The implementation of what we believe to be the first stored-program digital optical computer is described. The implementation domain consists of lithium niobate directional couplers that are modified to provide optical control and are interconnected with single-mode fiber. The architecture is also the first to employ time-of-flight synchronization. That is, there are no flip-flops used as synchronizing memory elements. Synchronization is achieved by the precise timing of the arrival of information at all points of interaction. The design is a minimal one, employing only 62 directional couplers. Previous papers have discussed the primary architecture and synchronization conditions for the machine. Here we focus on the secondary architecture, construction, debugging, and performance of the machine.

1. Introduction

The first general-purpose electronic computer, the ENIAC, was built by Eckert and Mauchly at the Moore School of the University of Pennsylvania in 1945.1 Early examples of other stored-program computers included the EDVAC, EDSAC, and the IAS.2 Although these computers were fairly simple, they contained the same basic architectural features found in most modern computers. These computers were considered to be general-purpose machines because of their programmability and their ability to perform conditional jumps. It is the programmability that distinguishes these machines from earlier calculators and special-purpose processors.

In these early computers, propagation time between logic devices was negligible compared with device switching times. Today, electronic computers are approaching the speed at which the propagation time of signals between gates cannot be neglected. Although electronic logic gates continue to become faster, smaller, and closer together, the propagation of signals between gates is significantly influenced by the resistance and the capacitance of the signal path. This influence manifests itself by increased signal dispersion and by variations in propagation time. This problem is one of several reasons for the recent interest in optical computing as optical signals are, in general, much less susceptible to effects that limit or alter propagation speed.

Electronic computers have traditionally used flip-flops for synchronization. One reason for the popularity of this synchronization method is that it is tolerant to differences in signal propagation speed that are due to manufacturing, temperature, and capacitive effects. However, the use of flip-flops for synchronization adds additional delays to signal propagation, limiting the maximum operating speed of the system. Without the use of latches, signals flow constantly, never stopping, and pass through fewer gates. We refer to this as a time-of-flight design. To exploit a time-of-flight design, we must synchronize signals arriving at all gates by ensuring that all inputs to a gate arrive at the same time and place, that is, that they are coincident. The tight control over propagation delay possible in optical systems permits digital optical designers to explore time-of-flight synchronization techniques in ways that electronic designers cannot. This timing control allows speed-scalable designs in which the clock rate of the system is dependent only on the distance signals travel; there are no capacitive effects that limit the clock rate.

There are two common approaches to optical computer design. One is to design new architectures with massive parallelism to take advantage of freespace interconnections.3–6 The other approach em-
employs optical devices in more conventional computer architectures, exploiting the advantages of optics wherever possible. This latter approach permits the optical computer architect to make use of the tremendous body of knowledge about computing that has been gained over the past fifty years while incorporating new architectural features allowed by the use of optics instead of electronics.

This paper describes the implementation of the stored-program optical computer (SPOC), to our knowledge the first demonstration of a general-purpose SPOC. Section 2 describes the research that culminated in the demonstration of the computer. Section 3 outlines the goals and motivations of the project. Section 4 outlines the implementation domain. Section 5 describes the SPOC architecture, including instruction set and timing details. The computer subsystems are described in Section 6. Section 7 details the implementation of the computer, and Section 8 discusses the construction and debugging process.

2. Previous Research

The primary architecture of SPOC was described in a paper by Heuring et al. The architecture is defined as primary because it assumed ideal switches with no cross talk or loss. Because the goal of their research was to develop a proof-of-principle machine, and as the only optical switches available were quite expensive, the authors chose the simplest, bit-serial design, with a single accumulator, program counter (PC), instruction register, arithmetic and logic unit (ALU), and a small delay-line memory. This paper describes the secondary architecture, that is, the architectural modifications that had to be made to compensate for power loss and cross talk. Synchronization of a complex optical system with many components and many feedback loops is a complex matter. Pratt and Heuring developed a graph-theoretic model that permits computation of the synchronization conditions in the time-of-flight domain, and they also developed a graph-theoretic model that permits the estimation of loss and cross talk.

Before the optical switches were available, Yadowsky demonstrated a time-of-flight mock counter to validate the time-of-flight design principles. The optical switches were modeled electronically by the use of optical receivers and transmitters on the inputs and outputs and electronic circuits to perform the switch logic. Signals propagated through an optical fiber between the mock switches. The mock counter allowed us to experiment with time-of-flight synchronization in the absence of the optical switches.

