EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN/EF 87-10
3 December 1987

THE DESIGN OF A CUSTOM CHIP SET FOR THE ALEPH EVENT BUILDER

P. Battaiotto
CERN, Geneva, Switzerland and U.N.L.P/CONICET, Argentina

A. Marchioro
CERN, Geneva, Switzerland

R. Marbot, J.Y. Parey and A. Violet
Ecole Polytechnique, LPNHE, Paris, France

S. Newett
Rutherford and Appleton Laboratory, Chilton, Didcot, Great Britain

Summary

This paper describes the design and implementation of two custom integrated circuits for a Fastbus module to be used in the data acquisition system of the ALEPH experiment at LEP. The aim of this project was to replace a large number of conventional circuits in a DMA controller and to reduce the board size needed. The architecture of the circuit is explained, together with a brief discussion of the tools used. A brief cost analysis comparing the ASIC technology with a traditional approach is given.

Submitted to Nuclear Instruments and Methods in Physics Research

1. Introduction

The present generation of High Energy Physics experiments at LEP requires the introduction of new technologies to satisfy the increasing demand for density and speed of the various data acquisition systems. Most experiments have chosen to implement a system consisting totally, or at least partially, of Fastbus (Ref. 1). Large board dimensions and the availability of more power than for other standards, was a promising argument when this system was proposed. Nevertheless, the complexity of the Fastbus protocol and of the functions required for many modules, especially Fastbus CPUs and intelligent modules in general, indicated the necessity for integrating complex modules using VLSI technologies.

As a typical example, the ALEPH experiment at LEP will be equipped with more than half a million analog channels, generating ~25 Mbyte of data per trigger, at a primary rate of 10 triggers/s. A massive data reduction will take place directly following the digitization of the analog signals. It is foreseen that about 150 Fastbus crates will be filled with detector-dependent digitizing electronics and 'intelligent modules' for data reduction. The ALEPH read-out system is organized hierarchically in three main levels. To minimize the dead-time, the system is buffered at various levels, and data travel independently through the branches of the acquisition tree. These asynchronous data streams are progressively merged together toward a main data acquisition computer. Data rates at the level of the main data acquisition computer are expected to be around 1 Mbyte/s, with a peak rate of 10 Mbyte/s, comprising ~10 events of ~100 Kb each. (Ref. 2).

2. The ALEPH Event Builder

One of the principal modules in this system is the Event Builder, which takes its name from the function it performs. It sits at the root of each different subdetector tree and its task is to collect and re-synchronize the data from the lower level acquisition modules.

The Event Builder is a 68020 based Fastbus controller, where the Motorola coprocessor concept has been exploited to build a Fastbus master port controller. The Fastbus transaction routines are implemented directly as new 68020 instructions and the programmer can access Fastbus with an extended set of 68020 instructions (Ref. 3). In the framework of the Event Builder project we have decided to integrate parts of an existing design, implemented in discrete logic, into a VLSI chip set. This project has been carried out as a common development between groups from three different European High Energy Physics Laboratories: CERN in Geneva, the Ecole Polytechnique in Paris and the Rutherford Appleton Laboratory in England.

3. The Event Builder architecture

The Event Builder is composed of several functional units. In particular, the coprocessor based Fastbus master port is controlled by a microcoded machine built around an AMD 2910 sequencer. This unit controls the complex, but user transparent, protocol with the 68020, and also performs all the single word transfer transactions in Fastbus. This approach guarantees great flexibility, but is clearly less satisfactory for block transfer operations, where speed is the most important concern. As can be seen from Fig. 1, the Event Builder architecture has a dual bus structure that allows the 68020 to execute its program while the DMA controller transfers data to Fastbus from the Event Memory and vice versa. The system is built on two Fastbus boards, one with the 68020 and Fastbus coprocessor and the second with the Event Memory. The Event Memory is dual ported, allowing concurrent access from the CPU board and the Fastbus slave port. It is required that the 68020 be able to access the Event Memory even during a block transfer operation. This necessitates a special controller to supervise the data flow between Fastbus and the Event Memory module. This controller must understand the complex Fastbus protocol for block transfer and interface properly to the Event Memory module. The Event Memory is connected to the CPU board via a bus which is almost identical to the standard 68020 bus.

Today several commercial DMA controller chips are available, but none of them properly fits our requirements and therefore a dedicated design has been decided upon.

