III. Strategy for an MCAP radix-2 FFT Processor

A Decimation-In-Time (DIT) formulation of the FFT algorithm was chosen to illustrate the type of implementation that is possible using the MCAP family. Our choice for the MCAP architecture is to provide the necessary processors for the computation of one *butterfly* [3] using as much pipelining and parallel processing as possible. The computational tree shows that the 6 real additions and 4 real multiplications are required for a DIT butterfly, see Fig. 2. We assume that adders and multipliers are 4-stage processors that allow pipelining, as illustrated in the architecture shown in Fig. 1.

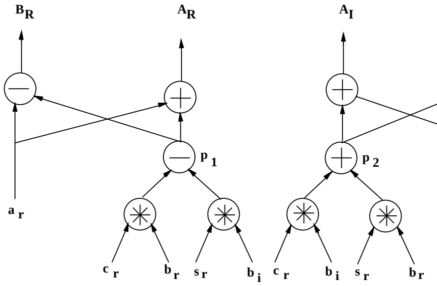

Figure 2: Computation expansion tree for DIT FFT butterfly: inputs are $a_r + ja_i$, $b_r + jb_i$, twiddle factor $c_r + js_r$, and the two outputs are $A_R + jA_I$, $B_R + jB_I$

In this paper we compare two similar FFT architectures that we have devised. The main difference between the two is the strategy used for twiddle factor [3] accessing. For architecture #1 (not shown) we assume that the twiddle factors are supplied by the attached processor's host. For the second architecture, shown in figure 1, we store the twiddle factors in bit-reversed order in a ROM.

The details of the algorithm execution on the MCAP architectures are available in the form of a detailed program in an assembly-type of language, which has been developed as part of the MCAP project. In our version of the DIT FFT, we assume that the data are originally provided by the host in normal, sequential order. The results are therefore produced in bit-reversed order, and we need to unscramble them before returning them to the host. Since we use *interleaved* memory, generating bit-reversed addresses leads to severe memory conflicts that may slow down the unscrambling dramatically. This is recognized in the literature as the *stride* problem [3], which we have solved in our FFT MCAP by utilizing the special patterning capability available in the programming of the D component.

IV. Results

Simulations were run for FFT sizes 32, 64, 128, 256, 512 and 1024. In all cases the data were taken from main memory and the FFT results were returned to main memory. Estimated results for architectures #1 and #2 are given in the Table 1.

Table 1: Algorithm execution time in microseconds

FFT size	32	64	128	256	512
Arch. #1	35.2	45.6	80.5	169.7	325.1
Arch. # 2	32.2	42.4	66.2	134.2	287.4

In general, the time T needed to execute an FFT algorithm is $T = S + Ft$, where S is the overall setup time, F is the number of FLOPs (additions and multiplications) and t is the average time required for each FLOP, including all memory accessing and communication times. Table 2 gives values of S and t for programs 1 and 2, and compares the sustainable MFLOP rates with the peak MFLOP rates of each of the FFT MCAPs proposed here.

Table 2: Setup times in ns, times per FLOP in ns, sustainable rates in MFLOPs, and percentages of peak rate

	S	t	Sust.	% of peak
Arch. #1	6062	3.3	303	71
Arch. #2	4694	2.9	345	81

The peak rate is the sum of the maximum rates of the adders and multipliers, and the sustainable rates are the values of 1/t. We estimate that if 1 micron CMOS technology is chosen to implement the architecture, a peak rate of 426 MFLOPs is predicted for both architectures.

References

[1] G. A. Gibson, V. P. Singh, S. J. Singh, Y. C. Liu, Y. C. Chang and S. D. Cabrera, *MCM Implementation of Modularly Configurable Attached Processors*, *IEEE International Computer Symposium*, Taiwan, Dec.,1994, pp 465-472.

[2] S. J. Singh, B. W. Gremel, V. P. Singh and G. A. Gibson, *Design Considerations for implementing a Modularly Configurable Attached Processor in a Multichip Module*, *MCMC'95* Santa Cruz, CA, January, 1995, pp 62-68.

