To the Editor:


An author replies

To the Editor:

I fully accept the general principles presented by the committee; this is indeed what I teach in my classes. Unfortunately, some of my statements have been misunderstood, misrepresented, and distorted.

In order to appreciate the points to be discussed, one should stress that the book of the Kartashevs, reviewed in Tabak, \(^1\) presents a new, reconfigurable system, whose design differs from that established by many computer manufacturers—and so does the line of thought of the authors.

The basic and widely accepted definition of the concept of computer architecture, such as Myers, \(^2\) is well known. It is not the only one. If the editor would allow, I could present a compendium of about 15 from the literature. I shall bring out just one example from Hayes, \(^3\) who refers to computer architecture on p. xi as “the study of the structure, behaviour and design of computers.” By the IBM and Myers definition microprogramming is indeed not a part of computer architecture. If we accept Hayes’s statement, then anything can be included.

Mathematics is obviously a much more precise discipline than computer architecture, and even there we have Euclidean and non-Euclidean geometry; it all depends on the first axiom that we agree to accept. Why can’t we accept a non-Myers definition of computer architecture? If we accept the general and “loose” definition of Hayes, then it makes sense to differentiate between von Neumann and microprogrammed architecture (if we decide to include microprogramming in architecture). Microprogramming had been proposed by Wilkes in 1951, five years after the von Neumann proposal.

It so happens that I personally accept the Myers definition, but apparently the authors of the book do not, and I wanted to present the reader with a different point of view. The review was long enough and there was no space for a treatise on computer architecture definitions.

The quotations of pages 53 and 60 of the review are incomplete since they do not continue to inform the reader that we are talking (in the book by the Kartashevs) about reconfigurable systems: on page 60, “Its basic principles are as follows: The instructions (micro instructions) store special reconfiguration codes,” and on page 53, “This is accomplished by the introduction of special reconfiguration codes stored in instructions that affect software-controlled reconfiguration in instruction microprograms.” Obviously the book is not talking about a regular computer, whose control unit has been realized by microprogramming, but about an unusual, nonstandard, reconfigurable computing system. The committee’s comments completely distort the meaning of both the book and the review.

It is not stated on page 60 that “any computer architecture defines a ‘fixed sequence of microoperations’.” This is a distortion, as is the assertion that I confuse architecture with implementation. However the reader should realize that the boundary between the two concepts may vary according to the definition of computer architecture that we agree to accept.

Daniel Tabak
Fairfax, Virginia

References


To the Editor:

In correctly comparing the TMS320 and the IBM PC in terms of FFT capability, \(^1\) a number of points must be noted:

First, the article by N. K. Riedel et al. \(^2\) does not reflect the true capability of the TMS32010. As Table 1 shows, the 320 can compute FFTs from 18 to 32 times faster than suggested in Riedel. The software architecture which gives rise to these times, described in detail in Morris, \(^3\) utilizes a radix-4 algorithm together with automated code generation techniques. \(^4\) Of these times, only the 64-point (0.54 ms) reflects the “true” 32010 “number-crunching” capability since the 128- and 256-point transforms entail movement of data between data and program memory due to the small (144-word) 320 data memory.

Second, it is more appropriate to compare fixed-point PC FFT capability with the fixed-point TMS320 (Table 1). Note that the PC is from 10.1 (80286) to 35.5 (8088) times slower for a 64-point transform, which reflects the true CPU speed ratios. For larger transforms (e.g., 256 points), the PC is only 5.2 (80286) to 17.4 (8088) times slower than the 32010...
due to the above-noted '320 data movement constraints.

Since fixed-point FFTs generally suffice to analyze data gathered from the real world via a fixed-point analog-to-digital converter, these times realistically represent both the absolute and relative PC and TMS32010 FFT performance.

Finally, for those applications which demand the increased precision of 64-bit floating-point (e.g., picture processing), the PC is at least twice as fast as suggested in Busigin.¹

Thus, we extend reader Busigin's conclusions by noting that good software can stretch both the 320 and the 80xxx processor's performance to impressive limits in computing FFTs.

L. Robert Morris
Ottawa, Canada

(Editor's note: Morris is president of DAPS Inc. and an associate editor of *IEEE Micro*.)

### Table 1.
FFT benchmarks (ms): TMS320 vs. IBM PC, IBM AT.

<table>
<thead>
<tr>
<th></th>
<th>Fixed-Point</th>
<th>IBM PC</th>
<th>IBM AT</th>
<th>Floating-Point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Riedel)</td>
<td>(DSPS)</td>
<td>(DSPS)</td>
<td>(DSPS)</td>
</tr>
<tr>
<td>No. Complex Points</td>
<td>64</td>
<td>17.6</td>
<td>0.54</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>46.5</td>
<td>2.36</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>117.2</td>
<td>6.30</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>--</td>
<td>--</td>
<td>610</td>
</tr>
</tbody>
</table>

### References

### Authors' reply
The board we constructed was to be a software development system for signal processing. The FFT example implemented on the board was merely to indicate the capability of the design. The benchmarks for this FFT routine were not given to indicate the power of the TMS32010 but to show the improvement over a straight Fortran program.

In comparing Riedel's benchmarks with those of fixed vs. floating point, it is important to note that Riedel's implementation is not a simple fixed point implementation. As noted in the article, pre-scaling and overflow detection are used to minimize round-off errors. This insures full use of the 16-bit dynamic range, increasing accuracy at the expense of time while at the same time making the implementation closer to floating point. Also, in order to meet our system specification of 512-point transforms, the radix-2 algorithm had to be used as opposed to the faster radix-4 algorithm.

As noted in Riedel in the hardware configuration section, to allow increased transform size, we opted for storing the data in the PC instead of TMS32010 program RAM as the DSPS implementation uses. For this reason our implementation is slower but more versatile.

Morris correctly points out that the major speed constraint is due to slow data transfer, particularly by the IBM PC. For instance, without using the PC DMA, which is limited in that it can't do a memory-to-memory transfer, 256 complex points require 6-10 milliseconds to transfer one way.

To conclude, let us point out that the times given in Riedel for the FFT are not only the times required to perform the actual FFT on the TMS32010 but also include all the data transfer time to and from the PC.

N. K. Riedel
D. A. McAninch
C. Fisher
N. B. Goldstein

---

**Reader Interest Survey**

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 180  Medium 181  Low 182

April 1986