Before the entire computer was built, several circuits and subsystems were built in order to develop construction and debugging techniques and to test the optical switches. A 50-MHz optical counter was built by Benner et al. Feuerstein et al. demonstrated a 100-MHz optical counter using an improved counter design. Jordan and Heuring also demonstrated the use of time multiplexing, in which multiple independent machines operate simultaneously on the same hardware by interleaving bits in time. Two independent 50-MHz optical counters were operated simultaneously when a 50-MHz counter was built and a 100-MHz clock was used. Soukup et al. demonstrated the 50-MHz memory subsystem, including the 1024-bit delay-line memory, the 16-bit memory counter, and the address comparator. Error rates and timing tolerances were measured for the circuits in order to determine the reliability of the system. Limits on memory size were determined based on phase errors over acceptable temperature variations.

An electronic monitor was designed by Lode and Heuring to interface with the computer. The monitor was designed to permit an operator at a workstation to read and write the memory, accumulator, and PC, and to provide run, halt, and single-step control. The original monitor system was a central unit with coaxial cables connecting it to the optical computer. Because long cables were needed for the connections, there were problems with electronic noise and attenuation. The monitor was redesigned by Feehrer as a distributed system, with a transistor-transistor logic board to interface with the workstation and high-speed emitter-coupled logic boards located near the optical circuits with which they were interfaced.

3. Goals

The primary goal of the SPOC project was to demonstrate a general-purpose SPOC in which signals propagate at the speed of light without latches. Building a computer with the ability to store and modify its own programs was essential to the advancement of optical computing into the general-purpose computing domain. Optical processors built in the past have usually needed an electronic host computer to provide control. SPOC has both code and data available in its own optical memory system. The ability to manipulate a program as data has been called the Von Neumann breakthrough and is a fundamental principle of computer science. From a computer science viewpoint, it is the stored program, rather than the computational ability, that distinguishes a computer from other digital processors.

Because the optical devices available at this time are expensive, it was desirable to minimize the hardware complexity of the computer. For this reason, the bit-serial architecture used in SPOC is very simple, permitting the system to be easily scaled to take advantage of new devices as they become available. The specifications of the computer are shown in Fig. 1.

4. Implementation Domain

The bit clock is the primary clock for the computer and is used as a timing reference for all other signals. A bit time is the period of the clock signal: 20 ns. The word size (WS) is defined as the number of bits contained in a word of data (16), while the word
Clock Speed: 50 MHz
Word Size: 16 bits
Architecture: latchless, bit serial, word addressable
Memory Size: 64 words
Active Components: 62
Operations: load complement store rotate right add jump OR conditional jump AND

Fig. 1. SPOC specifications.

The period is WS bit periods. The word clock signal is asserted every word period to mark the location of the low-order bit in a word.

The system requires ~50 mW peak optical power and five 1300-nm laser diode sources. We use return-to-zero bit encoding, which means the data are valid only during the first part of the bit period. A logic 1 is coded as a pulse of light whereas a logic 0 is coded as the absence of light. The second part of the bit period always contains no light (logic 0). All lasers are modulated by a single 50-MHz electronic clock source with a 20% duty cycle.

The optical switches used are AT&T lithium niobate (LiNbO₃) polarization-sensitive directional couplers. The switch is shown schematically in Fig. 2(a). It is an optical device that can be used as a three-input, two-output logic element. When there is a logic 0 at the control terminal (or C terminal), no voltage is applied to the switch, and light entering the A and B inputs will exit the diagonally opposite outputs. This is known as the cross state. With a logic 1 at the C terminal, 4.5 V are applied to the switch, and the signals entering A and B will exit the outputs directly across from their inputs, which are defined as the bar state. To use this device as an all-optical component, we connect an optical detector and amplifier to the control electrode, as shown in Fig. 2. Some additional electronics are used to stretch the incoming pulse to aid in synchronization. Details of the electronic design are given in Ref. 12. Propagation time through the drive electronics is measured and compensated for during system design. With the addition of the C-terminal electronics shown in Fig. 3, the directional coupler becomes an optical switch with entirely optical inputs and outputs. Butterfly polarization controllers are used to adjust the polarization of incoming light to minimize cross talk. These controllers typically add ~1 ns of additional latency. Polarization-maintaining fiber was not used because of the unavailability of connectors to connect this fiber when the machine was being constructed. There are also two dc bias adjustments that are used to bias the control electrodes correctly. The switches are packaged in sets of six, so the switch and its C-terminal electronics are assembled together into a six-pack module. Typical optical propagation delay through a switch is 4 ns, with 6-ns latency in the C-terminal electronics. Typical attenuation through the switch is 5.5 dB, and cross talk is ~−20 dB. Testing showed that if a weak pulse arrives at a C terminal soon after a larger pulse, it may not be detected by the C terminal. The largest difference in adjacent pulses that can be tolerated at a C terminal is limited to ~5 dB. Differences above this amount lead to immediate incorrect operation of the gate, and values below approximately 4 dB permit the gate to operate without error. The minimum signal level that can be detected at the C terminal is ~−20 dB. The monitor system interfaces with the optical computer through the C-terminal electronics, as shown in Fig. 3.