[3] Charles Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Phil., 1992.

A Modularly Configured Attached Processor for Fast Fourier Transform

Olga Kosheleva, Sreedhar Cherukuri, Sergio D. Cabrera, and Glenn A. Gibson *

I. INTRODUCTION AND MCAP ARCHITECTURES

A Modularly Configured Attached Processors (MCAP) is a new special-purpose processor architecture family being developed at the University of Texas at El Paso [1], [2]. The MCAP family of architectures has been conceived to attain quickness and high utilization rates by:

- Closely matching the architecture to the set of algorithms they are to execute.
- Overlapping processing and memory accesses using memory prefetching.
- Minimizing the movement of data.
- Using a high-speed technology in a MCM or wafer scale implementation

An MCAP is constructed from two types of asynchronous connections and 10 types of components. In addition, there is a standard set of rules that allow the components to be easily configured in different ways, to construct attached processors that can efficiently perform different sets of algorithms. One of the available component types is the Instruction (I) component. The others E, T, C, J, F, L, R, S, D, belong to three different categories:

- Processor: Elementary (E)–one input, one output; Two-input (T)–one output; and Comparator (C);
- Router: Join (J)–multiple inputs; Fork (F)–multiple outputs; Link (L)–multiple inputs and multiple outputs;
- Memory subsystem: RAM (R)–no partitions; Single-access (S)–with partitions; Dual-access (D)–two inputs, two outputs with partitions.

An MCAP operates by drawing an instruction stream from the instruction memory in the I component. The instruction and data streams are separate, thereby allowing the instructions needed for the next algorithm to be distributed while the current algorithm is executing, an important feature of MCAPs. The memory subsystem components perform all memory accesses, including the accesses to the Main Memory modules that are also available to the MCAP's host. The access patterns are programmed into the controllers for the memory subsystem components through the instruction stream. A similar programming is used for the router and processor components.

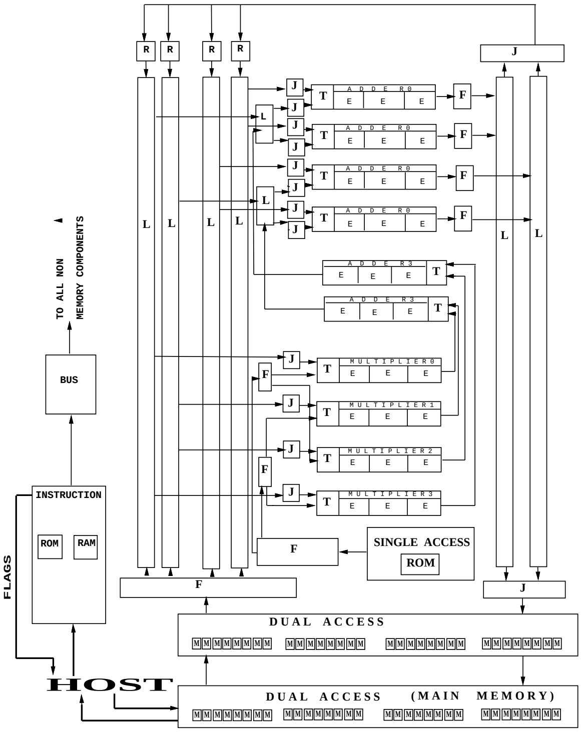

Figure 1: MCAP architecture 2 for Fast Fourier Transform

II. MCAP FOR PERFORMING FAST FOURIER TRANSFORM (FFT) COMPUTATIONS

An MCAP architecture for executing the FFT has been designed and is shown in Figure 1. It can be simulated, evaluated and analyzed using the SIMARC software package specifically developed for MCAP architectures.

---

* The authors are with the Department of Electrical and Computer Eng.,The University of Texas at El Paso. This work was supported by ONR Grant No. N00014-93-1-1343.