MODULARIZATION OF LARGE COMPUTING SYSTEMS

by

Sin-Kin Edwin Law

Memorandum No. UCB/ERL M80/27

20 June 1980

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Modularization of Large Computing Systems

Sin-Kin Edwin Law

Computer Science Division
Department of Electrical Engineering and Computer SCiences
University of California, Berkeley
Berkeley, California 94720

1. INTRODUCTION

With the growing complexity of very large scale integrated circuits approaching a million devices on a single chip by mid-1980's, some computer aided design tools will be necessary to lessen design time. This project proposes a set of modules from which the central processing unit of a large computer can be assembled. The set of modules is parametrizable so that they are suitable for silicon compilation. Two large computers, the S-1 [Wi80], developed at the Lawrence Livermore Laboratory, and the VAX 11/780 [BeMu78], manufactured by Digital Equipment Corporation, were analyzed.

2. APPROACH

Both top-down and bottom-up approaches were used to analyze the S-1 and the VAX. The S-1 was designed with the help of SCALD [McWi78a] [McWi78b] (Structured Computer Aided Logic Design). The SCALD language allows the processor to be designed in a hierarchically structured manner. Top down analysis of the S-1 is very easy and the result of the analysis is presented in section 3. Bottom up approach has been equally easy because the output of SCALD includes useful statistics about the bottom level. These statistics are given in section 4. Analysis of the VAX has been much more difficult. I have gone through several documents including the CPU technical description [DEC79], the TB/CACHE/SBI technical description [DEC78a], and the blueprints of the VAX CPU boards. Top down analysis has to be done by following through the block diagrams level by level down to the prints. The results are presented in section 5. The bottom-up approach is even more difficult. All the bottom level statistics are gathered by going through the VAX prints page by page. These are described in section 6.

3. TOP DOWN ANALYSIS OF S-1

3.1. SCALD

SCALD is a software package that was developed to design the S-1 Uniprocessor. SCALD is hierarchical in nature. Starting from a very high level description of the S-1, each higher level module gets expanded by calling lower level modules until the component level is reached. SCALD consists of several pieces of software including a graphics editor [He72], a macro expander, and a router. The graphics editor allows drawings to be input which is then expanded by the macro expander. The output of the router is the physical layout of the various integrated circuits packages.

3.2. S-1 Architecture

S-1 is a pipelined processor consisting of 5721 ECL 10000 series integrated circuits. Instruction preparation, operand preparation, and instruction execution are pipelined and each stage is controlled by a separate microsequencer. The S-1 has two caches: one for instruction and one for data. It is a 36-bit machine with 2^{30} virtual address space.

The processor consists of five main sections: the Instruction Box, the Execution Box, the Clock Generator, the Front End Interface, and a Dummy Box. This is in fact the highest level macro in SCALD. This macro is shown in figure 1.

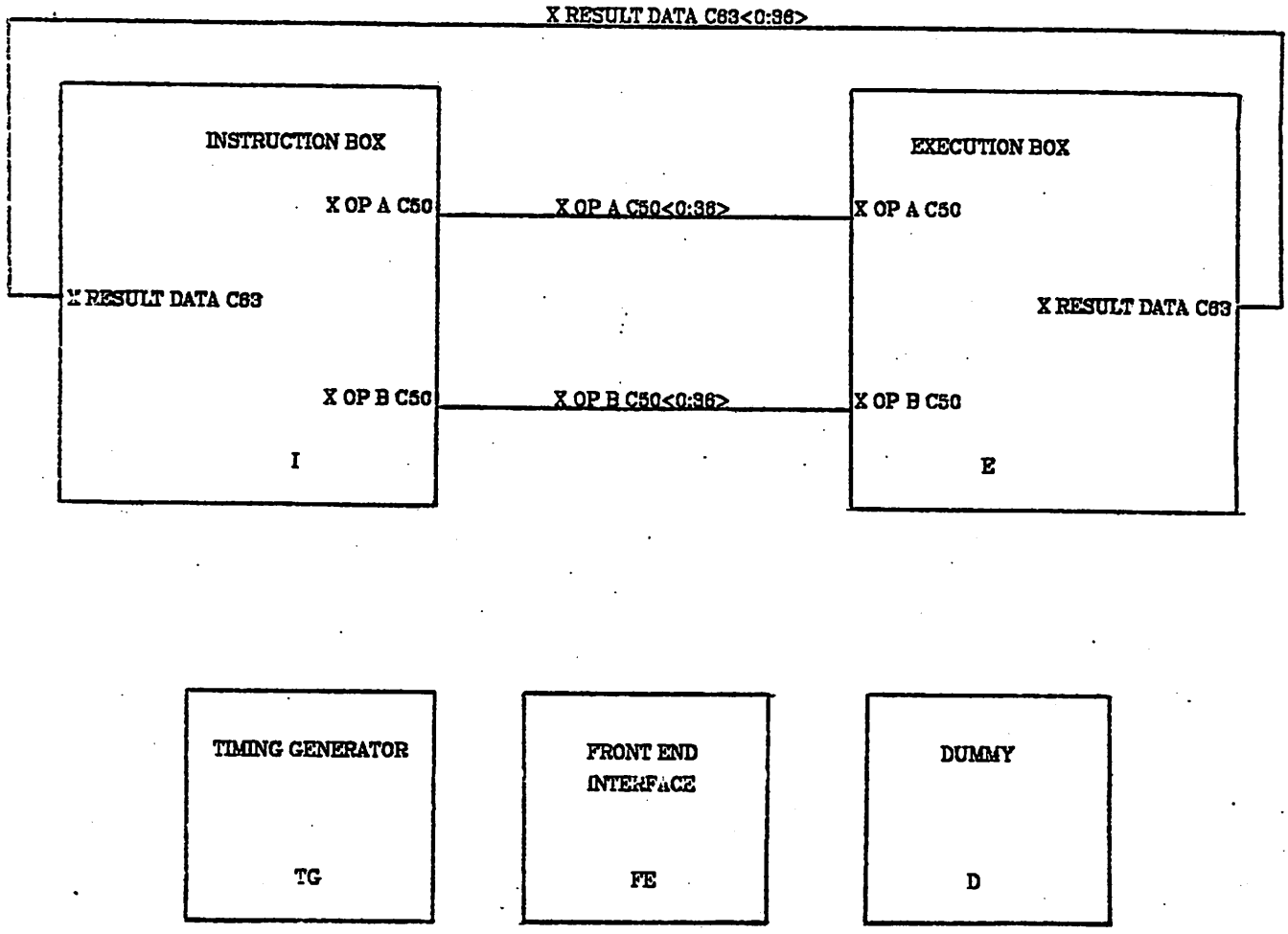


Figure 1 - S-1 CPU Macro

3.2.1. Instruction Box

The instruction box macro is shown in figure 2. The instruction box controls the fetching of instruction and operands, the interaction with the cross bar switch to read and write main memory into and from the cache, and all I/O operations. It has two four-way set associative caches, one for instruction and one for data; a least recently used replacement algorithm is used. It has three 256 words by 36-bit register files. One is the index register file and the other two are in the Data Cache and Register File Macro.

The Instruction Address Arithmetic, Instruction Address Translation, Instruction Cache, and Instruction Buffer and Decode macros all have to do with prefetching instructions. The Index Register File, Data Address Arithmetic, Data Address Translation, and Data Cache and Register File macros are for the preparation of operands. Memory interface reads and writes to main memory through the cross bar switch. The data paths to main memory supports single error correction and double error detection.

