Från Matlab till C

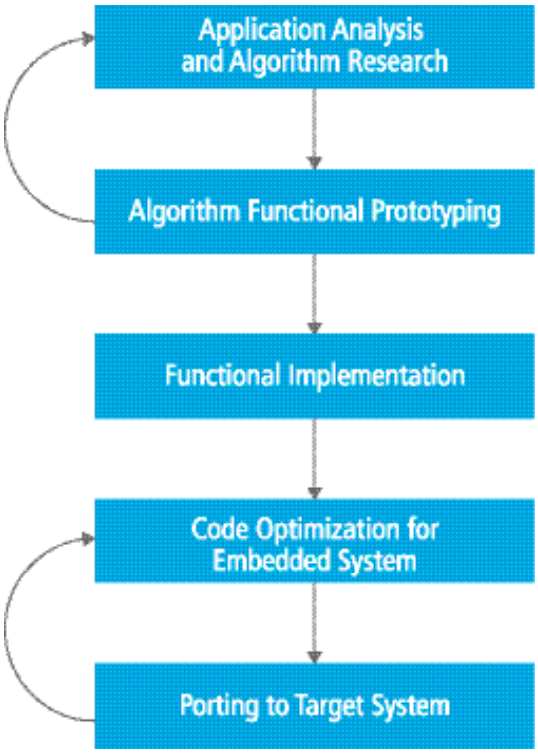
EXPERTARTIKEL

Så optimerar du en algoritm för maskinseende med Tensilicas DSP

Av Charles Qi och Han Lin, Cadence

Charles Qi är senior systemarkitekt på Cadence IP-grupp med ansvar för systemlösningar baserade på DSP:n Cadence Tensilica. Innan Cadence arbetade han på uppstartsföretaget Xingtera och utvecklade produkter för datakommunikation över elnätet i bostäder. Innan dess arbetade han på Broadcom.

Han Lin är designingenjör på Cadences IP-grupp med ansvar för att ta fram lösningar baserade på signalprocessorn Cadence Tensilica inklusive tillhörande IP för gränssnitten.



Figur 1. Flödet vid utveckling av tillämpningar för maskinseende.

På senare tid har maskinseende blivit en allt vanligare komponent i självkörande fordon. Men algoritmer för maskinseende är väldigt beräkningsintensiva och att använda dem i realtidstillämpningar – samtidigt som flexibiliteten bibehålls – kräver ofta kraftfulla signalprocessorer eller grafikprocessorer.

Det är en svår utmaning för utvecklare att mappa sina senaste algoritmer för maskinseende till prestandaoptimerad programvara som körs i realtidstillämpningar på en inbyggnadsplattform.

För att exemplifiera problemet har vi valt en algoritm för filföljning i ett ADAS-system. Vi hoppas det ska förklara mjukvaruutvecklingsflödet och demonstrera vilka utmaningar utvecklarna ställs inför när de ska skapa ett system med bra prestanda under begränsade systemresurser.

VI VISAR OCKSÅ hur ett prestandaoptimerat och funktionsrikt bibliotek för maskinseende kan användas för att korta utvecklingscykeln för mjukvara till bara några veckor genom att man övergår från generisk C-kod till DSP-optimerad kod som stödjer snabba och vektoriserade beräkningar i realtid.

Dessutom visar vi hur man kan optimera mjukvara för maskinseende med hjälp av en funktionsrik, kraftfull inbyggnads DSP, som Cadence Tensilicas Vision DSP.

Användningen av maskinseende i inbyggda system begränsas ofta av systemets hårdvaruresurser liksom av realtidskrav. Utvecklaren måste kunna optimera prestanda i sin tillämpning inom givna ramar. Prestandamått som genomströmning och noggrannhet måste balanseras mot andra parametrar som storlek på kod och data, storlek på minnet, tidsfördröjning och effektförbrukning.

SOM EXEMPEL i den här artikeln har vi valt en algoritm som används för att detektera vägmarkeringar och varna om bilen är på väg att byta fil. Funktionen är ett grundläggande ADAS-system i ett självkörande fordon.

