SDSoC™ Environment

XILINX ALL PROGRAMMABLE™

SDSoC Development Environment

Introduction

With the advent of smarter systems and the drive towards 'The Internet of Things', increasing the connectedness of people and things, most new products now leverage SoC-based platforms which allow companies to bring products to market faster, increase system-level efficiency and, most importantly, allow for continuous innovation and product differentiation.

Design teams must choose carefully how to differentiate their products while still meeting ever increasing market requirements and stringent cost targets to maximize return on investment. True platform differentiation relies on a combination of both new software features as well as novel hardware features. Given the requirements to bring products to market faster that are truly differentiated at all levels, tools and environments are needed that enable both the software and hardware differentiation to be done with the completeness and ease of use found in traditional ASSP programming environment but without compromising on architecture or performance.

For hardware differentiation today, many platform developers utilize FPGAs for their 'any-to-any' connectivity where the programmable logic is used to interface the platform's processor(s) to the standard interfaces such as PCIe® and Ethernet. In addition, many systems use FPGA as a coprocessor for acceleration of critical functions and algorithms, where the programmable logic's parallel architecture can provide over 100X performance advantage compared to running on a standard processor.

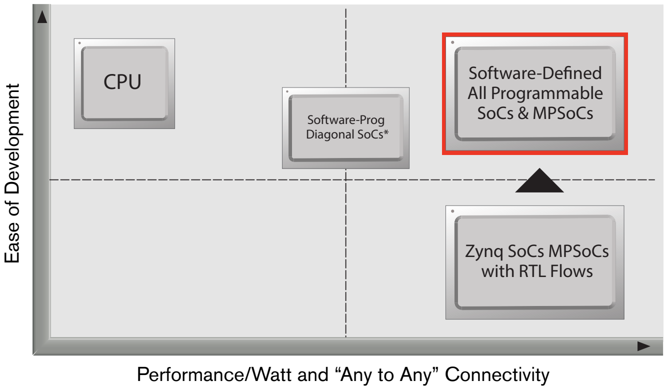
With the introduction of Zynq®-7000 All Programmable SoC in 2011 and now the new Zynq UltraScale+™ MPSoC, both combining powerful ARM®-based processing systems and programmable logic in advanced 28nm and 16nm nodes respectively, Xilinx is providing a proven alternative to traditional processors and domain-specific application SoCs. The Zynq SoCs and MPSoCs can increase system performance and lower system power all while reducing bill of material costs.

Introducing the Xilinx SDSoC Development Environment

The Zynq SoCs and MPSoCs are natural fit for design teams consisting of software and FPGA hardware engineers. Teams with limited or no hardware resources however have been challenged due to the RTL (VHDL or Verilog) development expertise needed to take full advantage of the benefit of the device. To resolve this challenge and enable more design teams to take advantage of Zynq devices, Xilinx has introduced SDSoC™, a new C/C++ development environment. The third member of the Xilinx® SDx™ family of development environments, the SDSoC development environment enables the broader community of embedded software developers to leverage the power of hardware and software 'all programmable' devices.

Software Defined Opportunity with SDSoC

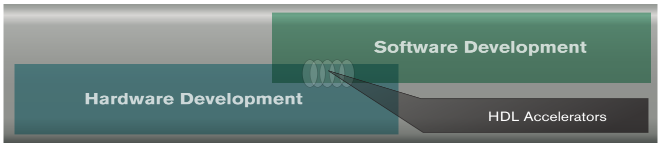
C/C++ Environment Delivering ASSP-like programming



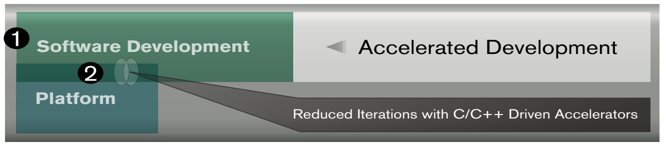
*Domain focused (e.g., image/video, SDR, etc.)
Note: Software programmable devices often paired with FPGA for connectivity and co-processing

The SDSoC Development Environment accelerates development of Zynq SoCs and MPSoCs in two ways. First software developers can get started sooner over a traditional hardware/software development flow by leveraging Xilinx, 3rd party or end user platforms. Second, SDSoC eliminates the churn between hardware and software teams typically required to take advantage of programmable logic as software accelerators, truly accelerating overall system development.