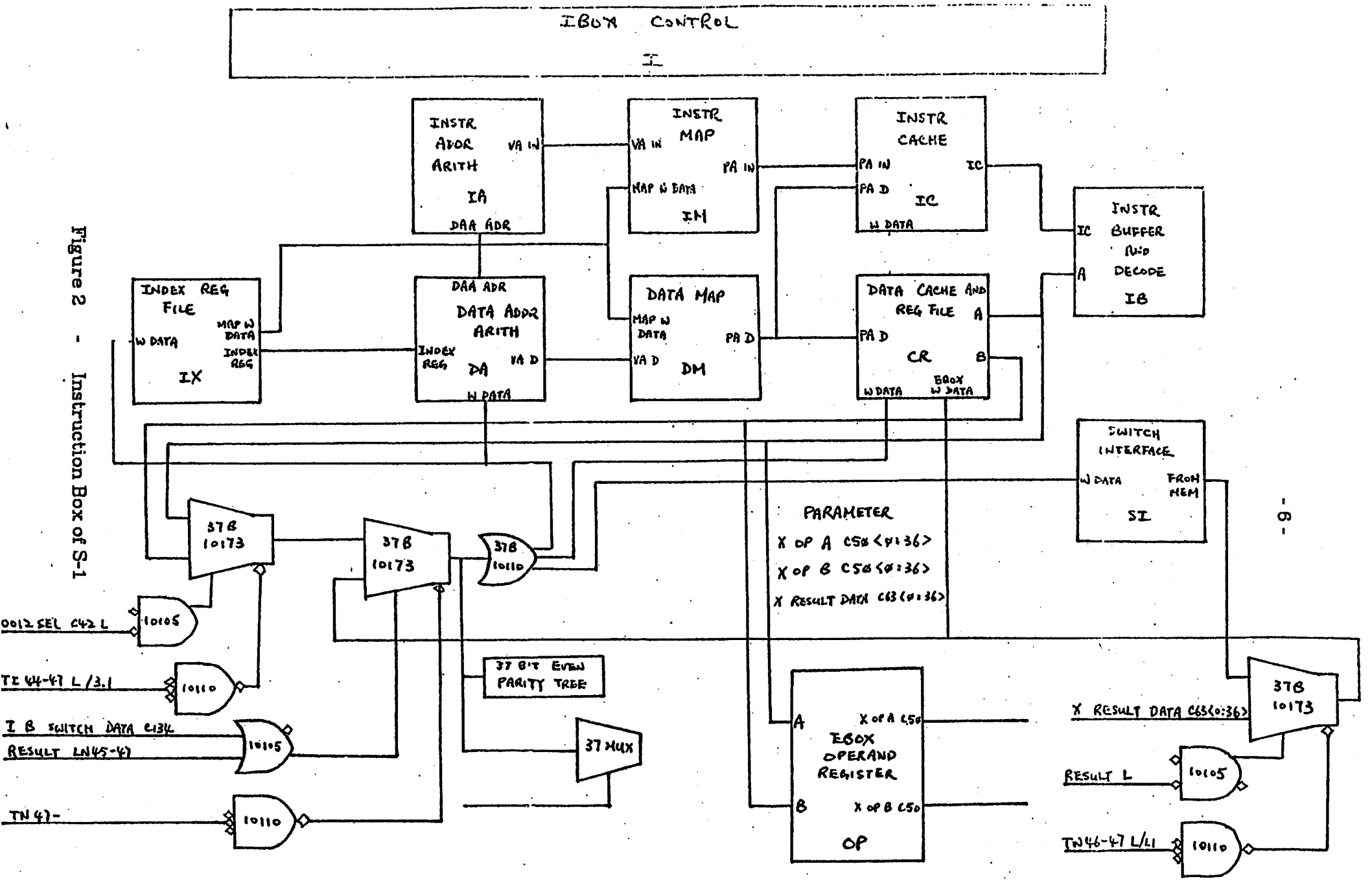


Figure 2 - Instruction Box of S-1

3.2.2. Execution Box

The execution box macro is shown in figure 3. The execution box performs variable precision arithmetic and logical operations. It consists of the EBOX Register File, EBOX Arithmetic Logic Unit, Exponent Box and EBOX Control macros. The operands prepared by the instruction box is passed to the execution box and stored in the EBOX Register File. It is a dual port register file so that either two reads or two writes can be done in each microcycle. It can also do shifting, sign-extension and can deliver zero operands.

3.2.3. Timing Generator

It generates an eight-phase clock used in the processor. It consists of an eight bit circular shift register which is initialized to the sequence 01111111, and it just circulates the zero.

3.2.4. Front End Interface

It allows the front end processor to control the S-1 processor. The operator at the console can start and stop the processor. Internal states of the processor can also be read out for diagnotics purposes.

3.2.5. Dummy

It has drivers and receivers for all signals that go outside the S-1 processor to help the design system. The logic in this macro is never built. It is just given to allow the design system to check the SCALD runs.

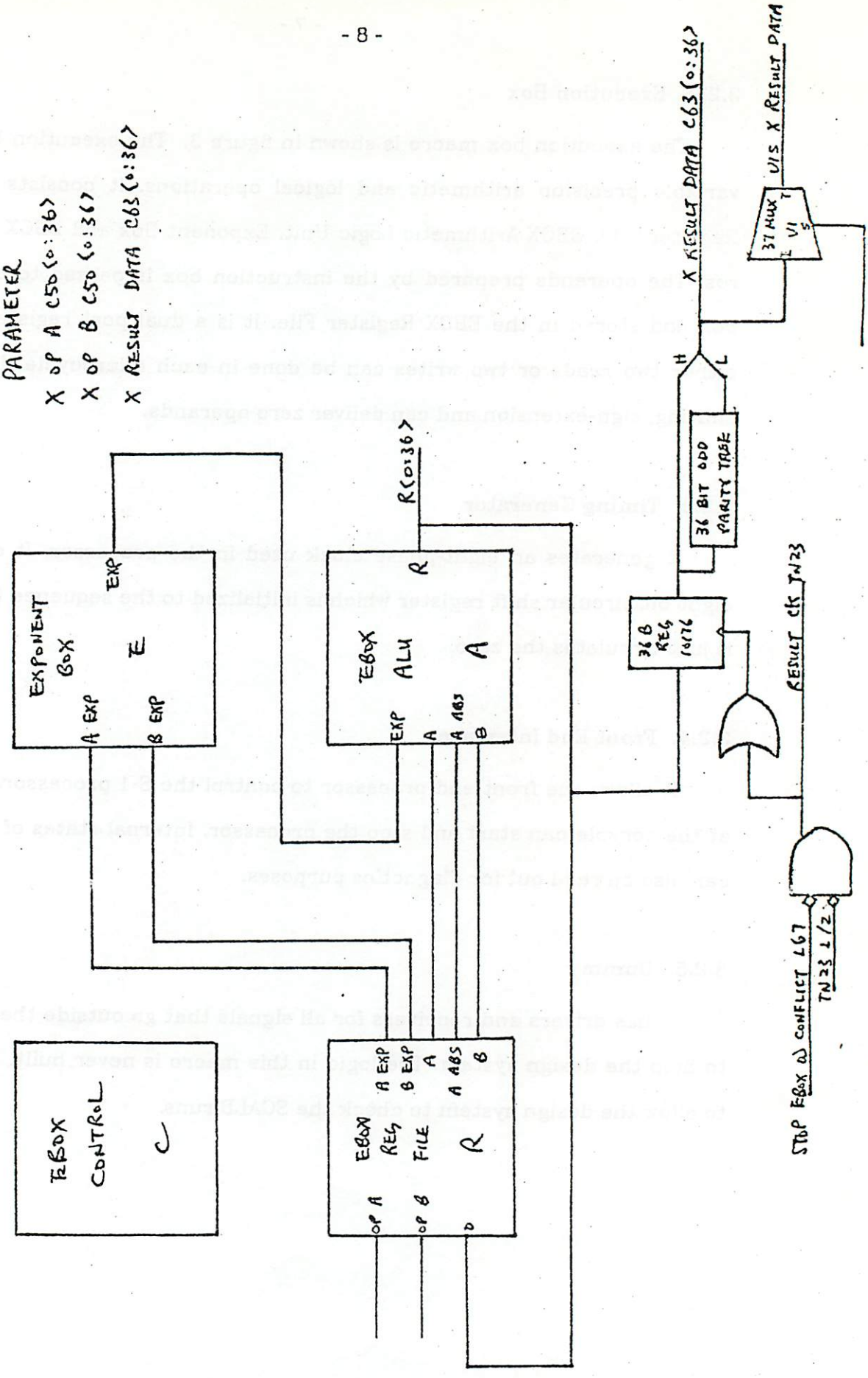


Figure 3 - Execution Box of S-1

3.3. Macros in S-1

There are a total of 256 macros definitions in S-1. These macros are expanded, starting from a top-level macro and continuing downward until no macro remains which has a definition (i.e. all remaining macros are available devices). The following is a list of macros that have been called at least twice by a higher level macro and consist of non-homogeneous integrated circuits.

Macros	Called by	# Calls	# Chips
37B x 256W IBOX Register	Index Register File, Data Cache and Register File	3	144
Value Saver	Data Address Arithmetic	2	28
Address Map	Instruction map, Data Map	2	314
Map Cache Module	Address map	4	112
Instruction Cache Module	Instruction Cache	4	308
Cache and Register File Control	Data Cache and Register File	2	30
Data Cache Module	Data Cache and Register File	4	312
Cache LRU Control	Instruction Cache, Data Cache	2	62
20 bit Decode RAM	Instruction Decode	2	146
Register Lock	Instruction Buffer and Decode, IRX Registers, P Sequencer	4	12
Operand Descriptor Decode	P Sequencer	2	26
36 bit Translate	EBOX Register File	2	44
9 bit Merge	36 bit MUX Merger	4	48
Total			1586

Table 1

These 13 macros represent 1586 integrated circuits which is only 28% of the S-1 CPU. This means that only about a quarter of the machine can be assembled given these 13 high level modules. Each call to these modules is identical. The only parameters are the connections within the caller macro. We should also notice that 5 out of the 13 macros are associated with caches and cache control.

4. BOTTOM UP ANALYSIS OF S-1

The S-1 Mark I Uniprocessor was built with a total of 5721 ECL 10K series integrated circuits. 50 different parts were used. The following table is based on the output of SCALD. The first and second column give the part number and the part name. The third column gives the the number of that particular integrated circuits being used in the S-1 CPU. The fourth column gives the number of transistors per chip. The data in column 4 is gathered primarily by counting the number of transistors shown in the schematic of the Fairchild ECL Data Book [Fa77]. For LSI chips like the RAM's, an estimation was made. I made the assumption that there are four transistors per RAM cell. For comparison purpose, I count only the decoding logic associated with the cells. Transistors associated with the sense amplifiers and read/write logic are not included in my transistor count in column 4. The fifth column gives the total number of transistors for that particular part used in the S-1. This is obtained by multiplying column 3 and column 4.

