

Software-Configurable Processors on the rise

New configurable technology gives software designers the power of hardware

By Albert Wang

IN AN IDEAL world, engineers would prefer to base designs on a flexible architecture such as implementing functionality in software. This would offer flexibility with new functionality introduced over time and would extend the lifecycle of devices already deployed in the field.

Unfortunately, today's leading edge applications demand the processing performance of dedicated hardware, and software running on general purpose or even specialized coprocessors just can't provide the performance. Configurable architectures, on the other hand, marry the flexibility of software programmability with the performance capabilities of ASIC designs.

Traditional configurable devices such as FPGAs or hybrid architectures which integrate a general purpose processor with reconfigurable logic require two design teams – one for hardware and one for software – and developers must account for interconnect issues such as passing of data, latency, buffering, deterministic response. This is complex, not least in managing the flow of information between the two design teams.

Software-configurable processors eliminate many of these issues by taking integration a step further and implementing programmable logic within the processor itself. The programmable logic is part of the processor, not merely an independent extension, and is accessed in the same way that commonly-used functions are accessed in the processor: via software instructions.

The Stretch Software-Configurable Processor (SCP) Architecture is the first off-the-shelf implementation of the Xtensa ISA (see Figure 1), a configurable and extensible processor implementation generator introduced in 1999 by Tensilica. Core to the SCP is the Instruction Set Extension Fabric (ISEF) which extends flexibility to a higher dynamic level by supporting run-time reconfiguration of instructions. The Stretch C Compiler manages the abstraction of custom instructions in application code, tracks dependencies across the ISEF and Xtensa cores, and optimizes allocation of ISEF configurable resources.

Completing the SCP architecture is a 128-bit Wide Register (WR) file that serves as a streamlined pipe for efficient passing of data between the Xtensa ISA and ISEFs.

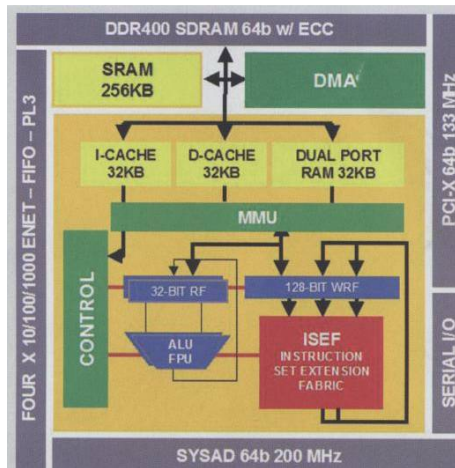


Figure 1 (above): The S5610 Processor Architecture
 Stretch's Software-Configurable Processor (SCP) Architecture is the first off-the-shelf implementation of Tensilica's Xtensa ISA. The Instruction Set Extension Fabric (ISEF) extends flexibility to a higher dynamic level by supporting run-time reconfiguration of instructions. A 128-bit Wide Register (WR) file efficiently passes data between the Xtensa ISA and ISEF.

With a software-configurable architecture, configurable logic is not architected as a coprocessor but rather as an integrated part of the execution pipeline.

Developers mark off "hot spots" within their program for the compiler to convert into Extension Instructions. Complex algorithms are then reduced to a handful of optimized custom instructions that each represent hundreds of lines of C code executed in a highly parallel pipeline. Such calculations execute in tens of cycles, down from hundreds or thousands of cycles, increasing the computational capacity of applications.

Key to the efficiency of the SCP is that Xtensa instructions and Stretch ISEF instructions share the same pipeline and the same instruction decode unit. Stretch has extended the five-stage pipeline structure of Xtensa to allow multi-cycle Extension Instructions. In this way, Extension Instructions are organic to the processor. Logically, they are the same as any other instruction the processor can execute. Thus, the implementation details of the programmable logic are encapsulated in the familiar format of software instructions. Issues such as Extension Instruction latency are automatically managed for the designer by the compiler and mechanisms within the processor. There is no differentiation between standard and Extension Instructions.

Accessing configurable functionality as software instructions has a tremendous impact on the way designers approach hardware design. Even though the Extension Instructions are implemented in hardware/programmable logic, designers create and use them in an entirely software context. This keeps the design of the system in a single development environment which is both conventional and familiar to software designers.

The Stretch tools simplify the task of writing Extension Instructions through the use of Stretch C. With only a few exceptions, Stretch C supports all standard ANSI C operators while introducing a few enhancements and limitations. The compiler optimizes the overall latency of the Extension Instruction, which directly translates to the efficiency of pipelining of instructions and how quickly Extension Instructions can be issued after each other. The Stretch C Compiler also assists developers in reducing the number of resources an instruction needs in ways that are difficult for a person to envision and implement. For example, when two or more instructions share structures, the compiler can implement these shared structures using the same resources. On the other hand, the compiler is able to distinguish the need for resources that may not be readily apparent to a person, such as those required for register forwarding, multiplexing, and context save/restore functions.