The fused fiber coupler shown in Fig. 2(b) is employed as an optical splitter for signal fan-out and as an optical combiner is used to perform a passive OR.
Signals that enter a 50:50 coupler suffer a nominal 3-dB loss. There is some excess loss, usually ~0.2 dB. The most commonly used coupling ratio in SPOC is 50:50, in which half of the light from each input is coupled to each output. Other ratios used in the computer are 60:40, 80:20, and 85:15. These odd coupling ratios are used to balance signal power.

A typical length is ~30 cm, or 1.5-ns latency. Care must be taken when performing a passive or because of the polarization dependency of the optical switches. In general, it is unlikely that the two inputs to the coupler will have the same polarization state. So, if the output is connected to the A or the B input of a switch, it can be difficult to adjust the polarization for proper switch operation. To avoid this problem, an additional butterfly polarizer may be used on one or both of the coupler inputs, permitting the polarization of the two inputs to coincide. Destructive interference may occur when the coupler is used as a passive or if the two inputs to the coupler come from the same laser diode. Measurements of this effect have shown that it is not significant in practice with our laser sources.

Delay elements [Fig. 2(c)] delay signals by some fixed amount of time, which is indicated schematically by a number or letter below the device. By convention, the absence of a number below the device indicates a delay of one bit period. The delay element is implemented with a single-mode optical fiber. Optical fiber is also used throughout the computer for synchronization and device interconnection. Fiber delay varies with temperature, but at 50 MHz this effect is significant only for large temperature variations or long fibers. The latency of a fiber, \( \delta \), is calculated with the length of the fiber, \( l \), the index of refraction of the fiber, \( n \), and the speed of light in a vacuum, \( c \), as

\[
\delta = \frac{l}{c} n. \tag{1}
\]

5. Stored-Program Optical Computer Architecture

Instruction Format

The 16-bit instruction used in SPOC consists of two parts: a 6-bit opcode field and a 10-bit address field, as shown in Fig. 4. The six opcode bits are further divided into three source-select bits, a destination-select bit, and two PC control bits. The source-select bits are used to select source operands and ALU operations. The destination-select bit selects the destination for the operation, either the accumulator or a memory location. The PC-control bits are used to determine whether the computer will continue sequential execution or perform a conditional or unconditional jump.

The lower 10 bits of the instruction are reserved for the address field. When the source or destination of an instruction is a memory location, the address field contains the address of the memory operand. If the PC-control bits indicate an unconditional jump, the address field contains the target of the jump. If the instruction does not use a memory operand, the address field is unused. Only the lower six bits of the address field were used in this implementation because the memory loop stores 64 words.

All three opcode fields are used for every instruction. This means that every instruction, including jumps, will select a result by using the source bits and will store that result into either the memory or the accumulator. Also, every instruction causes a search for the operand address, even if no memory operand is used. These characteristics stem from the simplicity of the architecture. Several more devices, as well as more opcode bits, would be necessary to eliminate unwanted operations and memory fetches. This would greatly increase the complexity of the design. The simplicity of the architecture also results in several illegal or undesirable instructions.