The major components for this DMA controller are shown in Fig. 2 and listed below:

1) a state machine to control the operation of the different sections
2) an arbiter to grant (on a cycle by cycle basis) the access to the Event Memory bus between the DMA controller and the 68020
3) a 24 bit read-write word counter to keep track of the number of 32 bit words to be transferred. (No byte or 16 bit-word mode is allowed in this implementation)
4) a 32 bit bi-directional register which offers one level of pipelining between Fastbus and the memory, allowing the memory cycle and the Fastbus cycle to occur simultaneously
5) a 32 bit read-write memory address counter
6) an instruction decoder which controls strobing of various bits of information from the sequencer to the controller
7) time-out counters for memory access, DS#DK and WAIT

Furthermore, an important requirement for the controller was to be able to access concurrently the Fastbus and the 68020 side.

The implementation of this circuit in discrete logic consists of about 52 MSI, mostly 20-pin F-series TTL chips, occupying a significant area on the Fastbus board. The power consumption of such an implementation is of concern, because a fully populated Fastbus board with more than 400 TTL chips and the necessary ECL converters and drivers easily exceeds the maximum power consumption of 75 Watts permitted by the specification. In addition, more space was needed for further program memory for the 68020. Fast static memories are packaged in large chips and the solution adopted, in order to save space, is based on daughter boards, with surface mounted devices on both sides. Such a board can hold 1 Mbyte of static memory with 1 wait cycle for a 20 MHz 68020. Four boards can be plugged on to the CPU card.

4. The ASIC approach

Initially a solution where the whole DMA controller would fit on one VLSI chip was investigated, but considerations on packaging costs and power dissipation suggested a chip set implementation.

In chosing a method of integration, the criteria were that the circuit should be at least as fast as the discrete version, it should be reasonably cheap and have a fast fabrication turnaround. Clearly there was no argument for a full custom design, but a gate array or standard cell implementation would be suitable. On a gate array, the user designs one or two metallization masks, which interconnect the transistors on a pre-formed array. Since the design requires only one or three masks (two for routing and one for vias on a double layer metal process), it is inexpensive. In a standard cell design, a library of cells is used. Cells are placed only where required. A full mask set is required, but routing channels need only be as wide as required by the circuit.

The final choice was to implement the circuit with a technique which the manufacturer calls 'optimized gate array', which is a technology between a standard cell and a gate array chip. It offers 2 µ CMOS double metal layer technology and a nominal speed exceeding 25 MHz for internal clocks. Like in standard cells, only as many transistors as are needed are placed, and routing channels are only as wide as they need to be. The basic element is a cell consisting of a transistor pair and the circuit primitives are formed from groups of these. The technology could house the entire circuit in one chip (estimated to be about 3000 gate equivalents). However one of the requirements was to have 24 mA drive on a 100 pF load on most bidirectional I/O pins; to be able to drive the bus to the Event Memory directly, without the need for additional buffers. This was estimated to be too difficult to achieve for the 64 address and data lines and instead a sliced solution was chosen. The data path is split into four 'data' chips, and a 'controller' chip handles the protocol. Considerations on packaging cost also suggested a less integrated solution. For the same reasons the least expensive 48-pin dual in line package was selected.

The controller chip is used to control the four data chips. The circuit consists of a state machine, an arbitration circuit, a 20 bit word counter (reduced as compared to the discrete version) and all necessary time-out counters. The word counter can be loaded and read by the sequencer, and is decremented by the state machine.

To maximize throughput during block transfer, the architecture of the controller allows concurrent access (instead of sequential) to the Fastbus and the 68020 side. This is achieved by using a pipeline register implemented in the data chip.