Importantly, code written in Stretch C can be compiled for targets other than an SCP processor. For example, the Stretch C compiler can produce a functionality equivalent code image for execution on an x86 processor, enabling developers to test software while hardware is still under development.

It is easy to underestimate the effectiveness of abstracting hardware as software. High performance applications typically have a processing bottleneck centered around a particularly compute-intensive algorithm; consider the complex calculations to compute the 41 motion vectors required for H.264 video compression or the sheer magnitude of image data sets for high resolution, high frame rate medical imaging applications.

To illustrate the effectiveness of the automatic parallelization of code, consider a common function like RGB to YCbCr video colorspace conversion (see Figure 2a). This function requires 9 multiplies, 8 adds, and 3 shifts per pixel. Figure 2b shows the extent of parallelism possible with several types of processors. Figure 2c shows the conversion function coded in



Figure 2c : An Extension Instruction

The color space algorithm written in Stretch C and implemented as a single Extension Instruction.

```

/* Stretch extension type and function name
/* operands are 128-bit Wide Registers
SE_FUNC void RGB2YCBCR (WR A, WR
*B) {
char Y[4], Cb[4], Cr[4];
char R[4], G[4], B[4];

/* preamble – defining Wide Register fields
(not all shown for simplicity)
R[0] = A(23,16); G[0] = A(15,8); B[0] =
A(7,0);
/* and so on. . . */
/* Calculation (automatic parallelization)
for (i = 0; i < 4; i++) {
Y[i] = (77*R[i] + 150*G[i] + 29*B[i]) >> 8;
Cb[i] = (32768 - 43*R[i] - 85*G[i] + (B[i]
<< 7)) >> 9;
Cr[i] = (32768 + (R[i] << 7) - 107*G[i] -
21*B[i]) >> 9;}
/* pack output
*B =
(Y[3],Cr[3]+Cr[2],Y[2],Cb[3]+Cb[2],Y[1],Cr[
1]+Cr[0],Y[0],Cb[1]+Cb[0]);}

```

Figure 2d: Using Extension Instructions

This loop converts four pixels of an image at a time. The entire conversion takes three instructions, including loading the pixels and storing the results.

Program Loop:

```

for (...) {
WRGET0(&A, 12);
/* Load 12 bytes (4 RGB pixels)
*/RGB2YCBCR(A, &B);
/* Convert 4 pixels
*/WRPUT0(B, 8);
/* Store 8 bytes (4 YCbCr pixels */}

```

Stretch C. This function becomes a single Extension Instruction which the compiler will automatically implement in the ISEF without manual effort from the developer. Figure 2d shows how the new conversion instruction is used in a program just like any other instruction. Stretch's 300MHz S5000 processors can convert 1447 8-bit-per-color, D1 images/second, compared to an estimated 300 D1 images/second by an industry standard 600MHz DSP.

Unquestionably, field programmability, both in terms of software and hardware, is required to enable engineers to meet the demands of these applications. Software-configurable processors yield a balance of flexibility and performance where SoC and ASIC design are economically unfeasible and general purpose software processors simply can't fulfill performance needs. Developers can continue to try to extend existing architectures, but in the end, they will be left behind as software-configurable processors redefine the design process for high-performance silicon.

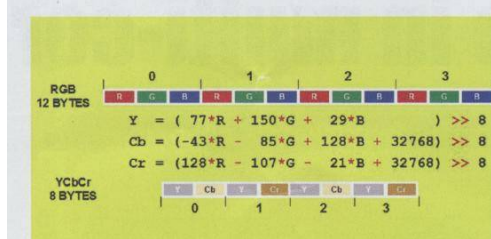


Figure 2a (above):

RGB to YCbCr video color space conversion requires 9 multiplies, 8 adds, and 3 shifts per pixel.

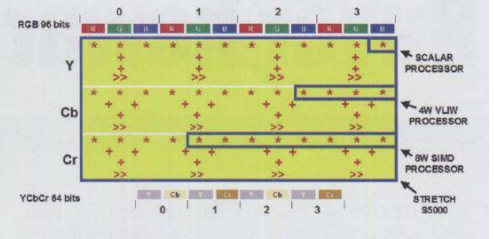


Figure 2b (above):

Different processor architectures are able to execute the color space algorithm in a different number of instructions.

Albert Wang is chief technology officer and co-founder of Stretch Inc.