Traditional Development Schedule



Software Defined Development Schedule

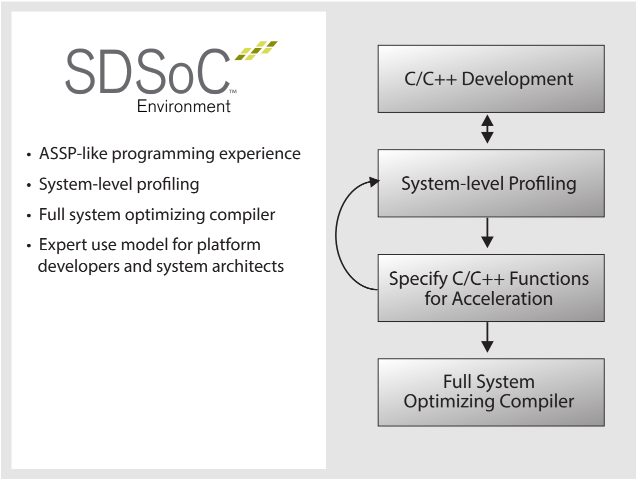


❶ Software development starts immediately, 3rd party and end user platforms

❷ SDSoC's ASSP-like development, system-level profiling and full system optimizing compiler empowers software developers to accelerate C/C++ functions

The SDSoC Development Environment provides a greatly simplified ASSP-like C/C++ programming experience including an easy to use Eclipse integrated design environment (IDE) and a comprehensive development platform for heterogeneous Zynq platform deployment. Complete with the industry's first C/C++ full-system optimizing compiler, SDSoC delivers system level profiling, automated software acceleration in programmable logic, automated system connectivity generation, and libraries to speed programming. It also provides a flow for customer and 3rd party platform developers to enable platforms to be used in the SDSoC development environment.

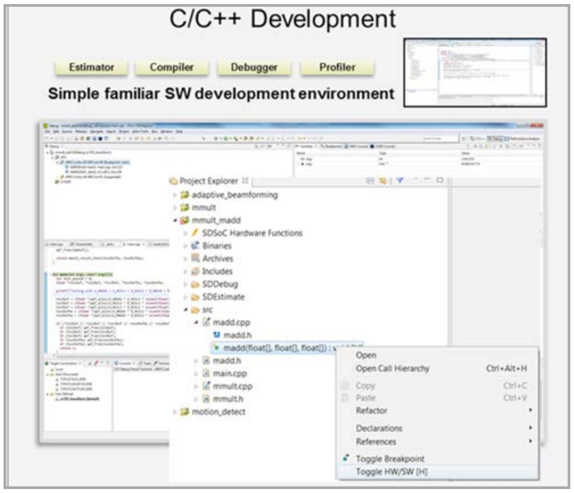
The SDSoC Development Environment


SDSoC™
Environment
C/C++ Development
System-level Profiling
Specify C/C++ Functions for Acceleration
Full System Optimizing Compiler

- ASSP-like programming experience
- System-level profiling
- Full system optimizing compiler
- Expert use model for platform developers and system architects

ASSP-Like Programming Experience

Used by systems and embedded software developers, SDSoC provides an Eclipse IDE to develop C/C++ applications running on bare metal or operating systems such as Linux and FreeRTOS. SDSoC enables the creation of complete heterogeneous multiprocessing systems, including software running on ARM/NEON processors, software accelerators in programmable logic and reuse of legacy HDL IP Blocks as C-callable libraries. Unlike traditional separate hardware-centric and software-centric flows, which can result in development delays and uncertainty in system architecture and performance, SDSoC is architected to provide rapid system profiling and software acceleration in programmable logic in a familiar embedded developer framework.



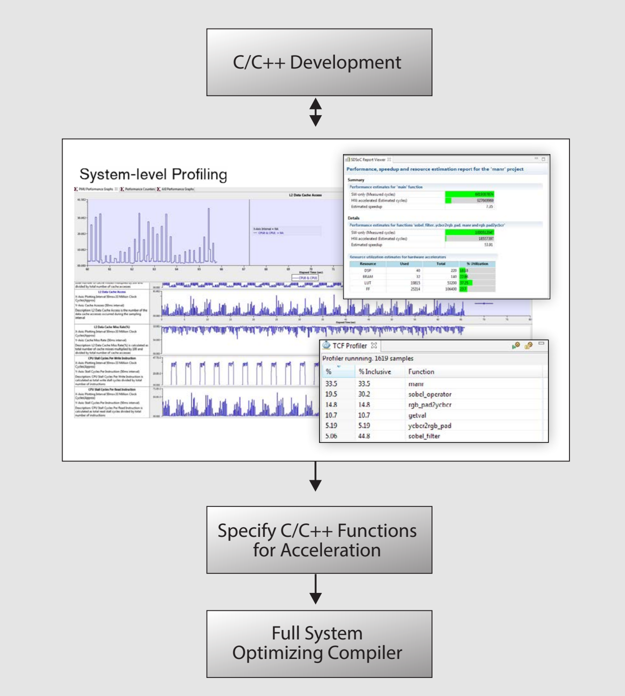
The easy to use IDE features a project creation wizard where the user can select the target platform and operating system, and use templates to create fully functioning SDSoC projects that can be used as starting point to build their own applications.

SDSoC also provides software teams with a simple GUI option to select functions to be accelerated in the programmable logic. The SDSoC compiler will generate all the necessary hardware and software pieces to be readily run on a target platform board.

System-Level Profiling

Building on the advanced software profiling found today in the Xilinx Software Development Kit (SDK), SDSoC adds system-level profiling for rapid system (hardware and software) performance estimation. This enables rapid generation and exploration for optimal total system performance and power.