The state machine has 8 states (Fig. 3) and it is controlled by 12 inputs which come either from Fastbus, the memory or are generated inside the chip itself. It rests in the idle state (state 1). When a memory block transfer begins, it goes to state 2 where it gives either a memory request (MEMREQ) or a Fastbus data strobe toggle (DST) signal, depending on whether it is doing a write or a read transfer. Next, the machine toggles between states 3 and 5, generating MEMREQ and DST each time state 5 is entered. When the word counter has counted down to one, the state machine goes to state 6 where either a DST or a MEMREQ signal is issued depending on the direction of the transfer. Normal completion of the transfer is through state 7. During a transfer it may wait in state 3, for memory ready (MREADY) and/or for a DS equal DK condition. Whilst in this state the timeout counters are running. If a Fastbus WAIT signal is detected, state 4 is entered until the condition disappears or a WAIT time-out occurs. The time-out periods are programmable from the sequencer. They are 2, 4, 64 μsec and infinite and 1, 2, 12 μsec and infinite for DK timeout (DKTIMEOUT) and memory error (MEMERR) respectively. The Fastbus Wait timeout is also programmable for 8 μs, 1msec, 16 msec and infinite. If a timeout occurs, or an SS error or a parity error is detected, the transfer is aborted, state 8 is entered and a status register is set. The state machine is clocked from an external source and provides divisions of the basic frequency for the internal timeout counters. The basic clock frequency is 20 MHz but some controller chips have been tested up to 33 MHz.

The arbitration circuit allows the 68020 to make a request to use the bus to reach the Event Memory, inhibiting the DMA access for a period. The 68020 can make a read, a write or a read-modify-write cycle. The arbiter works on a first-come first-served principle. If a read-modify-write cycle is performed, the state machine is kept waiting for the total duration of the access. Arbitration occurs on a word by word basis. This guarantees a very short latency when the 68020 requests a bus access.

The data chip contains two 8-bit back to back registers with tri-state outputs for pipelining the data transfer from the Event Memory to the Fastbus ECL interface. The registers are clocked with signals coming from the controller chip. An 8-bit up counter with tri-state outputs is included for address generation, and a ripple carry output is available to perform the cascading function. It also contains an 8-bit down counter with fast ripple carry logic to implement a word counter in applications not requiring the same controller chip. All these counters may be read or preset. The chips are capable of driving the required load as specified. Six power pins are necessary on each of the two chips. The manufacturer's experts have kindly designed special new I/O cells to satisfy our requirements.

In order to improve the testability of the design, both chips contain extra circuitry to access critical sections during testing. The controller chip also includes circuitry to simulate the memory being accessed during test.

Table I. contains a brief comparison between the old implementation with discrete logic and the ASIC approach.

Table I.

	Discrete design	ASIC design	
Number of chips	52	5	
Number of pins	~1000	240	
Current consumption	3500	100	mA
Working frequency	20	up to 33	MHz

5. The design tools

A SOLO 1000[¶] system was used. This system has a library of primitives ranging from simple gates to complex functions, including macros for repetitive structures, memories etc. Mostly components from the library were used, but for critical sections optimized components were built from elementary parts.

Whereas a PCB designer would probably have a prototype or wire-wrap board on which to cut tracks or replace wires, an IC design has to be correct before it goes to fabrication. Therefore, a good CAD package is essential. The debugging is done on a computer, using a simulator, which displays waveforms as on a logic analyzer. The simulator has to take into account circuit delays attributable to fan-out and routing capacitance for each individual gate. Fault coverage also has to be simulated to guarantee that the testing procedure effectively checks the parts.

The design was carried out simultaneously for the two chips at Ecole Polytechnique (data chip) and at the Rutherford Appleton Laboratory (controller chip), each using a CAD

¶ SOLO 1000 is a trademark of European Silicon Structures workstation for a period of about 4 weeks. A schematic capture based on a graphics editor was used for most of the random-logic sections of the circuit, but the state machine was written in a descriptive language supported by the system. Such a language is very powerful for repetitive structures. The graphics editor allows hierarchical design; parts can be grouped and structured in increasingly more complex functional blocks. The output produced by the schematic editor is then translated into the same functional language. The SOLO 1000 system is capable of designing digital logic and is orientated towards design engineers without specific experience in VLSI design. It allows a moderate degree of manual intervention: buffer dimensions can be specified, gates can be grouped to improve timing characteristics and critical signals can be routed first. The basic algorithm for placing transistors in the array follows a rather simple unidimensional expansion; this can be improved by proper grouping of parts. The system also includes optimization for removal of unused parts.

Simulation was done with an interactive program. The system offers a switch-level simulator. It takes its input from: the internal design language describing the circuit, the electrical process rules, the output from the layout program that has previously placed and routed the circuit and the manufacturer's process characteristics.

The simultation program was used to generate (up to 4096) test patterns to be used after fabrication by the testing machine. An unlimited number of test vectors is available during the design phase. If no manual adjustment is desired, placement and routing is provided automatically by the system. The peripheral cells placements were chosen by the designers to give the optimum pin-out. The designs were extensively simulated on the workstation.

