# Software Programmable DSP Platform Analysis, MM 3

**Time and Place:** Friday, April 15 at 12.30, room A5-006 on Niels Jernes Vej 12.

**Topic:** Understanding basic architectural features and limitations, and matching algoritms to the architecture.

**Literature:** *Local Microcode Compaction Techniques*, D. Landskov, S. Davidson, B. Shriver. Don't study the paper in-depth, but try to understand the idea of micro-operations. The "microoperation tuple" notation is particularly good to keep in mind, although this formal technique reaches beyond the scope of today's exercises, which will give you a taste of how to use the technique informally.

**Exercise 1:** A new company, "DSPs 'R' Us," decides to manufacture a new DSP that is intended to beat the low-end DSP competition. The engineers know that single-cycle MAC (i.e., multiplying two coefficients and adding the product to the contents of a register, storing the result in that same register) is almost mandatory, and in order to boast superior performance they decide to have two MACs.

Their initial DSP design has an architecture with the following properties:

- Harvard architecture allowing one data move (either read or write) simultaneously with each instruction fetch.
- Addressing modes: immediate, register-direct, register-indirect, and register-indirect with post-increment. There are four address registers, named A0, A1, A2, and A3.
- A load or store may be issued simultaneously with a data path operation.
- Zero-overhead block repeat, where an instruction indicates how many instructions are to be repeated, and how many times.
- Two accumulators and four registers named A and B, and R0, R1, R2, and R3. The accumulators can both be used as inputs for arithmetic operations, and either accumulator can receive the result.
- A data path with two multipliers with single-cycle MAC capability. Registers R0 and R1 must be used as operands for the first multiplier, which stores its result in accumulator A. Registers R2 and R3 must be used as operands for the second multiplier, which stores its result in accumulator B.
- There are no visible pipeline effects.

Assume that the instructions look something like this:

```
MOVE #1000h,A0
```

which stores the number 1000h in address register A0.

Or:

```
MAC R0,R1, A    (A0)+,R0
```

which multiplies the contents of R0 and R1 and adds the product to the contents of accumulator A, storing the result back into accumulator A. Next, the contents of the memory location with the address contained in A0 is moved to R0, and A0 is incremented by one.

Or:

```
REPEAT #4,#100
```

which repeats the next four instructions one hundred times.

You get the idea. All instructions take a single instruction cycle to execute, and all results are ready for use in the next instruction cycle.

The engineers know that at the heart of most DSP algorithms lies a vector dot product. For benchmarking purposes, they want to perform a dot product of two 256 element vectors, stored at addresses 1000h and onward, and 1100h and onward, respectively.

**a.** Write the equation for a vector dot product, and identify all of the micro-operations required in order to perform a vector dot product.

**b.** For each type of micro-operation required by the algorithm, write down how many of these micro-operations the DSP architecture supports in each instruction cycle. It is okay to leave the result in an accumulator.

**c.** What's the theoretical minimum execution time (measured in instruction cycles) in order to perform a dot product on the processor? Which executional units are underutilized?

**Exercise 2:** The engineers soon realize that two multipliers are overkill. They reduce the processor architecture by eliminating one multiplier and its associated accumulator and registers, leaving just registers A, R0, and R1 in the data path.

**a.** Repeat exercises 1b and 1c with the reduced DSP architecture. Where are the bottlenecks, if any?

**Exercise 3:** The engineers now realize something serious must be done. They expand the Harvard architecture so that the processor now sports two data bus sets, enabling either two data reads, one data read combined with one data write, or one data write per instruction cycle. However, two simultaneous memory accesses can only be performed if data is located in separate blocks of internal memory. Also, memory accesses can only be performed in parallel with another operation if the operation is a MAC operation.

**a.** Is the DSP architecture now better balanced in terms of bottlenecks?

**b.** In any event, write pseudo-instructions (like the ones provided earlier) and optimize the code for best possible performance. Please use the repeat instruction for the major part of the dot product so you won't be accused of being memory hogs. What is the theoretical minimum execution time (measured in instruction cycles), and what is the practical result?

**c.** Discuss how the processor architecture could be modified to speed up the execution time.

**Exercise 4:** Just as everyone is happy, new requirements arise. Both the multiplier and the block repeat functionality take up too much silicon area and consume too much energy, and simulations indicate that the memory interface is causing timing problems at higher clock speeds. The engineers decide to modify the architecture as follows:

- The block repeat support is reduced to a single-instruction repetition.
- The multiplier is changed to a multi-stage multiplier with single-cycle throughput and two-cycle latency. This means that the multiplier operates at full speed in a series of MAC operations, but the dot product cannot be used as an operand anywhere else until at least one instruction cycle has passed after the last MAC operation.
- The pipeline and the memory interface is modified so that memory reads and writes have two-cycle latency. The pipeline is not interlocked for memory reads. Any instruction executed immediately after a memory read will see the "old" values.

**a.** How well do the algorithmic requirements match the DSP architecture's features?

**b.** Reoptimize the code. What is the practical result now?