I princip alla system som varnar för filbyte börjar med att detektera markeringarna i vägbanan och kan enkelt beskrivas så här:

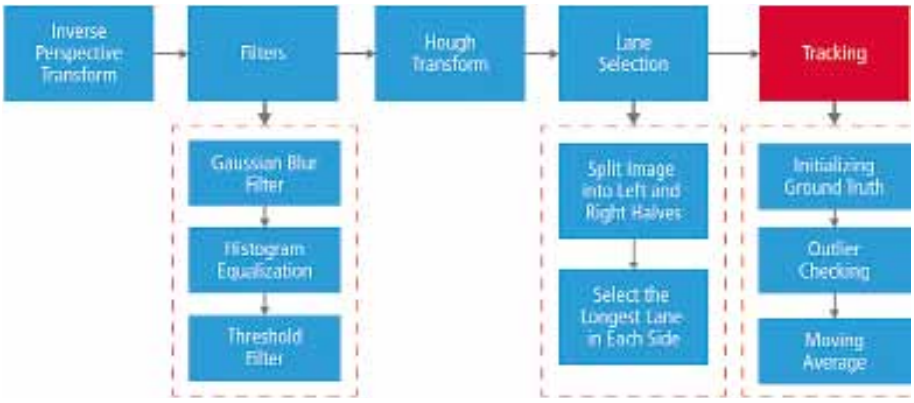
- Kartlägg vägbanan
- Ta bort extremvärden
- Lås på vägmarkeringarna och jämför med data från andra sensorsystem

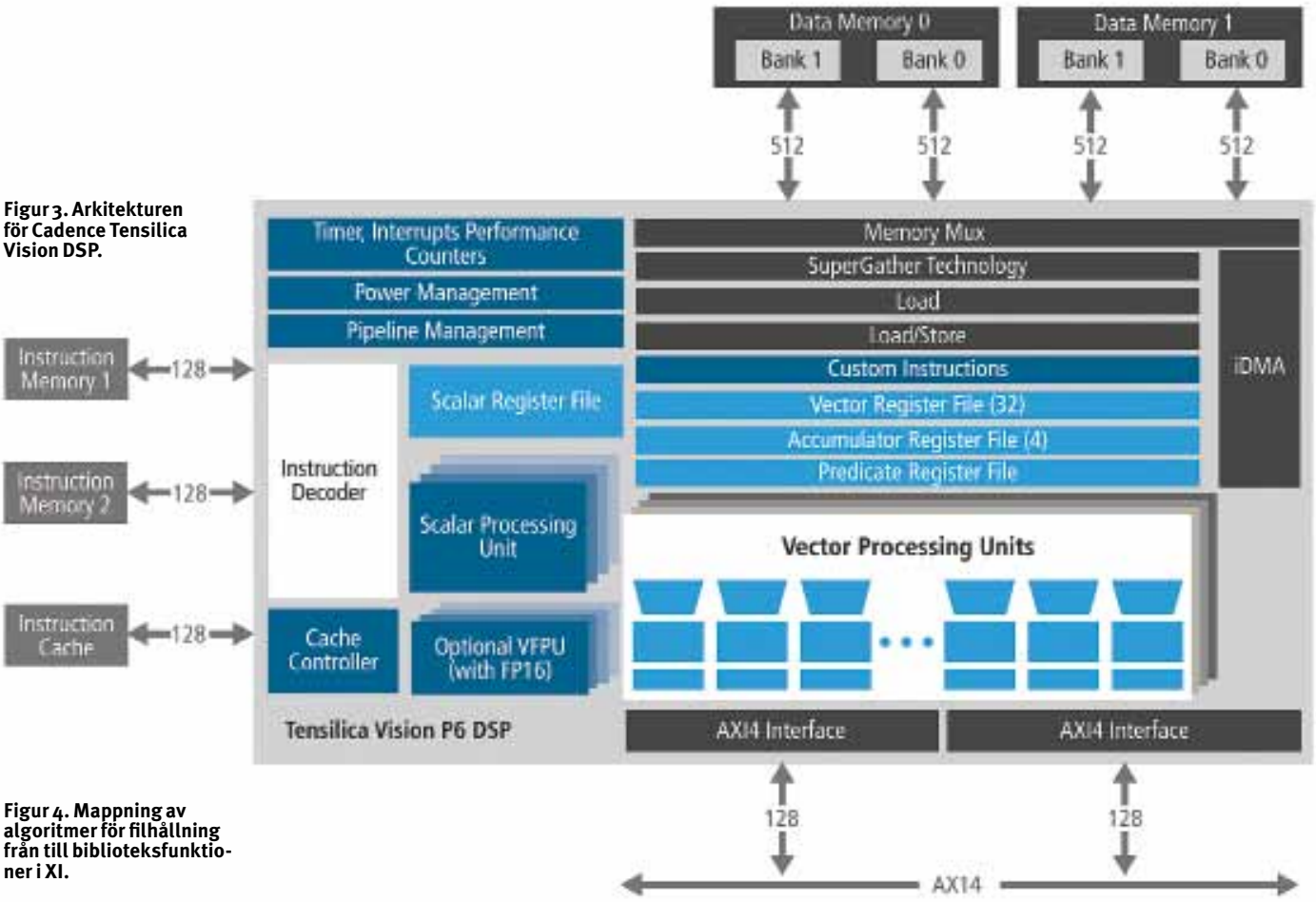
NOGGRANNHET OCH TILLFÖRLITLIGHET i ett system som varnar för filbyte beror på noggrannheten och robustheten i algoritmen som håller koll på vägmarkeringarna. Algoritmen måste kunna ta hänsyn till längden och utseendet på dessa, strukturen i vägen, olika ljusförhållanden, skuggor och objekt på vägbanan samtidigt som den i realtid följer fordonets rörelser.

