

# TriCore 32-bit MCU/DSP

## How to win both ways

*One of the targets of the Infineon TriCore 32-bit unified processor was to reach full-blown DSP performance while keeping the software friendliness of a 32-bit CPU. To reach that goal, all the DSP features were carefully integrated in the architecture while keeping a "software friendly" programming environment. This led to the TriCore architecture.*

*As shown in a series of DSP benchmarks, the TriCore results are similar to a Dual-Mac DSP, with as a big advantage that the TriCore architecture is easier to program and behaves much better on common CPU algorithms.*

### BACKGROUND

System designers have to deal with complex systems-chips, which include programmable cores of different varieties (i.e. 32-bit CPU, MCU, DSP).

The Infineon TriCore solution proposes a single unified processor which can efficiently handle Digital Signal Processing tasks, as well as multiple interrupts, while supporting HLL (high level language).

From a software developer viewpoint, TriCore offers a single unified 32-bit RISC CPU software model.

- A data register file (16 registers of 32 bits);
- An address register file (16 registers of 32 bits);
- Separate the load/store instructions from the arithmetic instructions;
- Full compiler support and full integrated RTOS;
- 4 Gbyte memory space;
- 32-bit orthogonal instruction set (with 16-bit subset);
- 3 basic addressing modes.

### THE DSP VIEWPOINT

From a DSP viewpoint then the obvious question is: how can TriCore reach full-blown DSP performance into a RISC-like software model.

#### INSTRUCTION SET ARCHITECTURE

TriCore is defined in an hierarchical manner: separating core from subsystems (memory and peripheral) and separating ISA (instruction set architecture) from implementation.

It is firmly based in the RISC school of ISA and the most relevant features with respect to DSP are the followings:

- Simple register file model (no dedicated registers);
- No mode bits;
- Very few status bits.

When the TriCore project was started 3 years ago these features were controversial since they consume opcode space and width. However all modern DSP architectures now follow identical principles. This is due to several reasons: the main one being compiler-friendly but also hardware simplification and future superscalar implementation plays a big role.

### MICRO-ARCHITECTURE

When DSPs were introduced at the beginning of the 80's, they differed from CISC CPUs by their use of the following:

- Harvard memories;
- High-speed clock and single cycle instructions;
- Pipeline execution flow;
- Parallel execution: memory, MAC, ALU are concurrently activated.

Those techniques are largely used today by all RISC CPUs. No wonder that TriCore makes heavy use of them:

- **Harvard memories** – The TriCore internal memory space is independently divided between code and data.
- **Single cycle instructions** – All major instructions takes 1 cycle. The exceptions are division, 32x32 bit Multiplication (2 cycles), mixed address/data registers, and branch cases. Also very important is the fact that (except for the MAC) there is no added latency.
- **Pipeline** – Identically to DSPs, TriCore makes the most efficient use of the time slots offered by a pipeline architecture (i.e. very few cases of stalling). But, contrary to DSPs, TriCore uses a shallow pipeline (4 stages). It simplifies hardware and gives good performances on all branches.
- **Parallel execution of units** – For that we used superscalar techniques (multiple issuing of instructions). In its first implementation, TriCore can issue 3 instructions per cycle in a loop: 1 Integer Processing (or IP), 1 Load/Store (or LS), 1 loop corresponding to the famous single cycle MAC loop of DSPs.

It must be noted that micro-architecture choices are implementation choices and can vary from 1 chip to another without modifying the software visible architecture.

#### DATA MEMORY AND BUS

A typical DSP algorithm spends around 60% of time on dual memory operand operations. Hence to achieve high performance:

## PROCESSOR

- Dual data memories ( called X and Y ) is the preferred scheme;
  - Multiple data buses (generally 16-bit) is also a standard.
- TriCore offers original solution, since it uses a single data bus/single data memory model. The main advantages are the followings:
- Unity in hardware (a unified, single-ported bus attached data memory);
  - Simplification of instructions;
  - More compiler friendly.

So the question is: how to reach the DSP basic requirement of dual operand operations?

When looking carefully at algorithms one can see that multiple 16-bit data buses are not required as such. What is required is to keep a throughput for a given operation. TriCore single 64-bit bus can provide the same bandwidth as up to 4 16-bit buses.

Also a dual memory access per cycle (X and Y) can be achieved by performing a single access of two 16-bit data items ( X memory access) followed by another

# PROCESSOR

single access of two 16-bit coefficients (Y memory address). The cost is the unrolling on loop algorithms. But, throughput is identical.

