

CW5631 Vector Processor for Multimedia Applications

John Redford
ChipWrights, Inc.

ABSTRACT

The ChipWrights CW5631 System on Chip (SoC) contains a DSP with a novel vector architecture. It exploits the parallelism and narrow data typical of image processing to gain high performance at a low cost and power. It contains sixteen 32-bit datapaths all working off of a single instruction. It can do thirty-two 16-bit MACs/cycle, and eight 32-bit memory accesses per cycle to 128KB of on-chip memory. A serial datapath handles low-performance code and OS functions. The SoC includes an ARM9 core for system code, a serial processor for Huffman accelerations, and several I/O blocks, including DDR2, video input and output, USB, Ethernet, and I2C, interconnected in a unique non-blocking fabric. The DSP is built in 130nm, runs up to 360MHz, and draws about 500mW.

1. INTRODUCTION

Consumer image processing applications need high computing performance at a low cost and power. Until recently, the only means of doing this was to use fixed-function hardware. This chip provides similar performance, cost, and power to a fixed-function chip in a fully programmable way, permitting new classes of applications, shorter time-to-market, and easy product differentiation.

High performance is achieved by exploiting specific features of imaging applications: easily found parallelism, narrow data, and regular memory-access patterns. Low cost is achieved by reduced overhead in the instruction fetch and dispatch, from integration of peripherals, and from using low-cost processes, design styles (i.e., no full custom) and packages. Low power is the result of low overhead on instruction handling and extensive clock control.

The architecture focuses on data handling rather than on instruction handling by using a single-instruction-multiple-data (SIMD) architecture instead of the super-scalar or VLIW approach. This paper describes how problems traditionally associated with programming SIMD machines are overcome by utilizing novel architectural features.

2. DSP ARCHITECTURE

The DSP is a variant of a vector architecture. It has 16 parallel units, a central serial datapath, and an 8-bank on-chip memory interleaved on a 32-bit basis. A single 32-bit instruction is fetched from the I-Cache in each cycle, checked for hazards, and distributed to the serial and parallel datapaths.

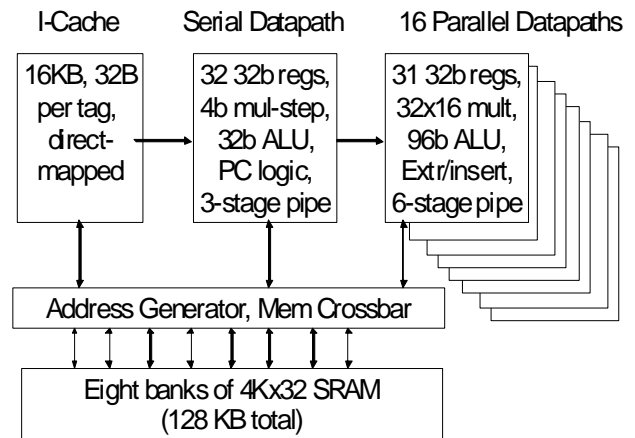


Figure 2.1 DSP and SRAM Block Diagram

From least to most interesting, the blocks are:

I-Cache: Can hold 16KB (4K instructions) of code. It uses a 64B line size, is direct-mapped, and can be filled from any memory in the chip's address space: on-chip SRAM, off-chip DRAM, or off-chip boot memory on the External Bus Interface. Individual lines can be locked down for determinism.

Serial Datapath: Typical 32-bit RISC with 32 registers and three operand instructions; the ISA is unique to ChipWrights. It is an easy porting target for low-performance application code and the operating system. It contains the PC, interrupt logic, PC breakpoint logic, and cycle counter for performance monitoring. It supplies data types and address bases to the parallel datapaths, and handles loop counts. It has a 4-bit/cycle multiply-step operation, and special instructions for unaligned data.

Primary Memory: Contains 128KB of SRAM organized as eight 4K x 32-bit banks, each with byte write enables. Addresses are interleaved so that up to eight simultaneous accesses are possible.

Address Generator and Crossbar: The generator combines addresses, strides, and offsets from the serial datapath, with offsets from the parallel datapaths to calculate the ultimate memory addresses. The crossbar permits any datapath or the I-Cache to get at any bank or to get out onto the system-wide memory access crossbar (MSS). There are no private memories on the chip – all are accessible. The MSS connection is also used for DMAs that move data between the primary memory and DRAM.

Parallel Datapaths: These 32-bit processing units each have their own register file, multiply-accumulators, and paths to memory. They all get the same instruction from the serial datapath, but can be selectively enabled and can generate their own addresses.