The SPOC instruction set is shown in Table 1. The format of the instructions is also shown in the table, where the addr and dest terms refer to the type of operand. The addr field is used to specify the address of a memory operand. It is also used in the jump (JMP) instruction to indicate the jump target. The dest field indicates the destination for the result produced by the instruction. It can either be an

<table>
<thead>
<tr>
<th>Table 1. SPOC Instruction Set</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mnemonic</strong></td>
</tr>
<tr>
<td>Load and store instructions</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Arithmetic instructions</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Branch instructions</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
address, to indicate a memory operand or labeled as ACC to indicate that the destination is the accumulator.

Conditional jumps cause a jump relative to the PC, rather than a jump to a specific address. In this architecture, a conditional jump causes the PC to be incremented twice rather than once, or double incremented. This is called a skip. However, the skip operation is not a separate instruction; it is added as a suffix to another instruction. There are two versions of the skip operation: the skip-if-not zero and the skip-if-carry. The skip-if-not-zero operation causes a skip if the result produced by the operation is not 0 whereas the skip-if-carry operation causes a skip if there is a carry out of the ADD operation.

Instruction Timing

Figure 5 is a block diagram of the computer with control operations shown in heavy lines, and Fig. 6 shows the details of the control signals and data paths. During the description below, the reader will wish to refer to these figures in turn. SPOC always operates in one of two states: the instruction-fetch state or the operand-fetch state. Execution begins with the instruction fetch, during which the computer searches for the current instruction to execute. This search is performed by the address comparator: the lower six bits of the PC are repeatedly compared with the lower six bits of the memory counter (MC) until the location of the current instruction is found. When it is found, a memory-found (MEMF) signal is generated, which is decoded into an instruction-found (INSTF) signal by the state control logic. During this word period, the contents of the memory location pointed to by the PC are loaded into the instruction register (IR) by a train of sixteen pulses generated by the INSTF signal. When this operation is complete, the machine changes state to perform an operand fetch.

During operand fetch, the address field of the IR is compared with the memory counter as these bits of the instruction make up the address of the memory operand. To keep the design as simple as possible, the operand fetch is performed even if there is no memory operand. When the lower bits of the memory counter match the address field of the instruction, another MEMF signal is produced, which is decoded as an operand-found (OPRF) signal. OPRF is used by the opcode extractor to convert the opcode bits to parallel. The source-select opcode bits force the output multiplexer (MUX) into the correct state,
gating the desired ALU result out of the MUX. The destination-select bit is then used to select either the memory or the accumulator to receive the ALU result. If the PC-control bits are set to indicate a jump or a skip, the PC is either loaded with the contents of the IR for an unconditional jump or incremented again for a skip. Although the entire instruction is loaded into the PC during a jump operation, only the address field (lower six bits in this implementation) is used in the address comparator. After operand fetch, the computer returns to the instruction-fetch state, comparing the new contents of the PC with the memory counter in search of the next instruction. This process is continuous; there is no halt instruction (although it can be simulated with a jump here instruction).

Synchronization
To switch a signal arriving at the A or B inputs of an optical switch, a signal must arrive at the C terminal before the signal arrives at the other input, and the switch state must stay constant until the other signal has been fully routed. As all signals in the system have the same duty cycle, a signal arriving at a C terminal is lengthened, or stretched, in order to switch the other signals fully to their correct outputs. Although the system could operate by using just a few percent pulse stretching, SPOC stretches C terminal pulses by ~20% of the clock period (4 ns). This provides a larger margin for error in fiber lengths and tolerance of timing shifts that is due to temperature effects. Restoration of amplitude and timing of attenuated signals is done by clock gating or regeneration, in which the attenuated signal is used to switch a copy of the clock into the circuit. Switches used for power restoration are called regenerator switches.

The SPOC architecture uses a single clock source that is fanned out to 18 circuits. The word clock signal is used throughout the computer to indicate word boundaries within a stream of bits and is used as an increment signal for the memory counter. An electronic word clock signal modulates the C terminal of an optical switch, generating an optical version by switching an optical clock. Some electronic fan-out of the clock to the five laser sources was employed, but most fan-out was accomplished optically. Using one arbitrary copy of the clock as a reference, we adjusted all other copies optically until they matched the necessary timing to within 0.5% of the clock period.

6. Subsystems
Figure 6 shows a schematic of the computer’s secondary architecture. Regenerator switches are shaded in the figure.