6. Results

The controller chip consists of about 1600 gate-equivalents, giving a 4.3 mm by 5.5 mm die. The data chip has roughly half the number of gates but about the same dimensions, being mostly pad bound. The two chips were tested at the labs where they were designed. The Rutherford Appleton Laboratory used a Hewlett-Packard 9816 based system, controlling an HP8180 generator with two HP8181 slaves, two HP8182 logic state analyzers and a HP15414 tristate unit. At Ecole Polytechnique the data chips were checked on a µAnalyst 2000 Mainframe System from Northwest Instrument. During the tests, special test socket adapters had to be built to conform to the high load requirements of the I/O ports.

Both systems reported results close to or better than the original manufacturer's predictions, although a rather large dispersion was found between controller chips.

The fault coverage was 95% (for single stuck at 0 and stuck at 1 faults), the missing 5% being mostly due to the extra test logic in the chip. All controller chips were properly working at 25 MHz, but only about half of them at 33 MHz. All data chips were better than the original manufacturer worst case prediction. Test were only performed at nomimal supply voltage and at 25°C ambient temperature.

A picture of the Fastbus board (Fig. 4) shows the chip on a wire-wrap prototype where they have been tested. The chips out of the first prototype series have been shown to be fully working, meeting the required design specifications.

7. Cost Analysis

For a successful introduction of this technology in the high energy physics environment, the economical aspects of such a project must be carefully analyzed.

Learning stage: the commercial packages available today for digital VLSI design of standard cells and gate arrays are approaching a state where 2-4 weeks of training are sufficient for a person with a basic background in CAD techniques. This is a one-time investment and if a new system has to be used later, the learning time will be substantially reduced.

Cost of necessary tools: this aspect must be considered in conjuction with the laboratory policy. The design of just one single project certainly does not justify the purchase of a dedicated workstation plus the necessary software. Workstations can be rented from manufacturers; this solution is much more economical for a limited number of designs.

Purely component cost: fabrication cost for typically 100 prototype integrated circuits of moderate complexity (less than 5000 equivalent gates) is today in the 20K$ range. Considering a typical integration factor of 10, as in our case, the cost of the integrated solution is about 10 times higher than the corresponding discrete version. The production cost of a subsequent series of about one thousand pieces can be negotiated with the manufacturers for a price reduction of about one order of magnitude. Therefore, pure component cost can be considered competitive for production quantities of the order of several thousand parts.

Several manufacturers are introducing direct silicon writing technologies, using electron beam machines instead of traditional optical masks. This technique will hopefully shorten turnaround time considerably (2-4 weeks are to be expected) and also make prototype production series more affordable.

System cost: the integration of large parts of a system in silicon offers several advantages. Board size is reduced, or board area can be saved for expansion of other functions, and usually power consumption is also reduced. Speed can be improved even using relatively non-sophisticated technologies. These factors are difficult to quantify accurately and are normally implementation dependent. Reliability is increased through minimization of interconnection. Testing time is reduced because chips can be checked immediately after fabrication. The extensive simulation of the parts during the design phase also considerably reduces the debugging phase: relatively complex designs can be made fully working in a few iterations.

Acknowledgements

We would like to thank the engineers of European Silicon Structures in Paris and Bracknell for the help given during the execution of this project.

References

1. E. M. Rimmer, *The fundamental of Fastbus*,
   Interfaces in Computing, Vol. 3 n. 1 pag. 1
2. S. R. Amendolia et al., *ALEPH Data Acquisition System, Hardware Functional Specifications*,
   ALEPH Datacq Note 85-21, Nov. 85
3. A. Marchioro et al., *The ALEPH Event Builder*,
   IEEE Transactions on Nuclear Science, Vol. NS-34, No. 1, Feb 1987, pag 133

Figure captions

Fig. 1 Block diagram of the Aleph Event Builder, only the data path is shown

Fig. 2 Block diagram for the DMA controller, signal names are explained below:

Req68020 : bus request from the 68020 to the arbitration logic

ReqDMA : bus request from the DMA controller to the arbitration logic

Gnt68020: bus access grant ot the 68020

GntDMA : bus access grant to the DMA controller

Instr: instruction field to the DMA controller

CLOCK: main clock to the DMA controller (20-33MHz)

GO: go bit from the coprocessor sequencer to the DMA controller, it starts all DMA transfers

MEM_ERR: memory error indication to the DMA controller

SS_ERR: Fastbus SS error to the DMA controller

PAR_ERR: Parity error on the Fastbus side

WAIT: Fastbus wait signal

DS#DK: DS unequal to DK on Fastbus

The instruction decoder includes the logic to drive all the other parts of the circuit.