Som framgår av de två figurerna spelar vi in en helt vanlig videobild från en kamera monterad på fordonet. Vi extraherar det intressanta området (ROI) – det som innehåller vägmarkeringarna. Sen utförs en invers perspektivtransformation baserad på kamerans parametrar för att få vägmarkeringarna parallella i bilden. Detta följs av några bildförbättringssteg inklusive filtrering, utjämning och tröskling för att minska bruset.

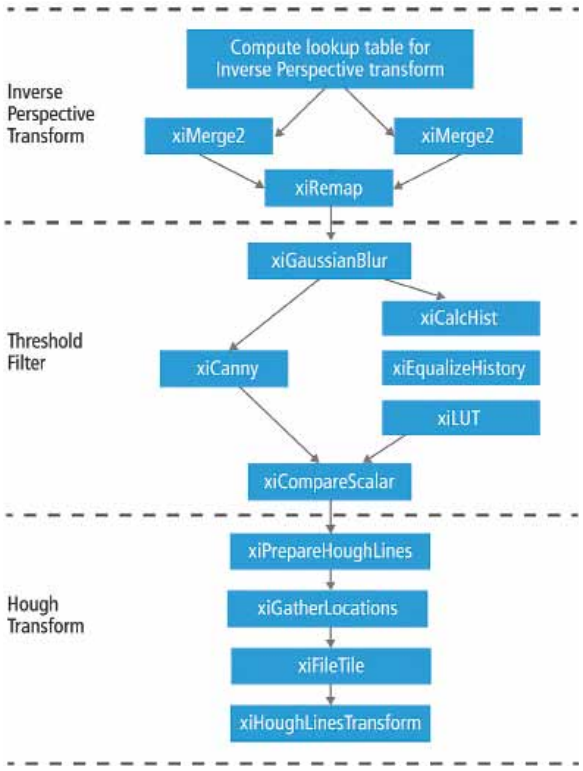
DÄREFTER DETEKTERAS och selekteras vägmarkeringarna med Houghtransformer. Slutligen förbättras resultatet genom att

Figur 2. Algoritmer med förbättrad robusthet för filhållning.





Figur 3. Arkitekturen för Cadence Tensilica Vision DSP.



Figur 4. Mappning av algoritmer för filhållning från till biblioteksfunktioner i XI.

vi jämför med tidsmässigt näraliggande bilder. Algoritmen har utvecklats i Matlab från Mathworks för att vi ska kunna verifiera funktionalitet och robusthet.

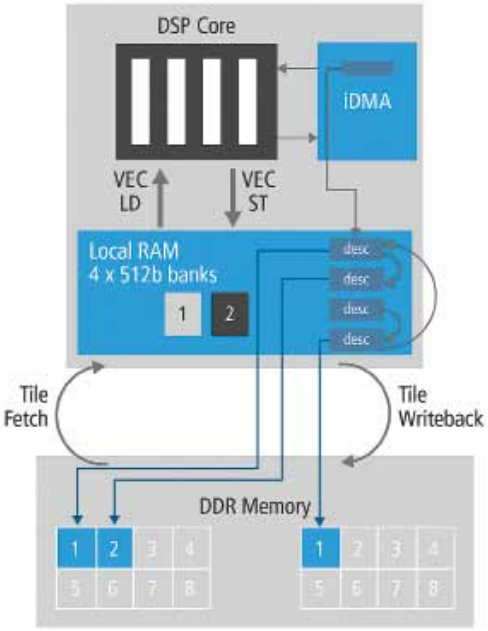
Medan det är relativt rättframt att utveckla algoritmen i Matlab är det betydligt svårare att portera den till ett realtidssystem. Beroende på komplexiteten i algoritmerna kan vanlig C-kod som sekventiellt bearbetar punkterna i bilden inte nå realtidsprestanda om den inte exekveras på en kraftfull server, och sådana kan av storleks- och effektskäl inte användas i inbyggda system.