System-level Profiling


C/C++ Development
System-level Profiling
Specify C/C++ Functions for Acceleration
Full System Optimizing Compiler

SDSoC Report Viewer

Performance, speedup and resource estimation report for the 'manr' project

Summary

Performance estimates for 'main' function

SW-only (Measured cycles)	6819387874
HW accelerated (Estimated cycles)	927669968
Estimated speedup	7.35

Details

Performance estimates for functions 'sobel_filter, ycbcr2rgb_pad, manr and rgb_pad2ycbcr'

SW-only (Measured cycles)	1000512047
HW accelerated (Estimated cycles)	18557397
Estimated speedup	53.91

Resource utilization estimates for hardware accelerators

Resource	Used	Total	% Utilization
DSP	40	220	18.18
BRAM	32	140	22.86
LUT	19815	53200	37.25
FF	25214	106400	23.7

TCF Profiler

Profiler runnning. 1619 samples

%	% Inclusive	Function
33.5	33.5	main
19.5	30.2	sobel_operator
14.8	14.8	rgb_pad2ycbcr
10.7	10.7	getval
5.19	5.19	ycbcr2rgb_pad
5.06	44.8	sobel_filter

SDSoC allows users to profile their software application, using a non-intrusive profiler based on PC sampling as well as using the standard gprof profiler, to identify the function that takes up most of the time and are candidates for hardware acceleration.

SDSoC then adds rapid estimation of system (hardware and software) performance and device utilization to enable fast system-level architecture exploration for optimal total performance, resource utilization and power. Users can specify which functions should be accelerated in programmable logic, and SDSoC instruments the C/C++ code to report software cycles, hardware cycles, estimates for the data transfer, overall application speedup as well as hardware resource utilization.

Leveraging the platform-based performance estimation flow, software developers can rapidly get an estimation of performance impact of moving one or more software functions to hardware for acceleration in a matter of minutes versus actual hardware generation which can take up to an hour or more.

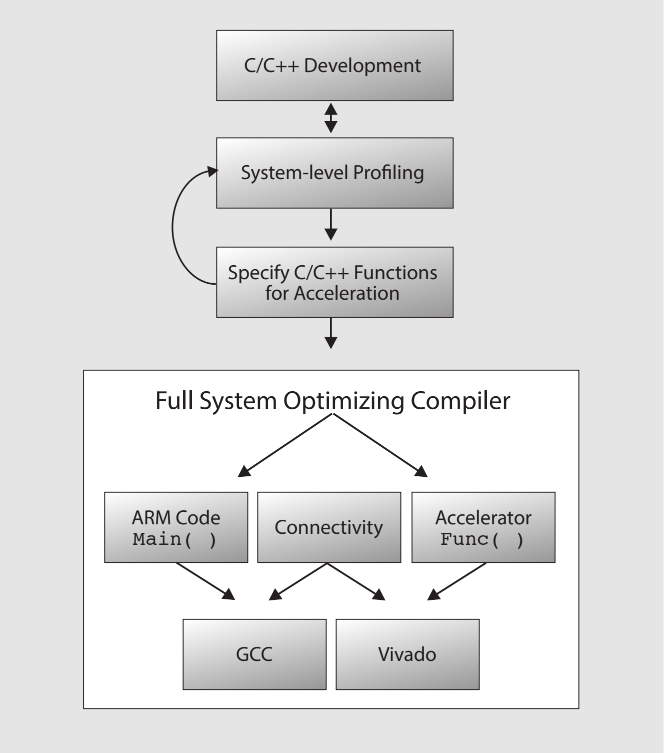
On the target platform, SDSoC provides automated performance measurements for cache, memory, software accelerator and bus utilization by using performance counters provided by ARM CPU and by automatically inserting AXI Performance Monitors (APM) into the programmable logic to collect hardware performance data. Software running on the platform collects the performance data and SDSoC uses this data to help identify performance bottlenecks in the system. This rapid performance feedback enables optimal partitioning of code to meet system level requirements for performance and power while saving weeks in the development process.

Full System Optimizing Compiler

SDSoC also features a full system optimizing compiler targeting both the ARM-based processing systems as well as the programmable logic. SDSoC is designed to enable system architects as well as software teams to use a 'golden C/C++ source' to rapidly configure macro and micro architectures with optimal system connectivity generation. This results in optimal system connectivity and memory interfaces, and enables rapid design space exploration allowing the developer to trade-off performance, throughput, and latency while maintaining short design iteration times.

The complier leverages a foundational high-level synthesis compiler technology that is utilized by more than 1,000 programmers for generation of high performance C/C++ based IP. Together the compiler and linker transforms programs into complete hardware/software systems based on target platform and user-directed automated software acceleration generated in programmable logic.

Full System Optimizing Compiler


C/C++ Development
System-level Profiling
Specify C/C++ Functions for Acceleration
Full System Optimizing Compiler
ARM Code
Main( )
Connectivity
Accelerator
Func( )
GCC
Vivado