Fig. 3 State diagram for the operation of the controller state machine; detailed operation on the transitions is explained in the paper.

Fig. 4 Photograph of the Aleph Event Builder CPU board. The five large VLSI chips are visible in the middle right part of the board, next to the connector for the Event Memory.

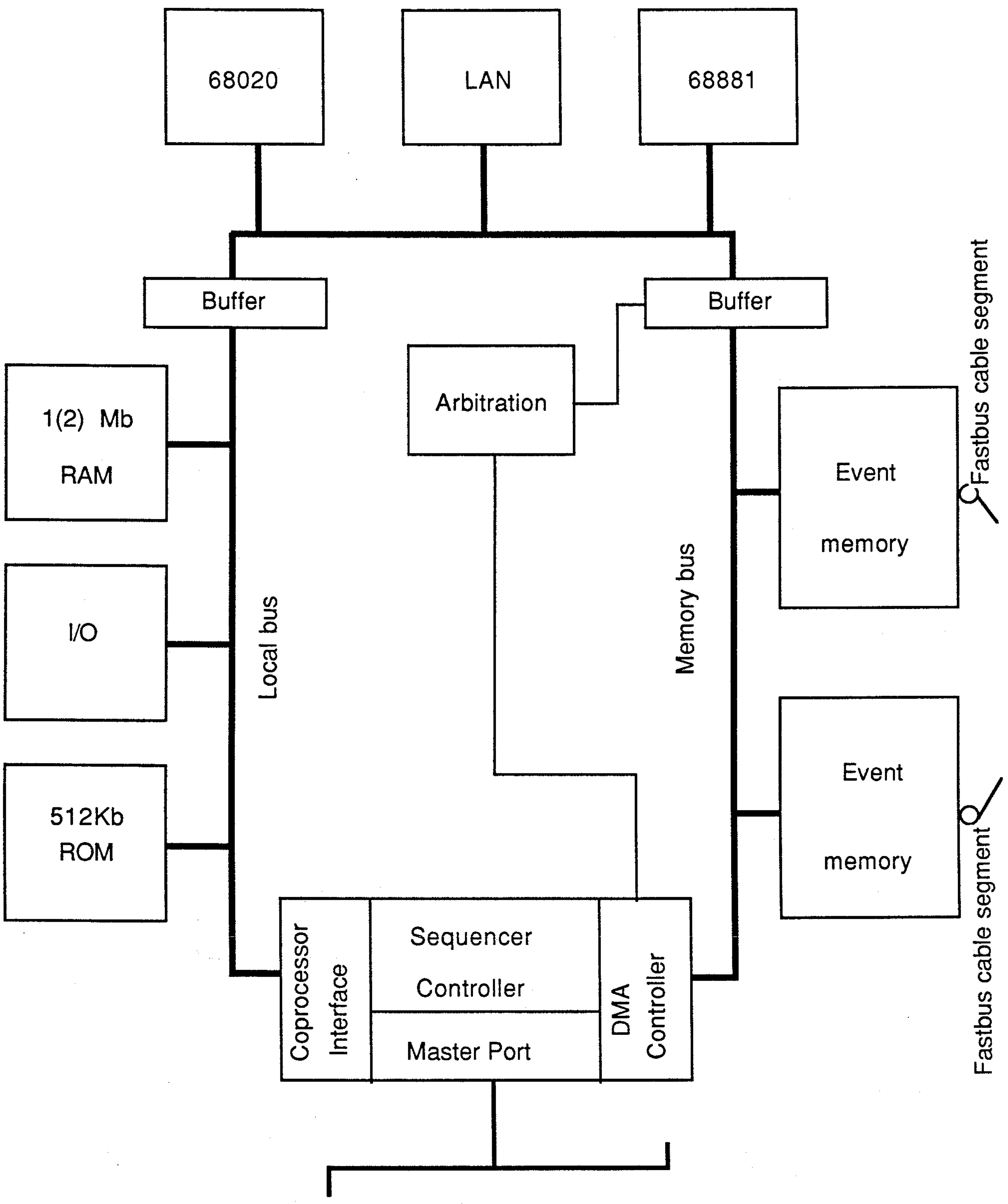
68020
LAN
68881
Buffer
Buffer
Arbitration
1(2) Mb RAM
I/O
512Kb ROM
Local bus
Memory bus
Event memory
Event memory
Fastbus cable segment
Fastbus cable segment
Coprocessor Interface
Sequencer Controller
Master Port
DMA Controller
Fastbus crate segment