4.1. Part List of S-1

Type	Name	Quantity	Tr/Chip	Total # Tr
MB7042	RAM 256 x 1	301	1568	471968
10016	Counter	22	243	5346
10101	Quad OR/NOR	443	28	12404
10104A	Quad 2-input AND	39	25	975
10104B	Quad 2-input AND	13	25	325
10105A	Triple 2-3-2 input OR/NOR	197	21	4137
10105B	Triple 2-3-2 input OR/NOR	76	21	1596
10107	Triple Exclusive OR/NOR	13	36	468
10109A	Dual 4-5 input OR/NOR	51	19	969
10109B	Dual 4-5 input OR/NOR	83	19	1577
10110	Dual 3-input/ 3-output OR	295	18	5310
10111	Dual 3-input/ 3-output NOR	144	18	2592
10113	Quad Exclusive OR	166	72	11952
10115	Quad Line Receiver	49	20	980
10117	Dual 2-Wide OR-AND/ OR-AND-INVERT	32	27	864
10118	Dual 2-Wide 3-input OR-AND	1	24	24
10119	4-Wide 4-3-3-3 input OR-AND	2	25	50
10120	4-Wide OR-AND/ OR-AND-INVERT	22	28	616

10130	Dual D Latch	37	32	1184
10131	Dual D Flip-Flop	60	32	1920
10132	Dual MUX w/Latch (Common Reset)	63	78	4914
10134	Dual MUX w/Latch (Separate Select)	5	82	410
10136	Universal Binary Counter	6	308	1848
10141	4-bit Shift Register	44	212	9328
10145A	64-bit RAM (16 x 4)*	179	524	93796
10153	Quad Latch (Negative Clock)	3	96	288
10158	Quad 2-input MUX (Non-inverting Output)	115	68	7820
10159	Quad 2-input MUX (Inverting Output)	25	68	1700
10160	12-bit Parity Generator/ Checker	82	100	8200
10161	Binary to 1-8 Line Decoder (Low)	12	53	636
10162	Binary to 1-8 Line Decoder (High)	10	53	530
10163	Error Detection/ Correction Circuit	6	132	2112
10164	8-line MUX	508	76	38608
10165	Priority Encoder	12	205	2460
10170	9 + 2 bit Parity Generator/ Checker	21	63	1323
10171	Dual 4-line Decoder (Low)	8	70	560
10173	Quad 2-input MUX w/Latch	298	108	32184
10174	Dual 4-to-1 MUX	332	51	16932
10175	Quint Latch	95	88	8360
10176	Hex D Flip-Flop	420	122	51240
10179	Carry Look Ahead	11	73	803
10180	Dual 2-bit Adder/ Subtractor	21	71	1491
10181	4-bit Arithmetic Logic Unit	138	382	52716
10186	Hex D Flip-Flop w/Common Reset	9	107	963
10195	Hex Inverter/ Buffer	12	24	288
10197	Hex AND	43	36	1548
10211	High Speed Dual 3-input 3-output NOR	7	18	126
10231	High Speed Dual D Flip-Flop	2	118	236
10405	128 x 1 RAM	12	796	9552
2110-1	1024 x 1 RAM	1176	4344	5108544
	Total	5721		5984773

Table 2

4.2. Functional Blocks of S-1

The 50 different IC's can be classified into 19 different functional blocks.

Function	Chip	Quantity	% S-1	Total # Tr	% S-1
AND					
	10104A	39		975	
	10104B	13		325	
	10197	43		1548	

		95	1.66	2848	0.05
OR					
	10101	443		12404	
	10105A	197		4137	
	10105B	76		1596	
	10109A	51		969	
	10109B	83		1577	
	10110	295		5310	
	10111	144		2592	
	10211	7		126	
		1296	22.65	28711	0.48
Exclusive OR					
	10107	13		468	
	10113	166		11952	
		179	3.13	12420	0.21
Inverter					
	10195	12	0.21	288	0.005
OR-AND/OR-AND-INVERT					
	10117	32		864	
	10118	1		24	
	10119	2		50	
	10120	22		616	
		57	1.00	1554	0.03
Line Receivers					
	10115	49	0.86	980	0.02
Flip-Flop (Register)					
	10131	60		1920	
	10176	420		51240	
	10186	9		963	
	10231	2		236	
		491	8.58	54359	0.91
Latch					
	10130	37		1184	
	10153	3		288	
	10175	95		8360	

		135	2.36	9832	0.16
Decoder					
	10161	12		636	
	10162	10		530	
	10171	8		560	
		30	0.52	1726	0.03
Multiplexer					
	10132	63		4914	
	10134	5		410	
	10158	115		7820	
	10159	25		1700	
	10164	508		38608	
	10173	298		32184	
	10174	332		16932	
		1346	23.53	102568	1.71
Adder					
	10180	21	0.37	1491	0.02
Carry Look-ahead					
	10179	11	0.19	803	0.01
Parity Generator					
	10160	82		8200	
	10170	21		1323	
		103	1.80	9523	0.16
Error Detection/Correction					
	10163	6	0.10	2112	0.04
Priority Encoder					
	10165	12	0.21	2460	0.04
Shift Register					
	10141	44	0.77	9328	0.16
Counter					
	10016	22		5348	
	10136	6		1848	
		28	0.49	7194	0.12

Arithmetic Logic Unit					
	10181	138	2.41	52716	0.88
Register File/RAM					
	10145A	179		93796	
	MB7042	301		1354500	
	10405	12		4800	
	2110-1	1176		5292000	
		1668	29.16	5683860	94.97

Table 3

4.2.1. Register Files/ Memories in S-1

The random access memories in the S-1 are used in various functions. They can be register files. They can be organized as queues or stacks. They are also used in caches and control stores.

The S-1 contains 5 register files : an index register file, 2 user register files, an EBOX register file, and a T register file. The T register file is used by the microcode to calculate operand addresses. The size of the various register files are given below :

Register files/Memories	Size	Chips	Quantity	Total # Tr
Register files				
Index	37B x 256W	MB7042	37	166500
User	37B x 256W	MB7042	74	333000
EBOX	36B x 32W	10145A	36	18864
T	36B x 16W	10145A	9	4716
Total			156	523080
Memory				
		MB7042	190	297920
		10145A	134	70216
		10405	12	9552
		2110-1	1176	5108544
Total			1512	5486232

Table 4

4.3. Ten most frequently used functional blocks by chips

Parts	number	% S-1
Register file/Memory	1668	29.16
Multiplexer	1346	23.53
OR/NOR	1296	22.65
Flip-Flop	491	8.58
XOR	179	3.13
ALU	138	2.41
Latches	135	2.36
Parity Generator	103	1.80
AND/NAND	95	1.66
AND-OR-INVERT	57	1.00
Total	5508	96.28

4.4. Ten most frequently used functional blocks by transistors

Parts	Transistors	% S-1
Register files/Memory	5683860	94.97
Multiplexer	102568	1.71
Flip-Flop	54359	0.91
ALU	52716	0.88
OR/NOR	28711	0.48
XOR	12420	0.21
Latch	9832	0.16
Parity Generator	9523	0.16
Shift Register	9328	0.16
Counter	7194	0.12
Total	5970511	99.76

4.5. Ten least frequently used functional blocks by chips

Parts	number	% S-1
Error Detection/Correction	6	0.10
Carry Lookahead Generator	11	0.19
Inverters	12	0.21
Priority Encoder	12	0.21
Full Adder	21	0.37
Counter	28	0.49
Decoder	30	0.52
Shift Register	44	0.77
Driver/Receiver/Transceiver	49	0.86
AND-OR-INVERT	57	1.00
Total	270	4.72

4.6. Ten least frequently used functional blocks by transistors

Function	Transistors	% S-1
Inverter	288	0.005
Carry Lookahead Generator	803	0.01
Driver/Receiver/Transceiver	980	0.01
Full Adder	1491	0.02
AND-OR-INVERT	1554	0.03
Decoder	1726	0.03
Error Detection/Correction	2112	0.04
Priority Encoder	2460	0.04
AND/NAND	2848	0.05
Counter	7194	0.12
Total	21456	0.35

5. TOP DOWN ANALYSIS OF VAX 11/780

The VAX 11/780 Central Processing Unit has a total integrated circuits count of 2492. The floating point accelerator is an optional unit and is not included in the statistics. The block diagram of the CPU in shown in figure 4.

To make the comparison between the S-1 and the VAX more meaningful, I made a similar system partitioning for the VAX as that for the S-1. Therefore the VAX CPU consist of the Instruction Box, the Execution Box, the Clock Generator, and the Front End Interface.

