# Enkelt för makers att anvä

**Av Aaron Behman & Adam Taylor, Xilinx**

**Aaron Behman** arbetar med strategisk marknadsföring av vision-lösningar på Xilinx inklusive robotik, ADAS-system, maskinseende, övervakning och på det medicintekniska området. Innan han började på Xilinx arbetade han på Silicon Graphics och JDSU.

**Adam Taylor** är en välkänd expert på konstruktion och utveckling av inbyggda system och FPGA:er till tillämpningar från radar till säkerhetskritiska styrsystem men även bildbehandling och kryptografi.
Han har skrivit en rad artiklar om elektronik och FPGA:er inklusive 130 blogginlägg om Zynq.

*Lösningen stavas ZynqBerry, Pynq och Snickerdoodle.*


Att vara ”maker” är en populär hobby för många och inspirerar unga att studera teknik och matematik. Många av projekten innehåller inbyggnadsprocessorer, vanligen någon medlem ur Arduinofamiljen eller Raspberry Pi, för att ge systemet intelligens.

Det finns en utvecklingsmiljö till Arduino och Raspberry Pi inklusive mjukvarubibliotek, moduler och kodexempel som gör det enkelt för utvecklare att snabbt koppla upp olika periferienheter, från kameror till accelerometrar och motorer. Enkelheten bidrar till popularitet bland makers.

**FRAM TILL NYLIGEN** har makers ansett att system-FPGA:er varit svåra att använda, något som bara passar specialister. Detta är inte längre fallet med Zynq-baserade kort som ZynqBerry, Pynq och Snickerdoodle i kombination med mjukvarubaserade utvecklingsmetoder.

Korten använder Zynq 7000 från Xilinx med två Arm Cortex-A9-processorer plus programmerbar logik hämtad från FPGA-familjen Artix-7. Detta gör det möjligt att accelerera funktionen med FPGA-delen för att signifikant öka systemets prestanda. I traditionell utveckling skiljer man mellan programmerbar logik och mjukvaran där den förra har krävt specialiserad kunskap. Detta är inte längre fallet.

När dessa kort kopplas till den senaste utvecklingsmiljön, som klarar att definiera hela tillämpningen i mjukvara blir de mycket intressanta för den här användargruppen. Särskilt som dessa utvecklingsmiljöer gör det möjligt att utforska den programmerbara logiken utan att vara FPGA-specialist. Man får det bästa av bägge världar.

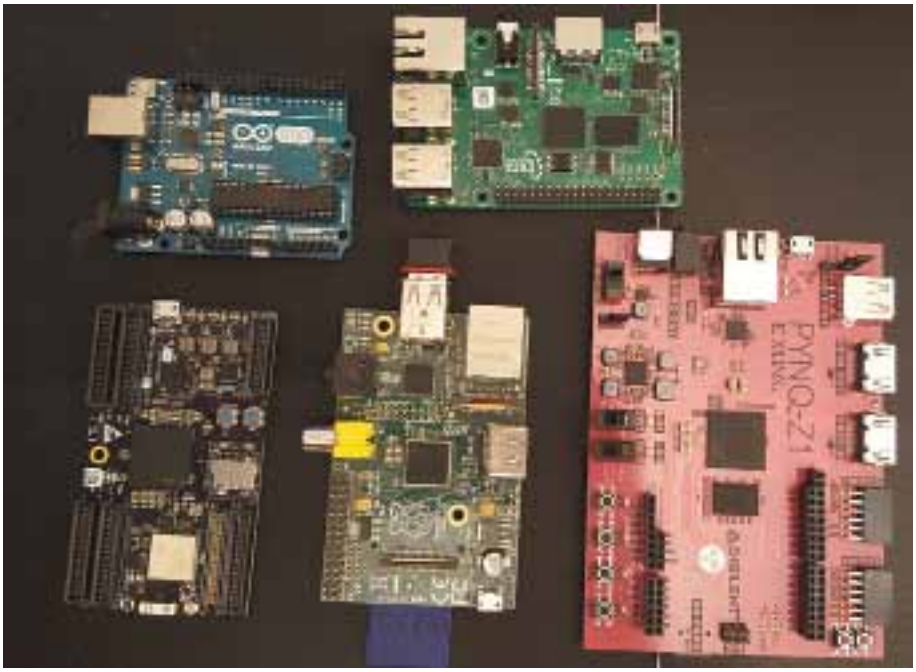
**DET FINNS TVÅ** utvecklingsmiljöer som kan användas för att skapa tillämpningar till Zynqbaserade utvecklingskort. Den första är SDSoC som är Eclipsebaserat och stöder C och C++, plus att man sömlöst kan flytta tillämpningen från A9-kärnorna till accelererade kärnor i programmerbar logik. SDCoC-miljön använder högnivåsyntes för att flytta utvalda C-funktioner till programmerbar logik. När högnivåsyntesen har tagit fram gränssnittet används ramverket för att integrera modulen med mjukvarutillämpningen. Förutom prestandalyftet som funktionen får när den exekveras i den programmerbar logik, så är processen transparent för användaren.

Att flytta funktioner mellan processor och programmerbar logiken är extremt enkelt och styrs i SDSoC via menyn i Project Overview.

SDSoC stödjer Linux – som är populärt bland makers – tillsammans med realtidsoperativsystemet FreeRTOS och raka programlösningar (bare metal approach).