Fig. 1

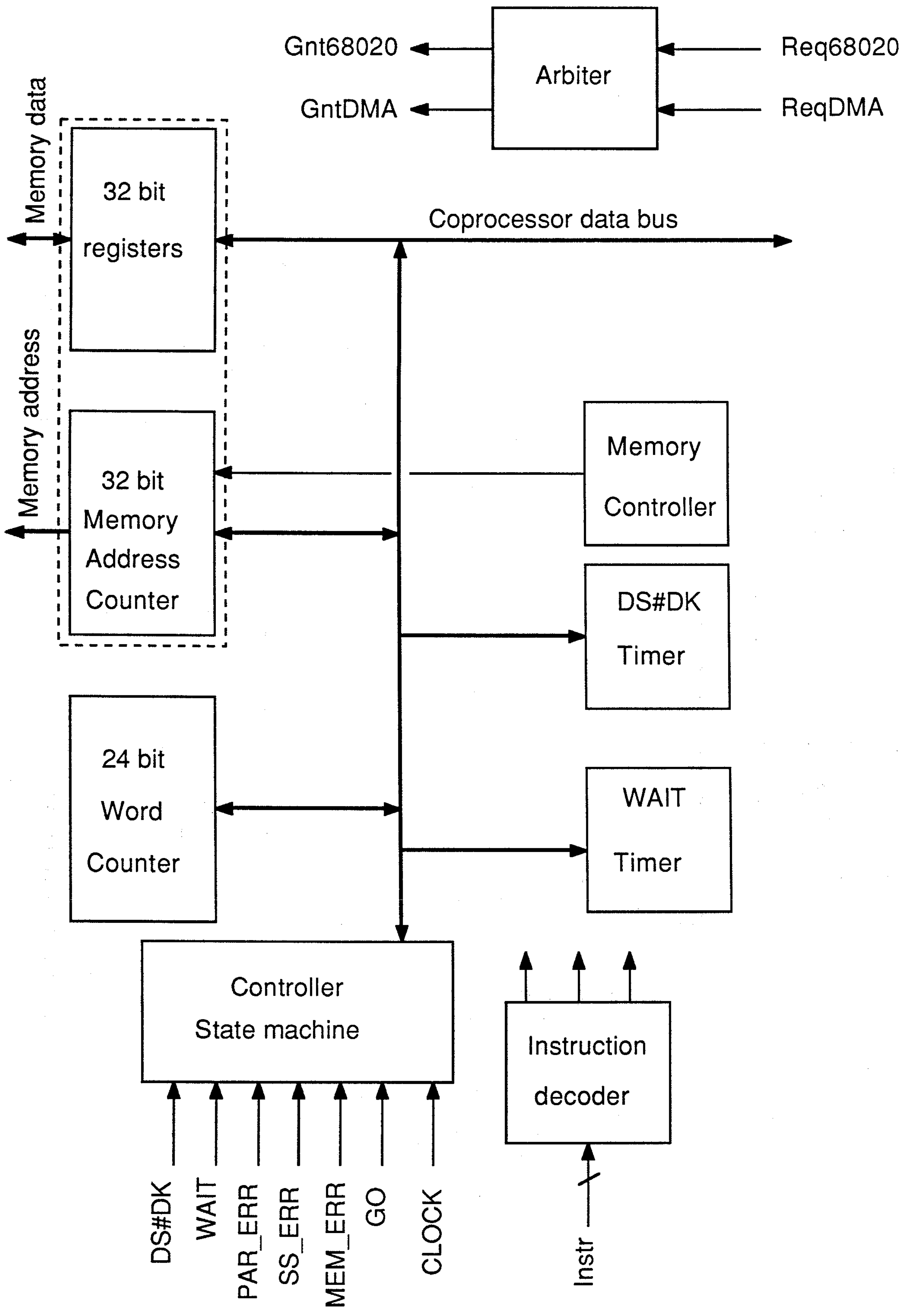
Gnt68020
GntDMA
Arbiter
Req68020
ReqDMA
Memory data
32 bit
registers
Coprocessor data bus
Memory address
32 bit
Memory
Address
Counter
Memory
Controller
DS#DK
Timer
24 bit
Word
Counter
WAIT
Timer
Controller
State machine
Instruction
decoder
DS#DK
WAIT
PAR_ERR
SS_ERR
MEM_ERR
GO
CLOCK
Instr