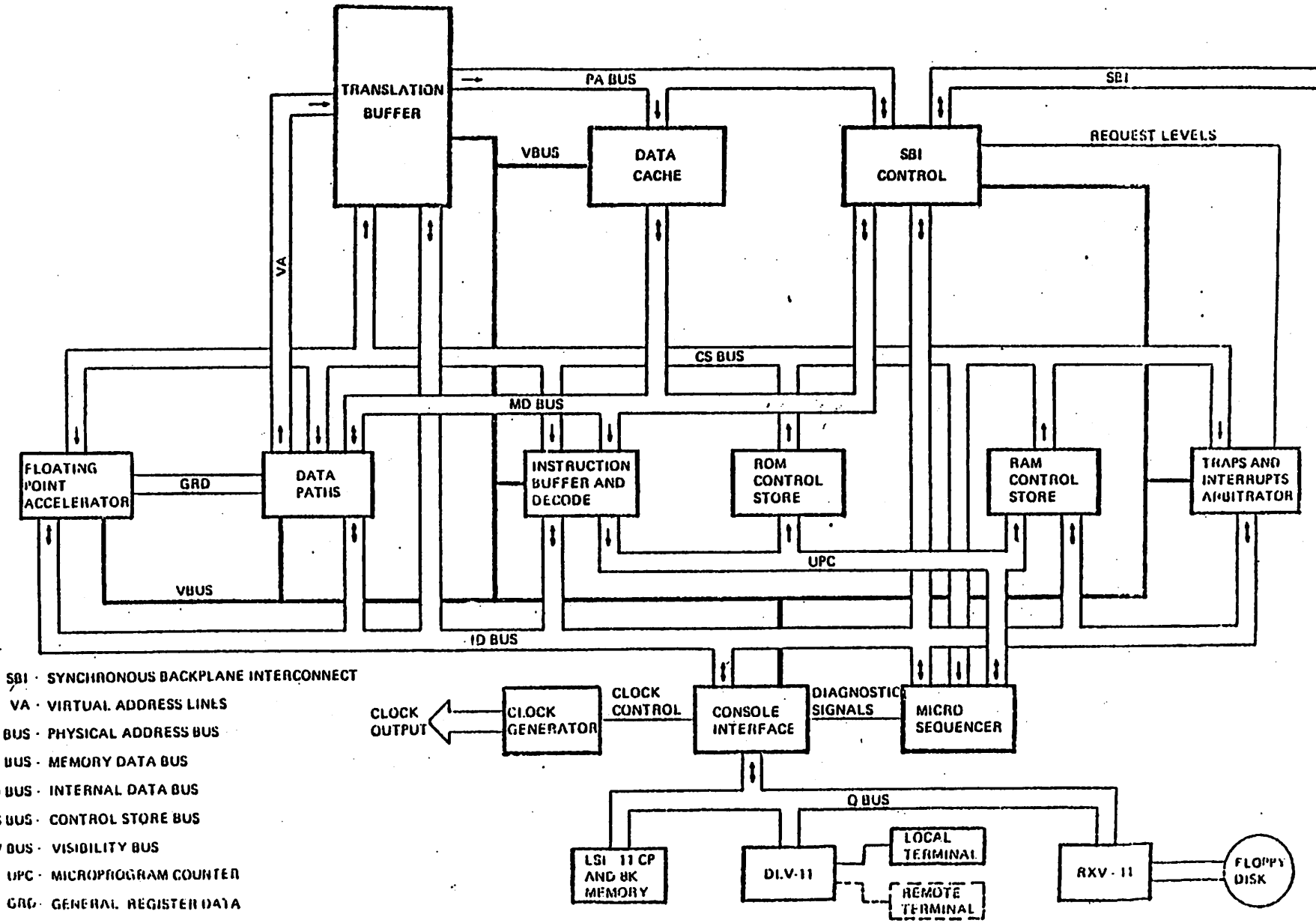


Figure 4 - VAX CPU

5.1. Instruction Box

Refering to figure 4, the Instruction Box consists of six modules:

1. Instruction Buffer and Decode - It handles instruction prefetch and instruction decoding.
2. Translation Buffer - It contains a cache of page table entries which is used to convert virtual address to physical address.
3. Data Cache - It is two-way set associative using random replacement algorithm.
4. Synchronous Backplane Interconnect (SBI) Control - It handles all main memory reads and writes and all I/O operations.
5. Traps Interrupts Arbitrator - The VAX has 32 interrupt priority level. Exception handling has the highest priority.
6. Microsequencer, the PROM Control Store and the Writable Control Store - They coordinate all activities within the CPU.

5.2. Execution Box

The Execution Box is shown in figure 5. The Execution Box is divided into four independent sections and each section can operate in parallel with the other three sections:

1. Address section - It is responsible for calculating operand addresses and updating the program counter.
2. Arithmetic section - It contains the arithmetic logic unit and three 16 by 32 bit register files.
3. Data section - It holds the operands and performs shifting, byte alignment and unpacking floating point data types.

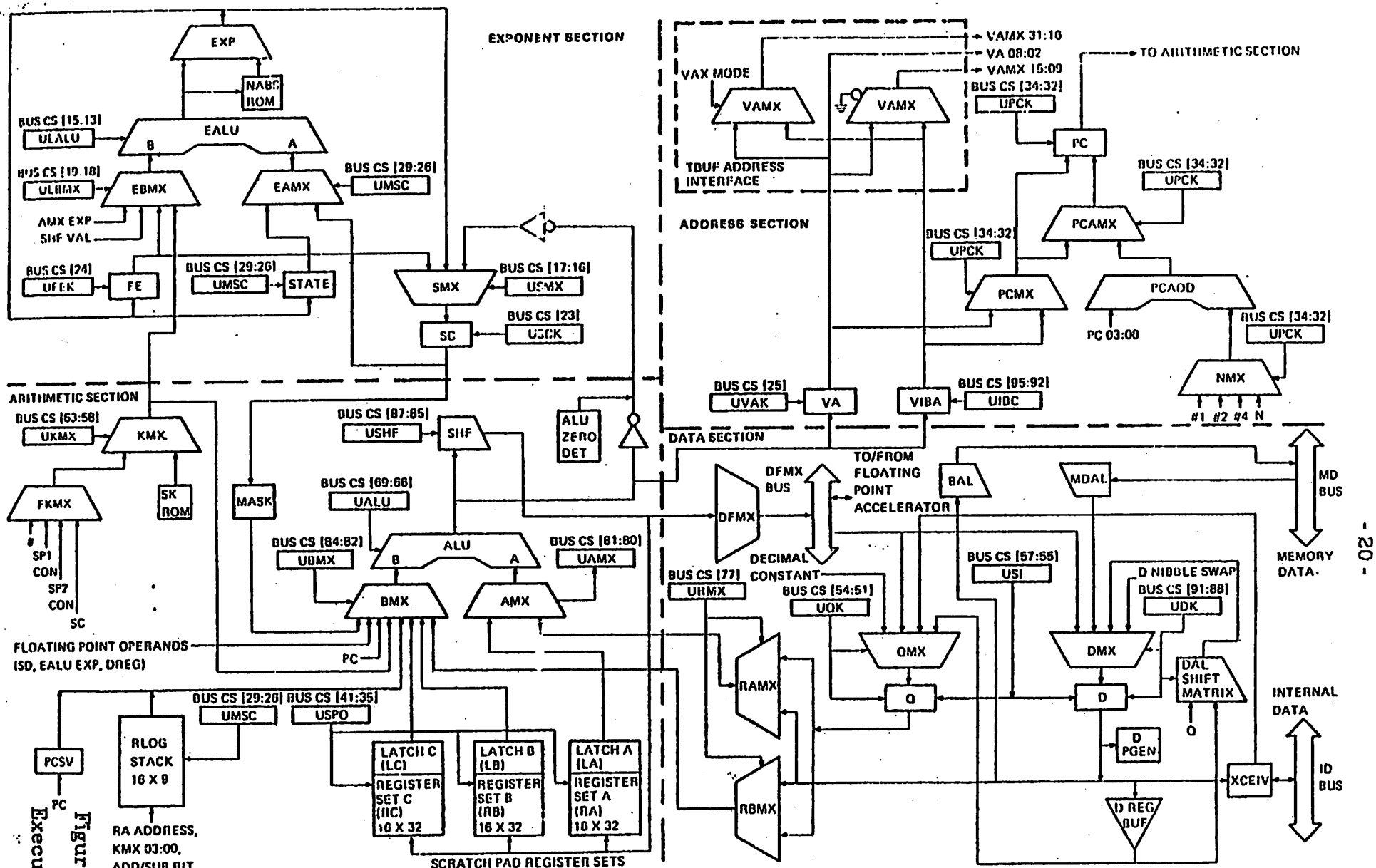


Figure 5 Execution Box

4. Exponent section - It contains an exponent ALU to process the exponent value of floating point numbers. It is performed in parallel with the fraction processing performed in the Arithmetic and Data sections.

5.3. Clock Generator

VAX has three clocks: the processor clock, the time of year clock and the interval time clock. The processor clock provides a 200 ns cycle with four 50 ns time states per cycle. There is also control logic for start, stop, and single stepping the processor.

5.4. Console Interface

The Console Interface allows the operator to control the processor. It allows bootstrapping, initialization and software update. It can access the CPU's major buses and key control points.

6. BOTTOM UP ANALYSIS OF VAX

The VAX 11/780 CPU uses 90 different integrated circuits. Total chip count is 2492. Column 1 and 2 are the Part numbers and the Part names. Column 3 is the number of that particular chip used in the machine. The data in column 3 is gathered by adding up the part list for each printed circuit board of the CPU. This part list is supplied with the VAX blueprints. Column 4 gives the number of transistors per chip. This is obtained by counting the transistors shown in the schematics of the TTL Data Book [TI76], the VAX System Maintenance Guide [DEC78b], the Fairchild ECL Data Book [Fa77], and the National Linear Data Book [Na78]. Like the analysis of the S-1 in section 4, I estimated the number of transistors for PROM's and RAM's by assuming one transistor per PROM cell and four transistors per RAM cell. Again only the decoding logic is counted for comparison purposes. Column five is obtained by multiplying column 3 and 4.

