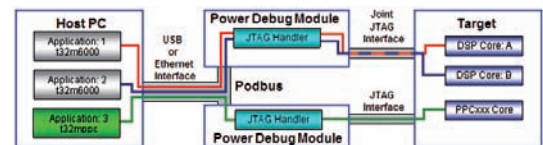


Fig. 4) Uppkoppling av ett debuggsystem för ett target med en "Dual-Core" DSP och en PowerPC. DSP- respektive PowerPC-sidan hanteras med varsin PowerDebug Module och synkningen mellan dessa använder sig av podbus-gränssnittet medan kärnorna i DSP:n hanteras via Joint JTAG Interfacet.

Att göra valet

När du kommer till stadiet att välja debugger för utveckling mot DSP-baserade målsystem tänk då på vilka nyckelfaktorer och områden som är viktiga att kunna styra och ha kontroll på i projektet.

Om faktorer som utvecklingsproduktivitet, långsiktig investering, arkitekturoberoende, samt en enhetlig miljö för hela målsystemet är kritiska bör valet av debugger komma då det är en av nyckelfaktorerna för projektets design och utvecklingsprocess.

Kvaliteten på en debugger är till syvende och sist dess förmåga att visualisera och framförallt instrumentera systemet.

Sammanfattning

Att debugga DSP-plattformar är långt ifrån trivialt – speciellt när de är baserade på multicore. När du använder Lauterbach-debuggern förenklar du denna process genom att modularisera systemet. Beroende på target-arkitektur kan man använda en eller flera debuggmoduler från Lauterbach.

För DSP-system påverkar inte debuggern realtidsegenskaperna vid exempelvis prestandamätningar och profileringar. Själva hanteringen av interna och externa minnen är helt transparent och till debuggern finns ett scriptspråk som ger dig möjlighet att skriva testsviter och profileringar, efter eget behov.

Lauterbach kan användas på system - baserade på alla kända processor- arkitekturer. Detta innebär att du behöver lära dig att hantera en och endast en debuggmiljö.

Joakim Larsson, M.Sc E.E.
DSP-integration
Nohau Solutions AB
telefon: 040-592200
joakim.larsson@nohau.se