Fig. 2

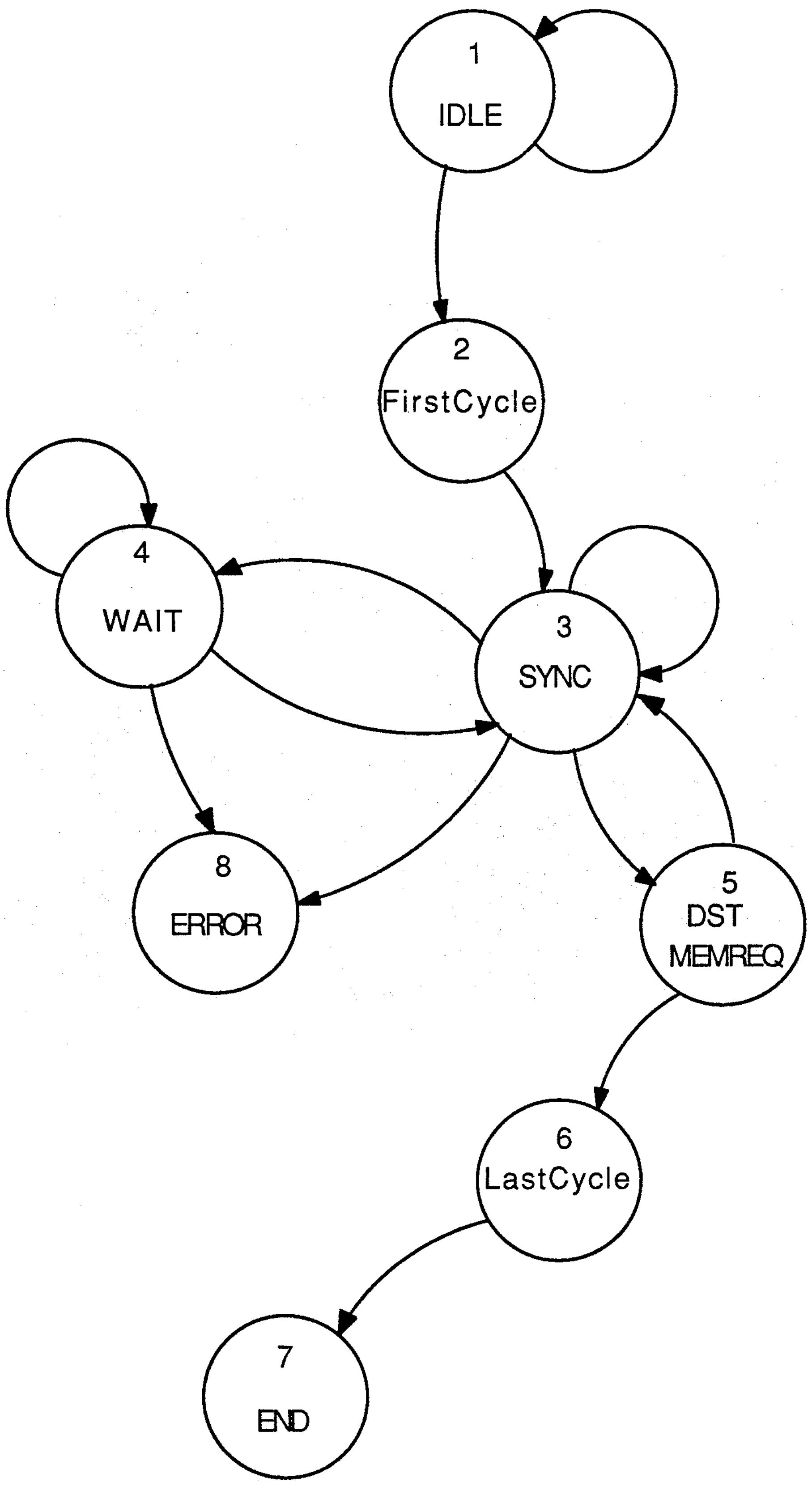
1
IDLE
2
FirstCycle
3
SYNC
4
WAIT
5
DST
MEMREQ
6
LastCycle
7
END
8
ERROR