## DATA PROCESSING UNIT (DPU)

Since the majority of DSP instructions are executed in the processing of data, it is the most important unit. Typically, it will have:

- A sophisticated MAC structure;
- A combined ALU + Barrel shifter;
- Support for "real signal" data types (left alignment, rounding, saturation);
- Several dedicated accumulators (with size larger than the native data size).

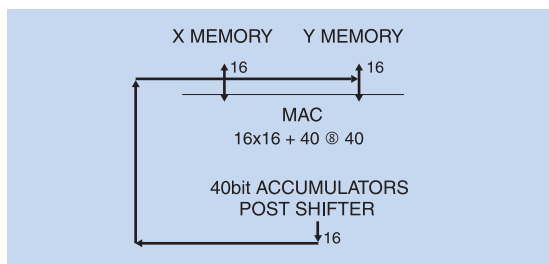


Figure 1. Typical DSP Memory/Bus/DPU

TriCore has a flexible and powerful datapath.

The dual MAC unit is made of 2 parallel 16x16 → 32 MACs which can be recombined into a single 64 + (16x16 + 16x16) → 64. It supports packed, complex, 16x32 and 32x32 operations.

The ALU is implemented as a 32-bit split datapath so that packed operations (2 x16-bit or 4 x 8-bit) are done in 1 cycle. Also a 32 bit BMU (bit manipulation unit) and a 64bit funnel shifter completes the unit.

From the beginning the TriCore ISA supported DSP data types (Q-format):

- All arithmetic instructions are normal or saturated types;
- Left alignment (to remove the redundant bit in signed multiplication);
- Rounded so that a 16x16 multiplication result can give a 16-bit rounded value;
- Detection of -1 \* -1 case makes necessary to support standard voice coding algorithm.

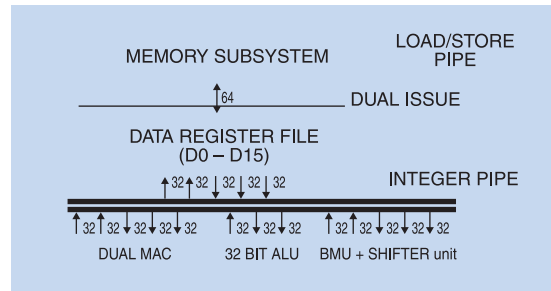


Figure 2. TriCore Memory/Bus/DPU

For compiler reasons no dedicated accumulators were created. Extended precision is dealt with by pairing 32-bit registers to provide 64-bit accumulations.

## ADDRESS GENERATION UNIT (AGU)

The classical DSP model will have several AGUs since it must access multiple memories. Each AGU is an ALU with a large number of specialized registers in order to:

- Support complex data structures;
- Realize special addressing modes (circular buffer, bit reverse).

TriCore departs from the DSP model by simplifying everything. It uses a regular register file (A0-A15) with a very simple instruction set (load, store, add):

- There is no need of complex multiple AGUs since TriCore works with a single memory model;
- The specific DSP addressing modes (circular, bit reverse) are implemented using the regular address set. They are just 2 special cases of inc/decrement addressing modes.

### Example

```
ld.w d0, [a0+]4 ; normal
ld.w d0, [a0/a1+c]4 ; circular
; a0 contains base
; a1 contains index and boundaries.
```

With 3 basic addressing modes (immediate, offset, post/pre increment and decrement) we found out we could synthesize most complex data structures without loss of performances. This is due to the superscalar structure where the address generation unit (LS unit) works in parallel with the MAC/ALU.

```
lmsloop: ;-----
maddm.h acc,acc,d3,d4 ul,#1 ; acc += x0*k0 + x1*k1
st.d [Kptr]-0x10,d6/d7 ;||st k'0, k'1, k'2, k'3,(next loop)
maddrs.h d6,d4,d3,d8 ul,#1 ;k'0 = k0+ x0*err, k'1 = k1 + x1*err
ld.d d2/d3,[Xptr+]8 ;||ld x2,x3,x4,x5
maddm.h acc,acc,d2,d5 ul,#1 ; acc += x2*k3 + x3*k3
maddrs.h d7,d5,d2,d8 ul,#1 ; k'2 = k2 + x2*err, k'3 = k3 + x3*err
ld.d d4/d5,[Kptr+]8 ;||ld k4,k5,k6,k7
loop LC,lmsloop ;-----
```

Figure 3.