6.1. Part List of VAX

Part #	Part Name	Quantity	Tr/Chip	Total #tr
74S00	Quad 2-input NAND	83	16	1328
74S02	Quad 2-input NOR	21	24	504
74S03	Quad 2-input NAND (o.c.)	3	12	36
74S04	Hex Inverter	147	24	3528
74S05	Hex Inverter (o.c.)	4	18	72
74S08	Quad 2-input AND	47	24	1128
74S10	Triple 3-input NAND	50	12	600
74S11	Triple 3-input AND	37	18	666
74S14	Hex Schmitt-trigger Inverter	1	30	30
74S20	Dual 4-input NAND	29	8	232
74LS21	Dual 4-input AND	1	12	12
74S22	Dual 4-input NAND (o.c.)	7	6	42
74LS27	Triple 3-input NOR	1	18	18
74S32	Quad 2-input OR	6	32	192
74S37	Quad 2-input NAND Buffer	27	20	540
74S40	Dual 4-input NAND Buffer	6	10	60
74S51	Dual 2-wide 2-input A-O-I	36	12	432
74S64	4-2-3-2 input A-O-I	122	12	1464
74S65	4-2-3-2 input A-O-I (o.c.)	8	10	80
74S74	Dual D Flip-Flop	41	32	1312
74S85	4-bit Magnitude Comparator	11	178	1958

74S86	Quad 2-input XOR	5	80	400
74S112	Dual J-K Flip-Flop	30	44	1320
74123	Dual Monostable	6	30	180
74LS132	Quad 2-input NAND Schmitt-trigger	1	20	20
74S133	13-input NAND	18	6	108
74S138	3-to-8 Decoder	23	66	1518
74S139	Dual 2-to-4 Decoder	6	72	432
74S140	Duak 4-input NAND Line Driver	11	12	132
74148	8-to-3 Priority Encoder	12	92	1104
74S151	1-of-8 MUX	110	52	5720
74S153	Dual 4-to-1 MUX	154	64	9856
74S157	Quad 2-to-1 MUX	91	44	4004
74S158	Quad 2-to-1 MUX (inverted)	41	44	1804
74C160	Synchronous 4-bit Decade Counter	1	168	168
74C161	Synchronous 4-bit Binary Counter	8	156	1248
74LS165	Parallel Load 8-bit Shift Register	74	264	19536
74173	4-bit Register (tri-state)	1	156	156
74S174	Hex D Flip-Flop	35	122	4270
74S175	Quad D Flip-Flop	56	84	4704
74S181	Arithmetic Logic Unit	13	310	4030
74S182	Carry Look-ahead Generator	5	50	250
74S194	4-bit bidirectional Shift Register	103	156	16068
74S240	Octal Tri-state Inverting Buffer	60	60	3600
74S241	Octal Tri-state Buffer	10	60	600
74S251	8-to-1 MUX (tri-state)	17	52	884
74S253	Dual 4-to-1 MUX (tri-state)	16	48	768
74S257	Quad 2-ti-1 MUX (tri-state)	39	44	1716
74S258	Quad 2-to-1 MUX (inverted tri-state)	3	44	132
74LS259	3-bit Addressable Latch	5	62	310
74LS273	Octal D Flip-Flop	11	160	1760
74S280	9-bit Parity Generator/Checker	57	144	8208
74S283	4-bit Binary Full Adder	9	164	1476
74LS298	Quad 2-input MUX with Storage	8	148	1184
74S373	Octal D Latch	31	192	5952
74S374	Octal D Flip-Flop	1	192	192
74LS377	Octal D Flip-Flop	6	164	984
75451	Dual Peripheral Driver	3	10	30
75452	Dual NAND Peripheral Driver	1	14	14
10101	Quad 2-input OR/NOR	5	28	140
10104	Quad AND	2	28	56
10105	Triple 2-3-2 OR/NOR	4	21	84
10107	Triple Exclusive OR/NOR	1	32	32
10121	4-wide 3-input O-A/O-A-I	1	23	23
10124	Quad TTL to ECL Translator	4	48	192
10125	Quad ECL to TTL Translator	42	38	1596
10131	Dual D Flip-Flop	9	30	270
10174	Dual 4-to-1 MUX	1	84	84

10176	Hex D Flip-Flop	1	90	90
10216	High Speed Triple Line Receiver	1	38	38
25S10	4 bit Shift Register	69	84	5796
26S10	Quad Bus Transceiver	76	84	6384
85S68	64-bit Register file (tri-state)	55	304	16720
8640	Quad Bus Receiver	12	16	192
8641	Quad Bus Transceiver	1	36	36
8646	Transceiver	19	320	6080
93S16	Synchronus Binary Up Counter	46	132	6072
DC003	Interrupt	1	184	184
DC004	Protocol	1	196	196
DC005	4 Bit Transceiver	4	148	592
DC101	SBI Arbitration Chip	1	400	400
DC102	8-input Comparator	14	356	356
LM 339	Quad analog voltage comparator	1	30	30
	PROM 1K x 4	7	4440	31080
	PROM 256 x 8	116	2592	300672
	PROM 256 x 4	2	1568	3136
	PROM 64 x 8	1	664	664
	PROM 32 x 8	1	340	340
8209	RAM 64 x 9	12	2456	29472
93425A	RAM 1K x 1	213	4344	925272
	Total	2492		1459979

Table 5

6.2. Functional Blocks of VAX

The 90 different IC's can be classified into 24 different functional blocks :

Function	Chip	Quantity	% VAX	Total # Tr	% VAX
AND/NAND					
	74S00	83		1328	
	74S03	3		36	
	74S08	47		1128	
	74S10	50		600	
	74S11	37		666	
	74S20	29		232	
	74LS21	1		12	
	74S22	7		42	
	74S133	18		108	
	10104	2		56	
		277	11.11	4208	0.29
OR/NOR					
	74S02	21		504	
	74LS27	1		18	
	74S32	6		192	
	10101	5		140	
	10105	4		84	
		37	1.49	938	0.06
Exclusive-OR					
	74S86	5		400	
	10107	1		32	
		6	0.24	432	0.03
Inverter					
	74S04	147		3528	
	74S05	4		72	
		151	6.06	3600	0.25
Buffer					
	74S37	27		540	
	74S40	6		60	
	74S240	60		3600	
	74S241	10		600	
		103	4.13	4800	0.33

Schmitt Trigger					
	74S14	1		30	
	74LS132	1		20	
		2	0.08	50	0.003
OR-AND-INVERT					
	74S51	36		432	
	74S64	122		1464	
	74S65	8		80	
	10121	1		23	
		167	6.70	1999	0.14
Driver/Receiver/Transceiver					
	74S140	11		132	
	75451	3		30	
	75452	1		14	
	10216	1		38	
	26S10	76		6384	
	8640	12		192	
	8641	1		36	
	8646	19		6080	
	DC005	4		592	
		128	5.14	13498	0.92
Monostable					
	74123	6	0.24	180	0.01
TTL-ECL/ECL-TTL Translator					
	10124	4		192	
	10125	42		1596	
		46	1.85	1788	0.12
Flip-Flop					
	74S74	41		1312	
	74S112	30		1320	
	74173	1		156	
	74S174	35		4270	
	74S175	56		4704	
	74LS273	11		1760	
	74S374	1		192	
	74LS377	6		984	
	10131	9		270	

	10176	1		90	
		191	7.67	15058	1.03
Latch					
	74S373	31	1.24	5952	0.41
Decoder					
	74S138	23		1518	
	74S139	6		432	
	74LS259	5		310	
		34	1.36	2260	0.15
Multiplexer					
	74S151	110		5720	
	74S153	154		9856	
	74S157	91		4004	
	74S158	41		1804	
	74S251	17		884	
	74S253	16		768	
	74S257	39		1716	
	74S258	3		132	
	74LS298	8		1184	
	10174	1		84	
		480	19.26	26152	1.79
Adder					
	74S283	9	0.36	1476	0.10
Carry Look-ahead					
	74S182	5	0.20	250	0.02
Comparator					
	74S85	11		1958	
	DC102	14		4984	
		25	1.00	6942	0.48
Parity Generator/Checker					
	74S280	57	2.29	8208	0.56
Priority Encoder					
	74148	12	0.48	1104	0.08
Shift Register					

	75S165	74		19536	
	74S194	103		16068	
	25S10	69		5796	
		246	9.87	41400	2.84
Counter					
	74C160	1		168	
	74C161	8		1248	
	93S16	46		6072	
		55	2.21	7488	0.51
Arithmetic Ligic Unit					
	74S181	13	0.52	4030	0.28
Register File/Memory					
	85S68	55		16720	
	8209	12		29472	
	93425A	213		925272	
	PROM 1K x 4	7		31080	
	PROM 256 x 8	116		300672	
	PROM 256 x 4	2		3136	
	PROM 64 x 8	1		664	
	PROM 32 x 8	1		340	
		407	16.33	1307356	89.55
Others					
	LM339	1		30	
	DC003	1		184	
	DC004	1		196	
	DC101	1		400	
		4	0.16	810	0.06

Table 6

6.2.1. Register files and Memories in VAX

The register files/memory category is further subdivided into register files, programmable read only memory, and random access memory. The VAX has only 3 register files of size 16 words by 32 bits. All three are in the Data Path section. Two of them contains duplicate copies of the general purpose register file which is 16 words by 32 bits. The third one is for holding temporary storage

for addresses or operands generated by executing the microcode.

