POUGHKEEPSIE

POUGHKEEPSIE
Dept. 539
Bldg. 965

November 24, 1958

MEMO FOR FILE

SUBJECT: Sigma Timing Simulation

REFERENCE: Mr. Kolsky's Memo of November 7, 1958

The above memo claims performance deterioration of the Sigma Computer due to economy waves, simplification of design and the stripping of specific sections of the machine. The following general comments are in order.

1. The simulator program has been a very useful source of design information for optimizing the design of the Sigma Computer. It has been most useful in providing information on repetitious occurrences such as found in the memory bus design. I believe these results are reasonable accurate when interpreted as one Sigma configuration RELATIVE to another Sigma configuration. When the simulator is dependent on or attempting to evaluate an interlock which depends on some unique conditions, it is operating in its weakest area. The reason for this is that it is not feasible or practical for the simulator to include all of the necessary interlocks to execute the complex instruction set. Similarly, it is not reasonable to require the simulator to anticipate all the design problems which enter into the decision of how to design interlocks. However, any attempt to accurately predict or optimize the action of the Sigma computer under unique conditions should take into account all contingent interlocks and much more design information than simply general rates of major systems components.

2. The simulator program calculates the time a Sigma program would require on the ficticious system. The results are compared to the required 704 machine time to execute a 704 program similar to the Sigma program used. The idea that the results are truly representative of the actual Sigma System has some important failings.

   a. The simulator does not begin to reproduce the interlocks and buses which are being incorporated in the final design of the Sigma System. Any attempt to bridge this omission by "reasoning it out" is at best patchwork.

b. In order to facilitate comparison to the 704, the Sigma programs used for comparison do not tend to include program approaches to a problem which would be highly unique to Sigma. In other words, the comparison is made between a 704 and a Sigma having essentially only the same capabilities.

c. Item (b) involves a triple penalty to the Sigma System.

1. The use of instructions and program approaches unique to Sigma would cause the comparison to be more favorable to Sigma.

2. Any comparison of the simulator to the 704 does not include the following which are incorporated in the Sigma System:

- Increased memory capacity
- I/O system
- Multiplexing I/O equipment and computer
- Larger word size
- Internal checking
- Generalized interrupt system
- Extensive result indicators
- Elapsed and real time clock
- Sixteen index registers
- Some of the unique features of the VFL instructions
- NO-OP requirements, etc.

3. These features (some more than others) have had a powerful effect on the design of the Sigma System and appear to inhibit higher processing rates in many conventional areas. A system which has a complex task to perform suffers a great deal from control delays, physical size, interlocks, interrupt and similar phenomena. One can always find instances where a primitive machine can outperform a sophisticated one, (of the same order of size) in specific areas. We have suffered some speed penalties by incorporating these special features, we had better count the benefits in any comparison of the actual Sigma System to another computer system.

3. With the requirements imposed by the present instruction set and the technology (circuits, memory elements, transistors) available to us, the present design in my opinion is an optimum one.

In the referenced memo, two hypothetical systems are compared, one of which more nearly represents the actual Sigma System.

Since the simulator results on which comments are based, concern an area where the simulators is most likely to be weak, it is not felt that the information is reliable enough to base any strong conclusions upon it. Further, the following comments apply to questions raised in the referenced memo.

1. Lookahead Forwarding Mechanism:

This feature was not eliminated in its entirety as erroneously claimed. Forwarding is still being executed on compare of a new fetch or store address with a single lookahead address register.

2. Store Type Instructions in Lookahead:

Only one store instruction can be accomodated in general in the lookahead at any one time. However, there is a significant exception to this rule: on VFL word boundary crossover stores both result words are stored by the lookahead simultaneously.

3. Index Memory Tie Delay and Forwarding of Index Quantities for Lookahead:

This is interpreted to mean that an updated index word which is to be stored back in index memory is temporarily stored in lookahead so that if the following instructions involve only index fetches (index operand) they can be prepared and the operand fetch from main memory begun. The Lookahead would store the index quantity during some unused cycle of the index memory. The following comments apply:

a. As mentioned in 1 above, it seems that the simulator is not properly equipped to evaluate this situation.

b. An index fetch in Sigma is a 0.4 non-destructive sampling and an index store from Lookahead is an 0.8 cycle. For this reason and the fact that the index store can not be delayed quite so arbitrarily (since there is only a limited amount of lookahead levels), a design which followed this course would be inferior to the present system.

c. Even if some of the problems of (b) could be ironed out, the required additional interlocks and compares, etc. would not only deteriorate the performance under these conditions, but also under many other 'innocent' circumstances.

d. The access from lookahead over the internal bus is appreciable time-wise. In addition, time must be allotted for comparing, forwarding and checking. This negates the desirability of using lookahead as a fast access index register and simply adds more complication to the machine.

4. Store Delay Due to Reduction of Lookahead Address Registers:

There is a delay of stores only if more than one store is executed within four instructions. In general, it can be shown that stores are executed faster than the following average floating point instruction.

5. Stores of Intermediate Results in Index Memory:

This operation is permissible in Sigma and is left up to the decision of the programmer. In many cases, it can be a fast way of operating, especially when multiple fetches of the same word are contemplated.

6. Transmit Instruction Delay:

The new method of executing transmit instructions gives a rate of 1.2 usec per word as opposed to 1.4 usec when forwarding and lookahead fetching was employed. Again, the old simulator was simulating a hypothetical machine rather than the actual Sigma System.

7. Replacing Core Index Registers by Transistors:

This would not achieve a speed-up commensurate with the costs for the following reasons:

a. Only about .1 usec is used for switching of cores. The rest of the cycle is used for address compares, decoding, and bus transfers.

b. The same functions as outlined under (a) must be performed for transistor register assemblies, since circuit limitations do not permit independent and/or simultaneous operation of transistor registers. A bulk storage is the most efficient way of operating.

Erich Bloch

EB:jam
cc: Messrs: E. I. Jordan
H. G. Kolsky
R. E. Merwin