## PROGRAM CONTROL UNIT (PCU)

DSPs have never invested much time and area of instructions control.

The 2 reasons are:

- Control flow is generally looped and independent of data;
- Deterministic behavior is a must and it has been considered incompatible with branch prediction techniques.

Instead of branch prediction techniques several "tricks" can be found in DSPs.

- The most common one is *zero overhead loop*. A specialized loop counter (and sometimes 2 or 4) controls the behavior of the loop; hence no instructions are needed to go back to the beginning.
- Another technique is *conditional execution* of instructions (also known as predicated in the CPU world).

On TriCore the zero overhead loop is realized with the help of a shadow buffer mechanism. No specific initialization instructions are needed and no special registers are user visible.

While not being a "predicated machine" TriCore has conditional version of several instructions (add, sub, sel, move).

## I/O AND SYSTEM

Two typical features of DSP I/O are as follows:

- The capacity to have deterministic (predictable) behavior to real time events;
- DMA capability which often links serial ports directly to memory

Deterministic behavior is not exclusive to the DSP world. In fact, it is a common feature of embedded world and more severe requirement for many micro-controllers applications. Since TriCore is a MCU-DSP it features a split fast context switching (16 registers are saved in 2-4 cycles), multi-level interrupt system (256 levels) and a multi-level memory hierarchy (scratchpad, cache, bulk memory). All features which enhances a modern CPU capability to answer the real-time challenge.

As such DMA is not strictly a feature of a core. However it is part of the TriCore system architecture. A specialized modular unit called PCP/DMA takes care of all high-speed I/O to memory data transfer.

## DSP INSTRUCTIONS

DSP instructions are characterized by:

- Compound instructions which control several units in parallel;
- An extended arithmetic instruction set (rounding, saturation, etc.) which typically applies to MAC.

TriCore replaced complicated DSP compound instructions by dual issuing of instructions (and even triple issuing for loop).

The TriCore extended arithmetic instruction set offers a large palette of possibilities:

- Single MUL, MADD(mul-add), MSUB (mul-sub) on 16-bit signed data;

- dual MUL, MADD(mul-add), MSUB (mul-sub) on 16-bit signed data;
- Same operations as above giving an accumulated results ( 48-bit);
- Single MUL,MADD,MSUB on 32-bit signed and unsigned data;
- Most ALU operations such as add, sub, shift, abs, cls (count leading sign) can be applied to single (32-bit) word, dual (16-bit) half-word or quad bytes.

As an illustration of the power of these instructions (see figure 3) we will show the implementation of a FIR filter with update of coefficients (LMS). In this algorithm 2 equations are necessary:

$$(1) \text{acc} += x[i]*k[i]$$

$$(2) k'[i] = k[i] + x[i]*\text{err} \text{ where } k'[i] \text{ is written back in place of } k[i]$$

This routine takes 4 cycles per loop iteration and 4 points are computed per loop. Hence TriCore can do an adaptive LMS filter (accumulation and update ) at a rate of 1 point/cycle.

## APPLICATION ORIENTED INSTRUCTIONS

Application oriented instructions such as viterbi decoder are now part of any DSP instruction set.

TriCore has a more generalized solution: this is a concept of a *Register-file coupled coprocessor with user-defined instructions*. For instance a viterbi decoder or a bit processing engine is implemented like another processing unit and can become part of the standard instruction set.

## CODE SIZE

In order to reduce code size, the most common DSPs used a very compact opcode (reduced width = 16-bit).

For TriCore the problem is slightly different since it can take the role of a host CPU. In this role 1-10 Mbytes of code is not uncommon (compared to the 16k-32Kbytes of DSP) with the overwhelming majority of the code written in C. Hence the user wants a very compact "C" code size more than a reduced "DSP loop" code size. This is achieved by having the most common instructions available in 16-bit form. To achieve this we use implicit registers, shorter displacements and shorter op-codes. ■

---

*Daniel Francis Martin is a senior architect at Infineon Technologies (San Jose, California). He is responsible for the definition of the DSP features, performance analysis and benchmarking of the TriCore 32-bit architecture. Prior to Infineon he was involved in all the aspects of DSP design with companies such as STC, Multitone, SGS-Thomson, ASCOM-HPF and SECOM.*

*Daniel Francis Martin was born in 1952 in Avignon (France) and has a degree in mathematics and physics from the University of Marseille (France) and a degree in Engineering and Telecommunications from the Polytechnic of North London (England).*