Register file/Memory	Size	Chips	Quantity	Total # Tr
Register file				
General purpose	36B x 16W	85S68	24	7296
Memory				
RAM		85S68	31	9424
		8209	12	29472
		93425A	213	925272
			256	964168
PROM			127	335892

Table 7

6.3. Ten most frequently used functional blocks by chips

Function	chips	% VAX
Multiplexer	480	19.26
Register files/Memory	407	16.33
AND/NAND	277	11.11
Shift Register	246	9.87
Flip-Flop	191	7.67
AND-OR-INVERT	167	6.70
Inverter	151	6.06
Driver/Receiver/Transceiver	128	5.14
Buffer	103	4.13
Parity Generator	57	2.29
Total	2207	88.56

6.4. Ten most frequently used functional blocks by transistors

Function	Transistors	% VAX
Register files/Memory	1307356	89.55
Shift register	41400	2.84
Multiplexer	26152	1.79
Flip-Flop	15058	1.03
Driver/Receiver/Transceiver	13498	0.92
Parity Generator	8208	0.56
Counter	7488	0.51
Comparator	6942	0.48
Latch	5952	0.41
Buffer	4800	0.33
Total	1436854	98.42

6.5. Ten least frequently used functional blocks by chips

Function	Chips	% VAX
Schmitt Trigger	2	0.08
Carry Lookahead Generator	5	0.20
Exclusive-OR	6	0.24
Monostable	6	0.24
Full Adder	9	0.36
Priority Encoder	12	0.48
Arithmetic Logic Unit	13	0.52
Comparator	25	1.00
Latch	31	1.24
Decoder	34	1.36
Total	143	5.72

6.6. Ten least frequently used functional blocks by transistors

Function	Transistors	% VAX
Schmitt Trigger	50	0.003
Monostable	180	0.01
Carry Lookahead Generator	250	0.02
Exclusive-OR	432	0.03
OR/NOR	938	0.06
Priority Encoder	1104	0.08
Full Adder	1476	0.10
TTL-ECL/ECL-TTL Translator	1788	0.12
AND-OR-INVERT	1999	0.14
Decoder	2260	0.15
Total	10477	0.71

7. PROPOSAL OF THE SET OF MODULES

Based on the analysis of the S-1 and the VAX, this section proposes a set of modules. The rationale of the choice of modules are :

1. The modules identified should be general and independent of the particular architecture used in the two machines analyzed.
2. The modules should be parameterizable so that they can be used in different parts of a new design.
3. The modules should represent some of the primitives that are useful in implementing computers.
4. The modules provided would be used in an interactive environment by the designer of a new computer.

7.1. High Level Modules

Using the top down approach, five modules are identified : Set associative cache, Arithmetic logic unit, Microsequencer, Shifter, and Programmable logic array. These five modules are described in the sections following.

7.1.1. Set Associative Cache

Both machines use set associative caches very extensively. The S-1 has two separate caches for instruction and data. Each one is 4-way set associative with a line size of four words. It holds the most recently referenced 4K words. The S-1 also has two separate translation buffers. One of them translate virtual instruction addresses to physical instruction addresses. The other one translates data addresses. Each translation buffer uses a 4-way set associative cache that holds the most recently referenced 64 page translations. The line size of the translation buffer caches is one word.

The VAX has only one data cache. It is 2-way set associative with a line size of one word and has a capacity of 2K words. VAX also uses a cache in the translation buffer. It is two way set associative with a line size of one word. Half of the

64 locations are reserved for system page table entries, and the other half are for process page table entries. System page table entries are not affected during a context switch.

The proposed set associative cache module is shown in figure 6. A description of the modules and the parameters is shown below. It should be noted that the parameters begins with a "C".

Parameters :

C1 Number of address bits

C2 Data word size (bits)

C3 Total capacity of cache (words)

C4 Set size of the cache

C5 Line size of the cache

C6 Number of parity bits for each data word

C7 Odd or Even parity

Since it is C4-way set associative, it has C4 memory banks and each memory bank holds $\frac{C3}{C4}$ words. This makes the low address register $\log_2\frac{C3}{C4}$. The high address register is then $C1-\log_2\frac{C3}{C4}$. The two address registers latch the physical address referenced in the current instruction.

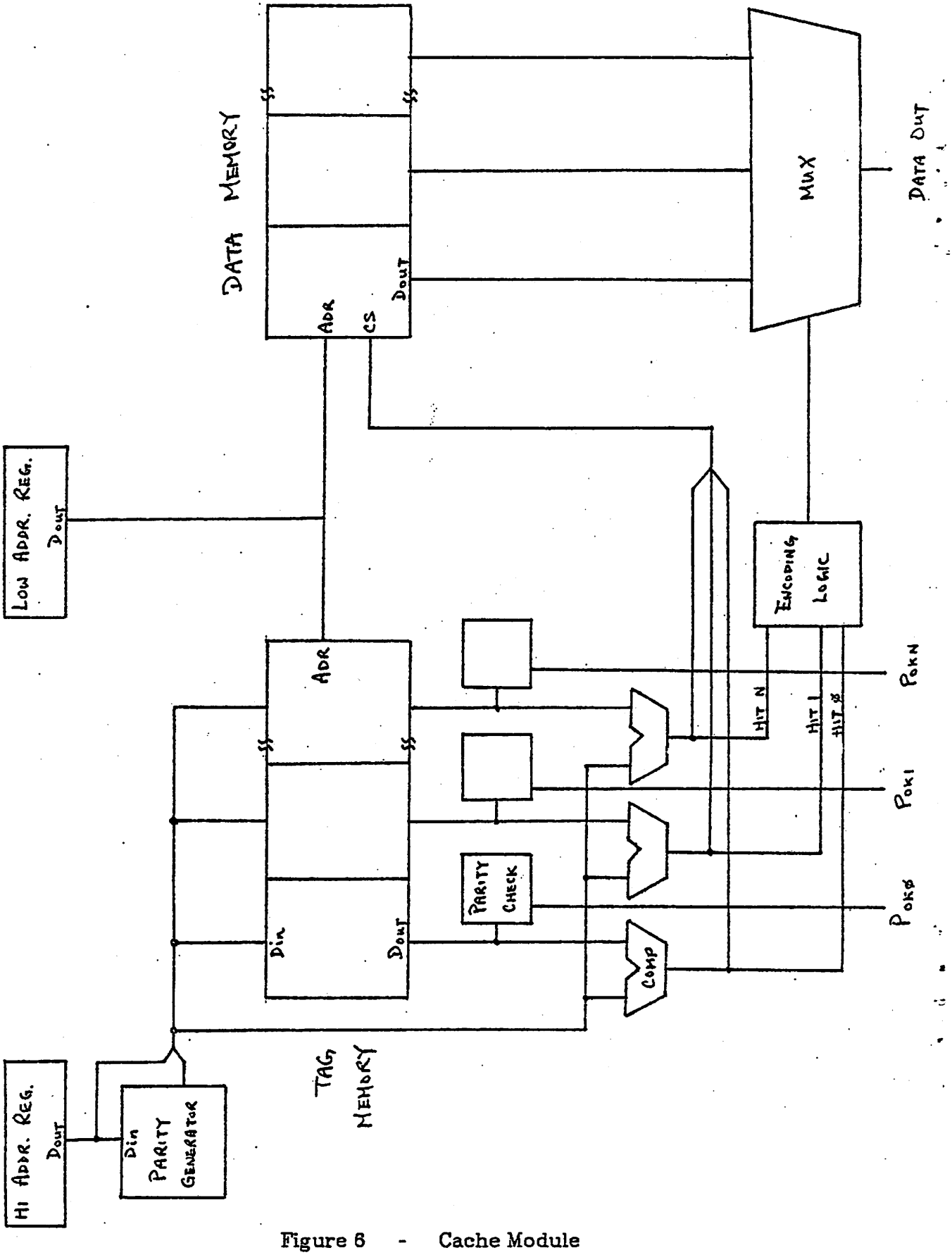


Figure 6 - Cache Module

During a cache read, the parity generator will generate C6 bits of C7 parity, where C7 is either odd or even. The parity bits are concatenated with the output of the high address register and compared with the output of the tag memory indexed by the low address register. If they are equal, it is a hit. Otherwise, it is a miss. The output of the tag memory is checked for parity. The low address register also indexes the data memory and C4 words are read out. The hit lines are used to select one out of C4 words in case of a cache hit.

During a write, depending on the cache strategy used, the data may or may not be written to the cache. If a write-not-allocate strategy is used, only main memory is updated. Otherwise logic external to this module has to decide which cache location should be thrown out in case of a miss. If it is a hit, the appropriate hit line should enable the corresponding bank of data memory.

This module can be further parameterized by making the address registers and the parity generator and checker optional. Also, depending on the technology used for implementation, the output data multiplexer can be a simple OR gate if the output of the memory is effectively open-circuited when it is not selected.

In order to make this module general, several features usually associated with caches are not considered. Logic for replacement in case of a cache miss is not implemented within the module since replacement algorithm is primarily architecture dependent. Also protection mechanism and logic for dealing with valid bit or dirty bit is not provided. It makes this module independent of the cache strategy.

7.1.2. Arithmetic Logic Unit

In the S-1, the arithmetic logic unit has been used in several different sections of the machine. It is used in the Data Address Arithmetic macro, the instruction decode macro, EBOX ALU macro, Exponent box macro, and the EBOX sequencer macro. All the arithmetic logic units used in these macros have input

multiplexers at either one or both inputs to the ALU. Also, one of the inputs to the multiplexers is usually either a register holding the operands or a register file.

The VAX uses three ALU's. One in the arithmetic section of the Data Path, one in the exponent section of the Data Path and one in the Condition Code Exception. Both ALU's in the Data Path have multiplexers at both inputs. The one in the arithmetic section has three 16 word register files, two go to the B multiplexer and one goes to the A multiplexer.

This module is shown in figure 7.