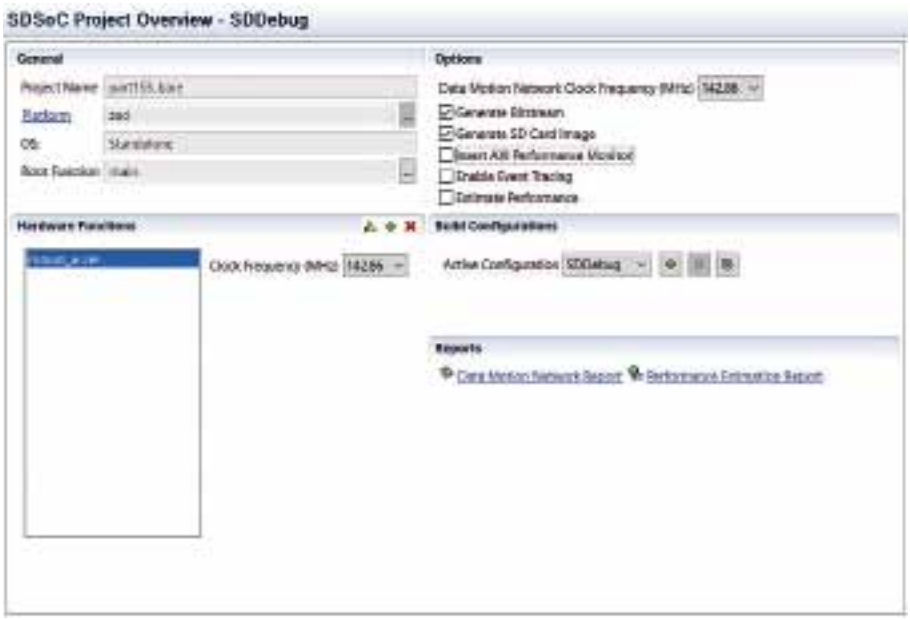
**DEN ANDRA LÖSNINGEN** kommer från Pynq som kommer med ett ramverk för utveckling baserat på Python och Juypter. Båda exekveras på en Linuxdistribution som körs på processorerna medan den programmerbara logiken har en overlay som definierar anslutningarna till periferienheterna på Pynq. Vidare finns funktioner för att direkt koppla in sig till perifierienheter via Python.

Pynq har två PMOD-gränssnitt med stöd för bland annat AD- och DA-omvandlare vilket underlättar integration med Python-tillämpningen. I Pynq laddas program-




**Utvecklingskort för makers. Från vänster till höger: Arduino, ZynqBerry, Pynq, Raspberry och Snickerdoodle.**

# nda programmerbar logik




**SDSoC Project Overview.**

merbar logik med en av overlayerna för hårdvaruacceleration. Förutom den grundläggande overlayen som följer med verktyget finns det andra i form av öppen källkod.

Användarna kan programmera Pynq genom att koppla kortet till en Juyptersever via ett webbgränssnitt. När den är ansluten till går det att utveckla och dokumentera tillämpningar i Python som sedan körs på Pynq.

Möjligheten att använda Python och ansluta det direkt till via ett PMOD-gränssnitt med hjälp av Python utgör en mycket kraftfull utvecklingsplattform.

Båda utvecklingsmetoderna ger möjlighet att använda öppenkodsramverk som OpenCV för video- och kameratillämpningar. Dessa tillämpningar kan använda webbkameror när Linuxdistributionen stöder video via USB eller specifika kameror som den till Raspberry Pi som också stöds av Zynqberry.

**OPENCV GÖR DET MÖJLIGT** att utveckla i antingen C, C++ eller Python. Med detta ramverk kan man snabbt och enkelt implementera komplexa bildbehandlingsalgoritmer som accelereras av programmerbar logik och därmed exekveras med betydligt högre prestanda.

Dessa tillämpningar kan processa bilder och detektera exempelvis objekt och ansikten.

När man ska implementera algoritmer för att upptäcka enkla objekt kan de köra Linux, Python och OpenCV på Zynq-baserade plattformar. Låt oss titta på vad som behövs för att implementera ett enkelt system med en webbkamera och OpenCV för målföljning. Algoritmen som implementeras är:

1. Ta in den första bilden från webbkameran. Den fungerar som bakgrundsreferens. Algoritmen detekterar allt som skiljer sig ifrån den.
2. Konvertera RGB-färgerna till gråskala. Det är en vanlig segmenteringsteknik för att skapa binära bilder. Segmenteringen av bilden omfattar olika teknik för att dela upp bilden i olika områden, ofta kallade superpixels, vilket gör det enklare att analysera innehållet. I tillämpningen används tröskelvärden för att skilja förgrund från bakgrund. På så sätt skapas en binär bild.
3. Applicera Gaussisk oskärpa på bilden. Många algoritmer för att detektera objekt eller kanter påverkas negativt av brus i bilden. Genom att göra bilden oskarp innan bearbetningen minskas bruset och tekniken används exempelvis vid detektering av kanter (bland annat med Laplacetransform för Gaussisk kantdetektering). Resultatet av operationen blir en referensbild som kan användas för att detektera förändringar.
4. Upprepa steg 1 till 3 för nästa bild från webbkameran.
5. Räkna ut den absoluta skillnaden mellan referensbilden och den senaste bilden.
6. Utför tröskling på den absoluta skillnaden för att skapa en binär bild.
7. Utför en morfisk operation för att förstärka skillnaderna
8. Plocka ut de konturer som blir kvar i den binära bilden, men ignorera de som har för liten yta.
9. Rita en box runt varje kontur som detekterats och skicka urspungsbilden till HDMI-utgången.

**UTVECKLARNA KAN KÖRA** den färdiga Pythonkoden direkt på Zynqberry eller i en Juypterdator på Pynq. De ser en bild liknande den i figuren som identifierar förändringarna relativt referensbilen och som markerar skillnaderna med boxar.

Det här exemplet visar både på kraften och enkelheten i Zynq när det gäller kameratillämpningar och vanliga ramverk i open source. ■

**Resultatet från en rörelseföljningstillämpning.**