Memory Subsystem
The memory subsystem, including the memory, address comparator, and memory counter is described in detail by Soukup et al.15,16 The accumulator is a one-word memory loop used for operand and result storage. Data are written to the memory or accumulator by decoding the OPRF signal by the use of the replicated destination-select opcode bit.

State Control
The state control logic includes the state toggle, which maintains the state of the machine (operand/instruction fetch), and the memory-found decoder, which uses the state toggle output to decode the MEMF signal as either OPRF or INSTF. The state toggle operates like a toggle flip-flop, although there is no steady-state output. Rather, the presence or absence of pulses propagating in the circuit determines the state.

Instruction Register and Control
The instruction register is a one-word delay line that holds the instruction that is currently being executed. When the INSTF signal is asserted by the state control logic, the data appearing at the output of the memory loop are loaded into the IR as the current instruction. The opcode extractor decodes the opcode bits of the instruction. This circuit is a serial-to-parallel converter that provides a parallel version of the opcode bits. By anding the OPRF signal from the state control logic with delayed versions of the instruction, the individual bits of the opcode are extracted from the current instruction.

To route 16-bit words by using one-bit control and opcode signals, we use pulse replicators. The pulse replicator shown in Fig. 7 takes in a single pulse and produces a train of 16 pulses. Pulse replicators are used to load the IR, memory, and accumulator, and to route data through the output multiplexer.

Program Counter
The PC performs the function of an instruction pointer and is incremented when an instruction completes. It can also be loaded from the IR to execute an unconditional jump and can be double incremented to execute a conditional skip.

This counter, like the memory counter, is designed to add the current value of the count with the increment signal and any carry bits produced by the increment. Unfortunately, this means that when the counter contains 16 1’s and is incremented, there will be a carry out of the high bit that will add with the low bit of the count during the following word period. This is not a problem in the case of the memory counter because it is automatically incremented by the word clock every word period. However, the PC is incremented only periodically (every 20 μs on average) by the OPRF signal. So, when
there is a carry out of the high bit of the count, the PC will increment itself, even if there is no increment signal. However, this does not occur because of the nature of the instruction encoding.

Arithmetic Logic Unit and Output Multiplexer

The ALU performs bit-serial computations. An interesting feature of the SPOC's ALU is that all results (sum, AND, OR, complement, etc.) are produced every word period. These results propagate to the inputs of the MUX, which determines the correct result to send to the memory and accumulator based on the opcode bits of the instruction in the IR.

The skip generator detects the conditions necessary for a skip operation and for incrementing the PC if a skip occurs. If the PC-control opcode bits (PCSEL0 and PCSEL1) indicate that a skip operation may occur, either the carry out of the ALU sum or the output of the MUX is tested to see if a skip occurs. In that case a pulse is sent to the PC to increment it a second time (after OPRF has already incremented it once).

7. Implementation

Computer-Aided Design Tools

The time-of-flight architecture demands strict control over signal timing for synchronization. The timing calculations for the entire computer were far too complex to manage easily by hand, so software tools, Hatch and XHatch, were developed to aid in this task. Hatch was used to validate the architecture and to design the counters and memory subsystem discussed in Section 2. XHatch was used to design the final architecture.

XHatch provides a graphic environment for the design and the simulation of optical circuits. With the delay distribution algorithm developed by Pratt and Heuring, signals are aligned with each other as they enter optical devices by adding delay to match the signal paths. The algorithm is based on a directed-graph representation of the optical circuits, with weighted edges to represent time-of-flight propagation delay. XHatch also permits the designer to use lumped delays initially, which are later distributed throughout the optical circuit, and provides a report of the power levels of signals as they enter or exit a device. This report is a useful tool for handling regeneration and fan-out issues.

Clock Rate

Although the computer operates at a clock rate of 50 MHz, it was important to permit a possible future version to operate at 100 MHz with as few architectural modifications as possible. Such modifications would be required if the minimum propagation time around a feedback loop were greater than 10 ns, for example. Implementation of a 10-ns one-bit feedback loop requires that only one switch be in the loop. A violation of this restriction occurred in the original state toggle circuit of Fig. 8(a). Figure 8(b) shows the circuit required for 100-MHz operation. Note that the toggle signal is used to produce the true and complemented form of a signal. Most of the signals that require the use of a regenerator switch to increase the power level of heavily used signals, another solution was to unanticipated losses in optical fibers, connectors, and devices.