FÖR ATT VISA att algoritmen fungerar med enklare hårdvara har vi valt ett signalprocessorblock i form av Tensilica Vision och implementerat det i ett FPGA-baserat prototypsystem.

Tensilica Vision DSP har en kraftfull arkitektur som stödjer 64-bitsinstruktioner och SIMD med vektoriserad laddning, lagring och beräkning. Arkitekturen kodar och skickar instruktioner i VLIW-format. Upp till fem instruktionsblock kan påbörjas och utföras parallellt på en enda klockcykel. Dessutom finns en instruktionsuppsättning optimerad för bildbehandling med parallella 8- och 16-bitarsoperationer på bildpunkter, vilket lyfter algoritmens prestanda.

BILDBEHANDLING KRÄVER hög minnesbandbredd på grund av storleken på bilderna. Signalprocessorn har ett internt minnesgränssnitt (iDMA) och två banker med lokalt och snabbt RAM för att lösa problemet.

Visserligen är DSP:n konstruerad för att stödja snabb bildbehandling, men att portera generell C-kod till att dra nytta av alla möjligheter är inte trivialt. DSP:n stöds av en sofistikerad kompilator som ▶



Figur 5.
iDMA Ping-pong Buffer Scheme for Tile Processing.

kan upptäcka och extrahera parallella delar i generisk C-kod. Trots detta är det ofta nödvändigt att utveckla bildbehandlingskärnor i handoptimerad C-kod för att maximera prestanda. Det finns ett stort antal färdigutvecklade sådana kärnor av produktionskvalitet i XI – ett OpenCV-liknande mjukvarubiblioteket – som du kan använda för att för att minska tiden det tar att portera och optimera.

I DET HÄR EXEMPLET har vi använt ett stort antal biblioteksfunktioner från XI för alla beräkningssteg i perspektivtransformeringen, filtreringen av bilden, utjämningen, för tröskelvärden, detektering av kanter och Houghtransformen. Användningen av XI-funktionerna i biblioteket minskar arbetet med att portera och optimera algoritmen för filföljning till signalprocessorn Vision P5/6. Realtidsprestanda kan uppnås i instruktionssimulatorn (ISS) inom 1–2 månader.

Genom hela filföljningsalgoritmen processas bilddata med en teknik kallad tiling med stöd av den interna minnesbussen iDMA. De breda SIMD-instruktionerna kräver att bilddata hämtas via vektoriserade instruktioner i snäva beräkningsloopar från snabba lokala minnen med breda bussar. Tiling-upplägget tillåter mindre delar av bilden att flyttas till det lokala RAM-minnet från det mycket långsammare systemminnet med hjälp av blockoperationer som stöds av iDMA:n.

FÖR ATT MINIMERA den påverkan accessen av minnet innebär använder implementationen också ping-pong-buffring. Den interna DMA:an är programmerad att hämta tile 1 från DDR-minnet och lägga det i pingbufferten i det lokala RAM-minnet. När det är klart börjar DSP:n att bearbeta ping-bufferten med tile 1. Under tiden hämtas tile 2 från DDR-minnet och läggs i pong-delen av det lokala RAM-minnet. Eftersom DMA:an gör den andra hämtningen parallellt med att DSP:n arbetar elimineras fördröjningen som minnet ger. Kombinationen av tiling och en DMA med ping-pong ger en förbättring på 15 till 20 gånger när algoritmen exekveras i prototypsystemets FPGA.

Vi har därmed visat att det går att ta fram en långt optimerad filföljningsimplementation och demonstrera den i realtid i prototyphårdvara med bara en bråkdel av den klockfrekvens signalprocessorn Vision är kapabel till i en systemkrets. ■