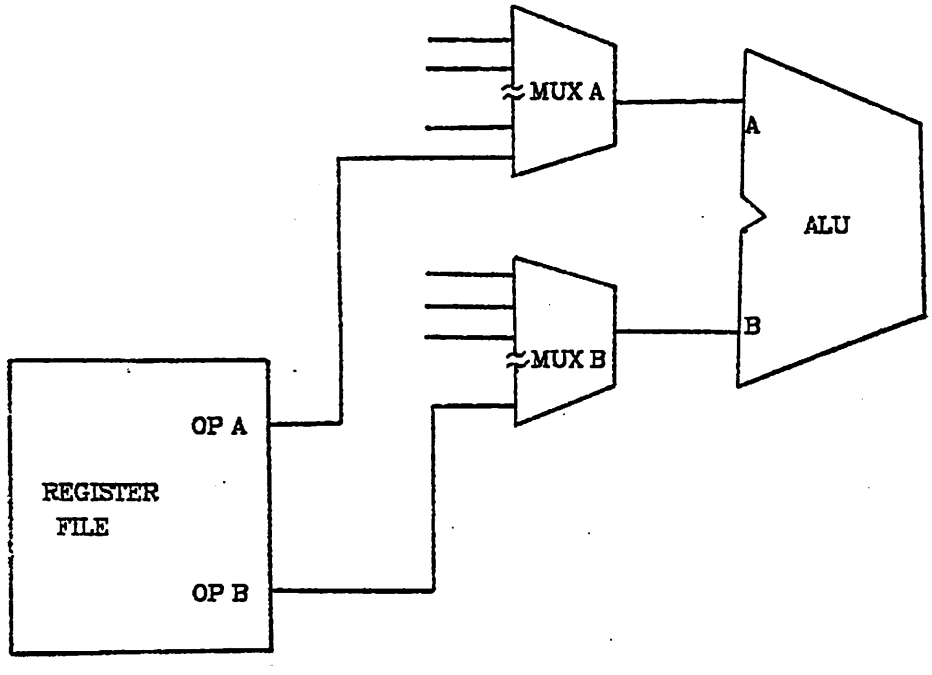


Figure 7 - Arithmetic Logic Unit Module

Parameters :

A1 Data path width (bits)

A2 Size of input multiplexer A

A3 Size of input multiplexer B

A4 Size of register file (words)

A5 Single port or dual port register file

Both A2 and A3 can be zero, meaning that input multiplexer A or B or both is non-existent. If A4 is zero, no register file is needed. If A4 is one, there will be only one register. If A5 is one, the register file is only associated with MUX A. If A5 is two, the register file is only associated with MUX B. If A5 is three, two separate register files are associated with MUX A and B. If A5 is four, there is one dual port register file.

7.1.3. Shifter

A shifter made out of multiplexers is a rather popular part in both machines analyzed. It is used to align data on certain word boundaries. It is used to do multiplication and division when used together with the ALU. It can also be used to perform the shift instructions in the instruction set.

Parameters :

S1 Data path width (bits)

S2 An array which specifies the number of bits that can be right shifted.

S3 An array which specifies the number of bits that can be left shifted.

An example of a shifter module is the EBOX shifter in the S-1. It can shift right 1 to 16 bits and shift left 0 to 47 bits. S1 is 72 bits. S2 is an array of 72 elements. The first 16 elements have values 1,2,3,.....,16. The rest of the elements is empty. S3 is an array of 72 elements. The first 48 elements have values 0,1,2,....,47. The rest of the elements is empty. This shifter module is shown in figure 8.

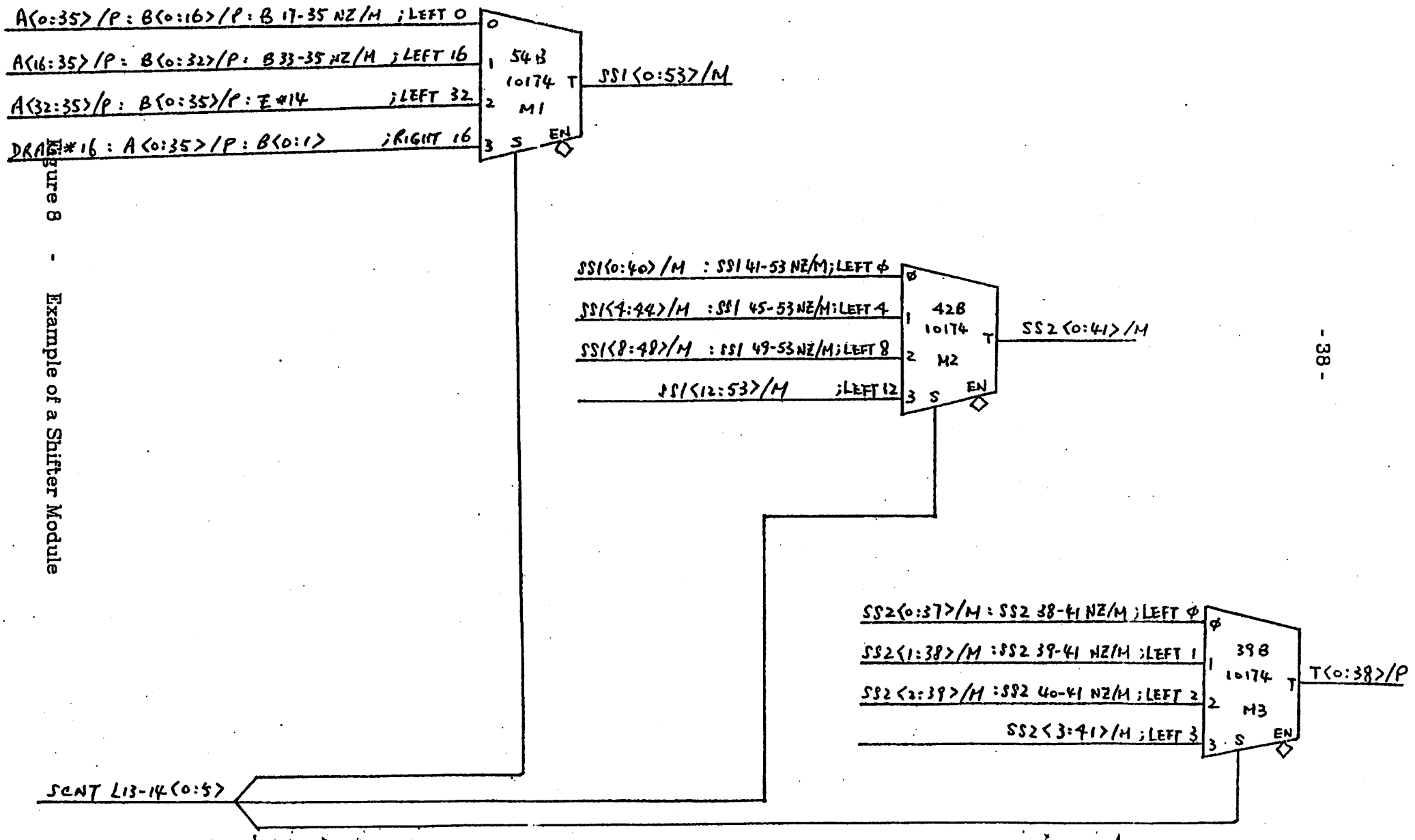


Figure 8 - Example of a Shifter Module

7.1.4. Microsequencer

7.1.4.1. S-1 Microsequencer

The S-1 has four separate microsequencers to do extensive pipelining. The Instruction Box contains three sequencers: the I Sequencer, the P Sequencer, and the F Sequencer.

7.1.4.1.1. I Sequencer

The I Sequencer is a powerful micro-programmed controller. It can branch anywhere in its control store. It can make micro-subroutine calls which can be nested up to 16 levels deep. It can also performs microinterrupts by stacking the return address. It does all complex operand address calculation, controls the caches and translation buffers, controls main memory transfers and execute complex instructions. The control store is indexed by a binary counter which can either count up or load from a multiplexer. The output of the counter goes to a register called the microprogram counter. The microprogram counter is connected to the input of a micro stack. The stack is actually a 16 word random access memory indexed by a 4 bit up/down counter. A push will increment the counter and data is written into the RAM. A pop will decrement the counter and data is read. The output of the control store goes to a register and is parity checked. Size of the control store is 1K by 224 bits.

7.1.4.1.2. P Sequencer

The P Sequencer controls the reading and writing of register direct operands. For non-register-direct operands, it starts up the I Sequencer. This is a much simpler microsequencer. It does not use a counter to address the control store. Instead, it always branch to the next address field of the current micro word. It does not have a micro-stack. Control store size is 1K by 64 bits.

7.1.4.1.3. F Sequencer

The F Sequencer controls the fetching of instruction and the execution of execute instructions. It branches on whether the instruction length is 1, 2, or 3 words, and whether the instruction wants to be followed by the next sequential instruction, or branch. Addressing the control store is similar to the I Sequencer. It also uses a counter which increment or load. The output of the control store is parity checked. It does not have a microprogram counter nor a micro-stack. Control store size is 256 by 24 bits.

7.1.4.1.4. EBOX Sequencer

The EBOX Sequencer has similar features as the I Sequencer. It uses a counter and a multiplexer to determine the next address. It has a micro program counter and a micro-stack. The capacity of the control store is 4K by 196 bits.

7.1.4.2. VAX Microsequencer

It has a multiplexer to select one of the many sources of the next address of the control store. However, it does not have a counter It get its next address from the next address field of the current control word. It does have a micro program counter which is called the UPCSV REG. It also has a 16 level deep micro stack.

The VAX has two control stores: the PROM control store and the Writable contol store. The PROM control store is 4K by 99 bits where three of the 99 bits are parity bits. The output of the control store is parity checked. The writable control has similar structure but the size is only 1K by 99 bits.

7.1.4.3. The Microsequencer module

After analyzing the 5 different microsequencers, the following microsequencer module is proposed. It is shown in figure 9.