Although some fan-out problems were solved by the use of a regenerator switch to increase the power level of heavily used signals, another solution was to make use of both the true and the complemented versions of a signal. Most of the signals that require a large fan-out are already available in both true and complemented form, including the memory, accumulator, and IR. One of the circuits for which the complemented signal was used is the opcode extractor, as shown in Fig. 6. This technique was used throughout the computer to balance signal loads. Optical couplers with odd coupling ratios were also used to balance power distribution.
A four-level table was designed to house the computer, with the two upper levels for the computer and the lower levels for lasers and power supplies, as shown in Fig. 9. From the lower shelf, level 1, wires were routed to the upper levels, where a power bus distributes the necessary voltages to each module. The optical couplers and fibers associated with clock fan-out were placed on level two with the lasers and laser temperature controllers. Optical fiber was routed along Velcro at each level, permitting the fiber to be secured to prevent movement of the fiber from changing the polarization of the light traveling through it. Fibers that interconnect modules on different levels were routed through slots in the top-level shelf so that they were not exposed.

After the SPOC table was complete, the optical circuits for individual subsystems were mapped onto the hardware, based on the goal of minimizing fiber interconnection lengths. Functional blocks were assigned stations on the table in groups so as to keep heavily interconnected circuits near one another. After circuits were assigned to individual modules, the final lengths of interconnecting fibers were entered into the XHatch design. All fibers that connected one module to another were assigned a minimum length according to the distance between the modules. A minimum length of 30 cm was assigned to each fiber that connected two terminals on the same module. The design used 62 optical switches in 11 switch modules, 200 optical fibers, and 75 optical couplers.

8. Construction and Results

The construction sequence is discussed in this section. First, the clock signal generator and lasers were placed on the table and connected into the dc power distribution bus. Next, fibers and splitters were laid out along level two to fan out the laser clock sources. Then construction began on individual subsystems of the computer. Soukup presents a detailed, step-by-step description of the construction of the memory counter, address comparator, and memory loop. Main presents details of the complete SPOC construction. The monitor system was used for testing circuits throughout the SPOC construction.

Debugging

The monitor system was vital for debugging as it permitted access to the memory and the registers and provided execution control. Several new debugging techniques had to be developed because of the time-of-flight design of the machine. In a time-of-flight design, the system clock can operate at one and only one frequency, so the usual technique of increasing of decreasing clock frequency to find timing-related errors was unavailable to us. Furthermore, it was impossible to stop the clock to examine the machine state as no signals are latched in the time-of-flight design. Instead, we had to trigger our oscilloscope and monitor on specified events. A particularly useful event was the occurrence of a particular count in the memory or program counter as these counters controlled much of system operation. Software routines were developed to test various computer subsystems and sequences of operations. One such routine, called test.asm, is shown in Fig. 10. This code exercises all of the computer’s capabilities. It tests all possible paths through the output MUX (SUM, ZERO, ACC, MEM, MEM*, RORA, AND, and OR), and all possible PC-control operations (JMP, SNZ, and SCY). To test the skip generator, we test both outcomes of a skip: skip occurs and skip doesn’t occur. At the

/*---------------------------------------------*/
/* test.asm */
/* Todd Main, 10/1/93 */
/*---------------------------------------------*/

JMP A ;NOP
A LDA M1 ;ACC <- C35A
ADDSCY ACC,M2 ;M2 <- 4E61
ADD M1,ACC ;(should not execute!) ACC <- 1BB
CLRM M1 ;M1 <- 0
STASNZ M1 ;M1 <- C35A, SKIP
LDASNZ M1 ;(should not execute!) ACC <- 4E61
ADDSCY M3,ACC ;ACC <- C35B, NO SKIP
STA M4 ;M4 <- C35B
NOTM M2,M2 ;M2 <- B19E
RORA ACC ;ACC <- E1AD
OR M2,ACC ;ACC <- F1BF
STA M5 ;M5 <- F1BF
AND ACC,M4 ;M4 <- C11B
B JMP B ;NOP

.DATA
M1 C35A
M2 8B07
M3 1
M4 FFFF
M5 0

.END

Fig. 9. SPOC.

Fig. 10. Test program: test.asm.
end of the program is a JMP here instruction to provide a halt. This tests the long-term ability of the computer to perform jumps without error.