The parallel datapaths implement vector operations. Like traditional vector architecture, they perform the same operation on different registers and do strided or scatter-gather accesses to memory. A strided memory access starts with one address for datapath 0, and adds a constant offset for each subsequent datapath. For instance, a read with a stride of 100 starting at address 1000 reads 1000 into datapath 0, 1100 into datapath 1, 1200 into 2 and so on. This is useful for dividing arrays among the datapaths. A scatter-gather memory access again starts with a base address, but gets an offset from each datapath. These can also be called table accesses because they permit each datapath to access a different entry in a table. For instance, if datapaths 0, 1, 2, and 3 had offsets of 10, 200, 2, and 5, a gather read was done at base address 1000, and the data is read from addresses 1010, 1200, 1002, and 1005.

Also, like in traditional vector architecture, all operations are conditioned by an enable bit for each datapath. If the enable is off for a particular datapath, it does not modify registers or memory. The enable can be set by comparison instructions. This permits if-then-else operations to be done by turning on some datapaths for the then-clause, and a different set for the else-clause. However, because there is only one instruction stream, both the then-clause and the else-clause must be executed if any datapaths are enabled in them.

The architecture has been extended substantially beyond what older machines do. For example, unlike traditional vector architecture, each datapath has its own register file and can be envisioned as operating by itself. Programmers (and compilers) do not have to think in parallel to use the machine. Instead, they can think about one datapath working on one part of an array, and the hardware takes care of spreading the other datapaths around the other segments of the array.

This hiding of parallelism happens in the following ways:

The number of active datapaths is controlled by a register and can vary at run-time, and among implementations.

There are looping instructions that know the number of active datapaths. If more are active, fewer loop passes are done. If a run-time check finds that no parallelism is possible, the number of active datapaths can be set to one.

Loop instructions know when the end of an array is reached, and can disable datapaths that are beyond it. Arrays do not have to be a multiple of the number of datapaths long.

Branch instructions can control the enable bits. They save the old state of the enables, set the new state, and branch if all the datapaths are inactive. Some datapaths must execute the then-clause of a branch and some the else-clause. The enables are set differently for each. The ability to easily save the state of the enables makes it possible to nest if-then-else's arbitrarily deeply. In a standard vector machine, extra work is needed for the save and restore of enables.

Some scatter-gather instructions let each datapath have its own private data structures, free from interference by the others. In a typical scatter operation, there is no assurance that two vector elements won't be written to the same address. In this machine, the scatter op can insert the datapath index into the LSBs of the address to ensure a unique address for each datapath. These also allow a programmer to have deterministic access time to tables by using multiple copies of them.

In addition to parallel operation, the DSP has several instruction set features that accelerate image processing:

Sub-word Parallelism: Registers can be treated as four 8-bit items or two 16-bit. Instead of doing a single 32-bit multiply- or ALU operation, one can do four 8-bit or two 16-bit operations. This is similar to the Intel MMX™ and PowerPC AltiVec™ instructions. A problem with those architectures is that the results may quickly overflow the 8-bit or 16-bit slots, forcing the programmer to convert to a wider form (8->16, or 16->32). This DSP handles that by having a wide accumulator. When doing four 8-bit ops, the accumulator is treated as four 24-bit registers (96 bits total). An 8x8 multiply has a 16-bit result, so many multiplies are accumulated before the 24-bit register overflows. Likewise, doing two 16-bit ops causes the accumulator to be treated as two 40-bit registers. With 16 datapaths, and the ability to do two 16-bit MACs/datapath, the DSP can do thirty-two 16-bit MACs/cycle. The DSP supports applying a single value to a packed register (multiplying all fields by one constant), and supports dot product operations (multiply each field and sum) and sum-of-absolute-differences for motion estimation.

Built-in extract and insert: Instructions can combine the extraction or insertion of bit-fields with other operations, so that handling narrow fields adds no overhead. E.g. one instruction can extract a byte or word from a 32-bit register, do a multiply-accumulate, shift right to remove fractional bits, do a saturation check on the result, and do an insert into a byte or word of a 32-bit output register. The extraction and saturation check can be signed or unsigned. The extract or insert position can be incremented or decremented as part of the instruction for sweeping through registers. The right shift and saturation check can also be done on packed operations, which saves cycles.

Many-operand instructions (VDIW): A typical RISC instruction has two input operands, either two registers or a register and a literal, and one output result. This DSP has instructions that can have up to seven input operands and three outputs. The seven inputs are: an A operand register, the A data type (8b, 16b, 32b, packed 8b, packed 16b; and signed/unsigned), a B operand register and the B data type, a 9-bit literal, the accumulator, and the output data type. The three outputs are a parallel output register, and serial input and output pointer registers. On the memory side, one can have up to four input operands (a base address, an address offset or post-increment, a stride, and a parallel write data register), and two outputs (an updated address pointer and a memory destination). With complex instructions like these, many operations can be specified by a single 32-bit instruction. This can be called a Very Dense Instruction Word (VDIW™), and permits high performance with a single-issue instruction dispatcher, a small instruction cache, and a small amount of hazard checking logic. This saves area and power compared to trying to dispatch many instructions in one cycle.