Fig. 3

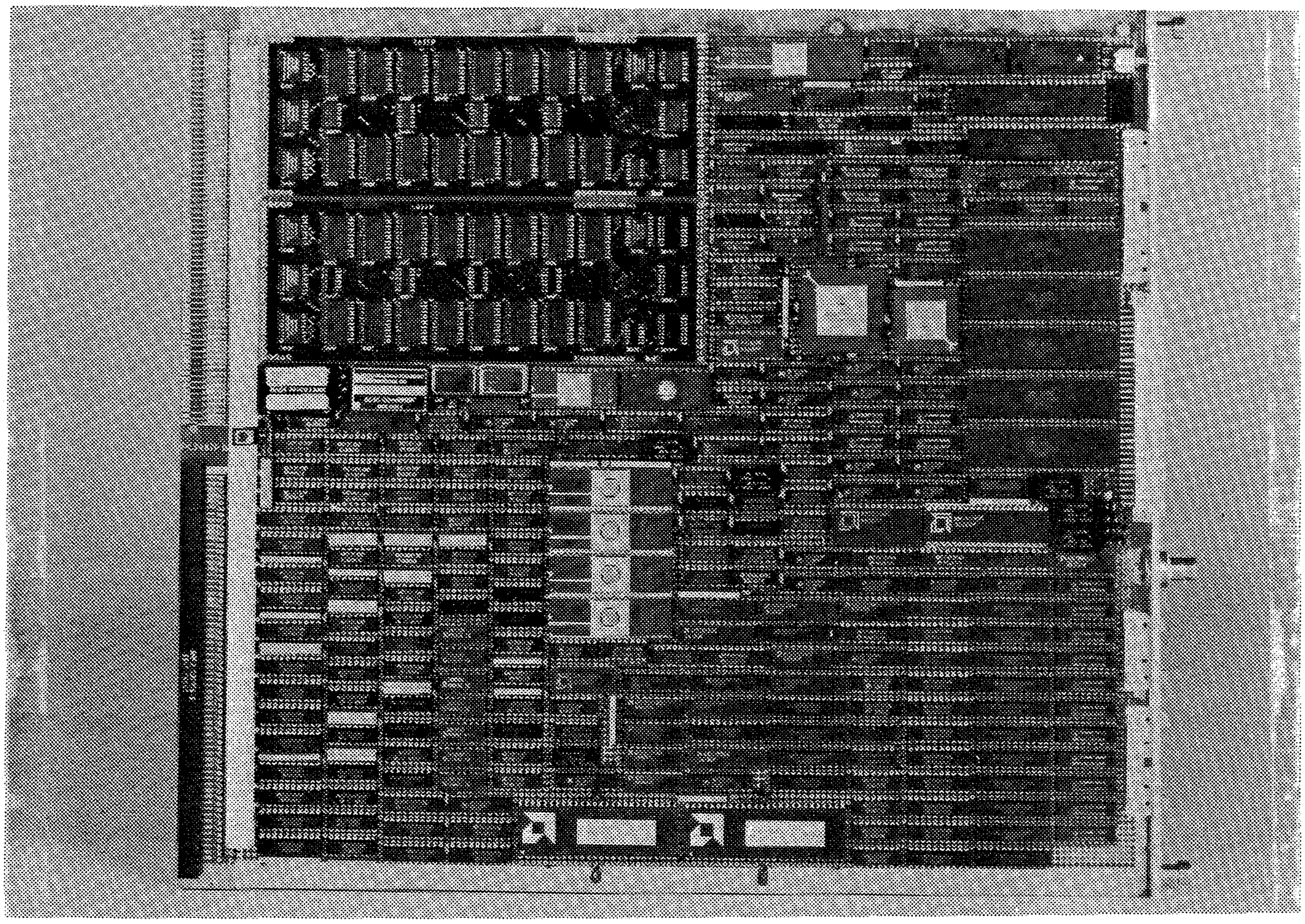

FIG. 4