Other code was developed to test individual instructions multiple times, such as 30 ADD's in a row. This was done to check for borderline operation of individual paths through the output MUX and into memory or the accumulator. If the instruction worked most of the time, but failed occasionally, these routines might catch the error, whereas programs like test.asm would miss it. Two other programs, checkacc.asm and checkmem.asm, perform all operations and store all results in the accumulator or memory, respectively. These programs were used to evaluate the performance of the memory against that of the accumulator, to see whether errors are duplicated when storing results in different places.

System Reliability

Three factors influence the reliability of the computer: polarization, timing, and signal power.

Although the polarization sensitivity of the LiNbO₃ optical switches is an inconvenience, it is not a serious problem with respect to short-term reliability. However, it does affect the long-term reliability of the system. Although the fibers that interconnect components are fastened to the SPOC table with Velcro, casual contact can disturb the fibers and change the polarization of the light passing through them.

Synchronization, or timing, was not a problem in the SPOC architecture. Timing to the subnanosecond regime was easily achieved by cutting fibers to the correct length. The 20% stretch in the C terminal also provided for a large timing error window.

Signal power has been the most obvious factor that limits short-term reliability. The design goal for minimum power at a C terminal was ~ -20 dBm, and bench testing of the optical switches showed that all switches could easily detect such a signal. However, in the electrically noisy environment of the SPOC table, in which all switches in each module were in use, some switches had trouble detecting -20 dBm signals. This greatly affected the short-term reliability of the system.

Performance Limitations

The computer's performance is limited by the clock rate, the memory access latency, and the bit-serial architecture. The maximum clock rate is limited by the components that make up the system and the minimum length of interconnections. It would be difficult to increase the clock rate beyond a few hundred megahertz with discrete components because the distances between components would be larger than the necessary interconnection lengths. However, an integrated version could operate in the gigahertz range.

Another way to increase performance is to time multiplex multiple computers on the same hardware. The latchless nature of the design permits N machines to operate simultaneously on the same hardware by simply increasing the clock rate of the computer by N. Each individual multiplexed computer operates at the clock speed of the original design, that is, 1/N of the new clock speed. The multiplexed computers could communicate through a time-slot interchanger. Such a scheme for increasing performance is most useful in situations that involve high-bandwidth, long-latency devices. An integrated, multiplexed version of SPOC that could operate at 20 GHz has been proposed.

The high cost of the optical switches used for the SPOC project necessitated a simple memory system: the delay-line memory. The average memory access time is roughly one half of the memory period, and as each instruction requires two memory accesses, instruction fetch and operand fetch, this averages to one memory period per instruction. One solution to this memory latency problem is to use N individual one-word memory loops, where N is the number of words of memory. However, this solution requires 2N optical switches just for the loops as well as a great number of switches to replace the memory counter with a conventional address decoder. Several alternative memory systems have been examined in detail. The solution with the best cost-performance characteristic is somewhere between the completely serial and completely parallel extremes.

Although much attention has been placed on the bit-serial nature of the SPOC architecture, this characteristic is not an inherent constraint of time-of-flight computers. Using a 16- or 32-bit parallel architecture would increase performance by a factor of ~ 10 compared with a serial design because of the resulting reduction of the clock rate. However, the number of optical devices required by such a design would be likely to exceed the capability of current optical device technology. A parallel design built with discrete components would require a room full of devices unless it were built by the use of integrated optic technology. However, the field of integrated optics permits the use of perhaps 100 interconnected directional couplers on a chip. A parallel version of this architecture would require ~ 1000 devices.

9. Conclusions

We have demonstrated what is to our knowledge the first SPOC. Operation of the SPOC has proven that it is possible to build large optical systems by the use of only time-of-flight synchronization without latches. Synchronization was accomplished by electrically stretching the pulse at a C terminal and by clock gating.

The SPOC provides the commonly found ALU operations, as well as conditional and unconditional jumps. Its general-purpose instruction set is versatile enough to permit implementation of any programmable algorithms. Although there are several performance limitations such as clock rate, memory latency, and bit-serial encoding, these could be overcome in an integrated version.

This research was funded through the Optoelectronic Computing Systems Center by National Sci-
ence Foundation grant ECD 9015128 as part of the Engineering Research Center program and by the Colorado Advanced Technology Institute, which promotes advanced education and research at universities in Colorado for the purpose of economic development.

References