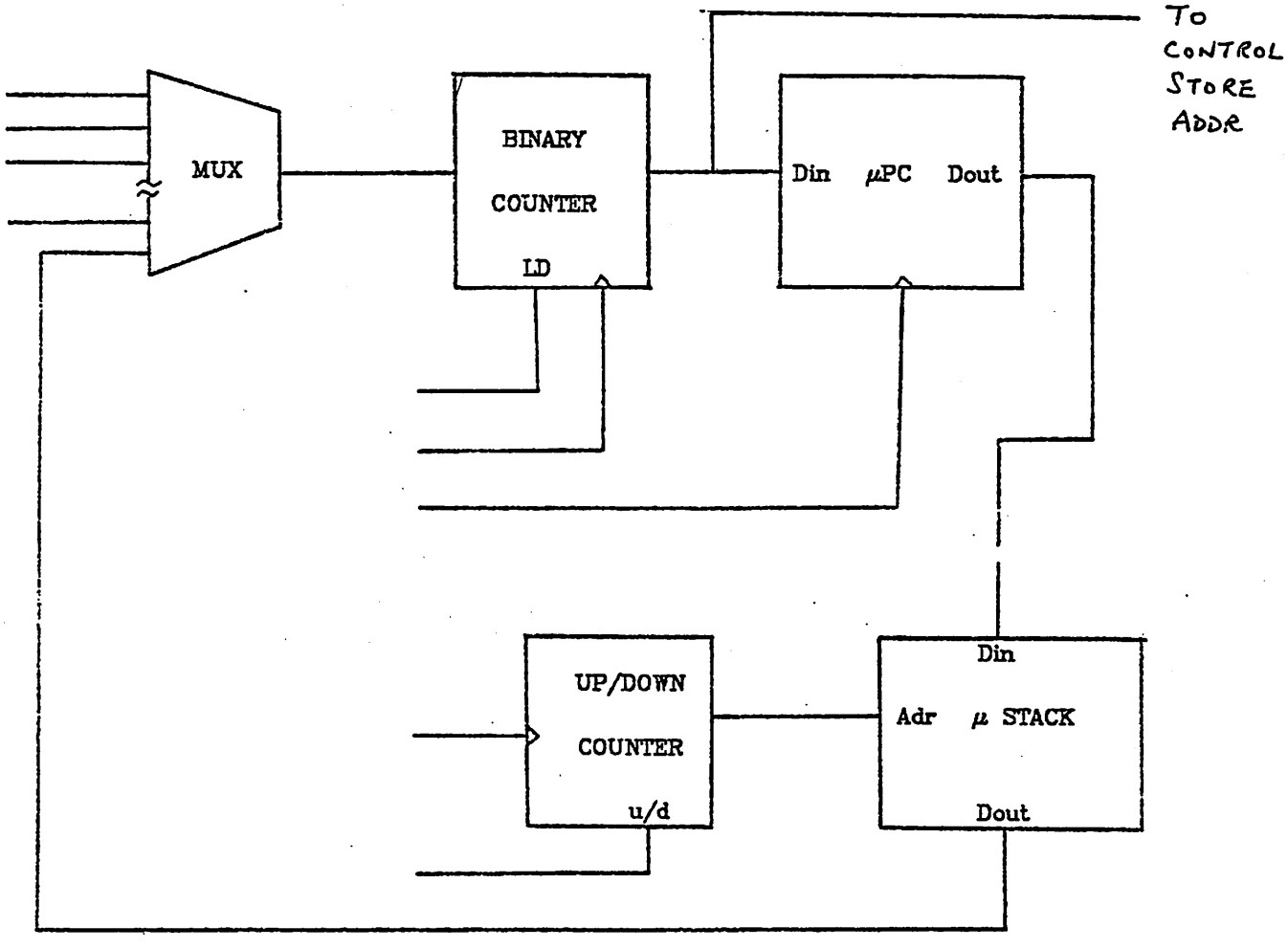


Figure 9 - Microsequencer Module

Parameters :

M1 Size of multiplexer

M2 Size of binary counter (bits)

M3 Size of micro-stack (words)

M4 Number of bits of each micro-word

M5 Number of parity bits for each micro-word

An input multiplexer will select a source for the next address of the control store. The output of the multiplexer will go to the parallel input of a binary up counter. The size of this counter will be specified. The size of the control store is then 2^{M2} words. This counter will be controlled by the current microword. If the next sequential microword is desired, the counter will simply be enabled so that it will count up with the incoming clock pulse. If a branch is desired, the counter will load from the output of the multiplexer. A microprogram counter will be provided to store the address of the current microword. The micro-stack will be optional depending on the needs of the designer using this module. The micro-stack pointer is an up/down counter with size $\log_2 M3$ bits.

The control store will be governed primarily by two parameters: the number of words and the size of each word. All the address decoding and buffers will be provided. Another parameter will be the number of parity bits.

7.1.5. Programmable Logic Array

As the machine complexity grows, it becomes clear that the PLA is a better solution for implementing random logic. Recently, Data General announced their highest performance minicomputer, the MV/8000 [AlHo80]. They claimed that 10% of the CPU components is implemented by PLA. In places where propagation delay is not critical, PLA becomes a very attractive alternative for random logic. In VLSI, PLA has a more regular structure and less chip area than conventional random logic. It also makes the debugging stage much easier. Small errors in the logic equations could be corrected with a least amount of changes in the lay-out.

The parameters are the logic equations to be implemented by the programmable logic array.

7.2. Low Level Modules

Using the bottom up approach, 12 modules are identified. There are 19 low level modules in S-1 and 24 in the VAX. I grouped the simple logic gates under a category for both machines. These logic gates include AND, NAND, OR, NOR, XOR, inverter, buffer, and AND-OR-INVERT. I also grouped the storage elements, the flip-flops and the latches under the same category. Since parity generator is a kind of error detection, it is grouped with error detection / correction circuits under the same category. Arithmetic elements like the adder, the carry look-ahead and the ALU are grouped under arithmetic logic unit. Finally, the Schmitt trigger, monostables, and logic family translators are discarded because they represent a very insignificant percentage of the machine.

1. Logic gate
2. Driver
3. Flip-Flop/ Latch
4. Decoder
5. Multiplexer
6. Comparator

7. Error Detection and Correction
8. Priority Encoder
9. Shift Register
10. Counter
11. Arithmetic Logic Unit
12. Register file/ Memory

7.2.1. Logic Gate Module

Logic gates including AND, NAND, OR, NOR, exclusive OR, inverter, and AND-OR-INVERT will be used in speed critical regions or implementing very simple logic equations where a PLA is not justified. It will also be used as buffers.

Parameters :

G1 Type

G2 Number of inputs

G3 Data Path Width (bits)

7.2.2. Driver Module

The module will be used only for interfacing the VLSI computer with the outside world. It will not be as significant on the chip level as it is on the board level.

7.2.3. Flip-Flop/ Latch Module

Parameters :

F1 Type - D-type, JK-type, T-type or D-latch

F2 Data Path Width (bits)

A register can be made from this module by specifying F1 to be D-type and F2 to be the length of the register.

7.2.4. Decoder/Demultiplexer Module

Parameters :

De1 Type - Decoder or Demultiplexer

De2 Number of inputs

De3 Inverting or Non-inverting outputs

A decoder tree can be assembled by specifying De1. The second parameter, De2, is provided to minimize extra logic. The number of select lines is $\log_2 De\,2$.

7.2.5. Multiplexer Module

Parameters :

Mu1 Number of inputs

Mu2 Inverting or Non-inverting output

Mu3 Data Path Width

A multiplexer tree can be assembled by specifying Mu1. The number of select lines is $\log_2 Mu\,1$. A vector of multiplexers can be assembled by specifying Mu3.

7.2.6. Comparator Module

Parameters :

Co1 Type - Equality or Magnitude Comparator

Co2 Data Path Width (bits)

7.2.7. Error Detection and Correction Module

Parity generator/checker and Hamming code generator/corrector can be assembled using this module.

Parameters :

E1 Type - parity (single error detection) or
hamming code (single error correction, double error detection)

E2 Data word size (bits)

The S-1 uses the MC 10163 (a Motorola part) for generating the modified Hamming single-error-correction, double-error-detection (SEC-DED) code used in the IBM 370/145 memory. An example of this module for data word size of 64 bits is shown in the appendix.

When writing into memory, the check bit generator generates check bits which are stored with the data bits. To generate these check bits, one error detection-correction chip and one parity generator- checker are used for each bit. The detail of connections depends on the data word size.

During the memory read operation the fetched check bits previously generated are exclusive-ORed with newly generated check bits to generate the syndrome bits. The syndrome bit decoder would determine if there is any error. If single error is detected, the decoder would generate a "fix" word which has a "1" in the position of the error bit and zero otherwise. That bit is "fixed" by exclusive-ORing with the incoming data word. Multiple errors are also detected by the decoder.

7.2.8. Priority Encoder Module

Parameter :

P1 Number of priority levels

The number of outputs is $\log_2 P$ 1.

7.2.9. Shift Register Module

Parameter :

SR1 Data Path Width (bits) An N bit shift register can be assembled by specifying SR1.

7.2.10. Counter Module

Parameter :

Ct1 Synchronous or Asynchronous

Ct2 Type - Binary count, decade count or divide by N count

Ct3 If Ct2 is divide by N count, Ct3 is used to specify N.

Ct4 Up-down count capability - up-down count or up count only

Ct5 Preset capability

Ct6 Data Path Width (bits)

A great variety of counters can be assembled from this counter module.

7.2.11. Arithmetic Logic Unit

Parameters

a1 Type - Full adder or arithmetic logic unit

a2 Type of carry propagation - carry-save, carry propagate or carry look-ahead.

a3 Data Path Width (bits)

7.2.12. Register File/ Memory module

Both the VAX and the S-1 are memory intensive designs as revealed by the statistics provided in table 6 and table 7. The obvious parameters for this module is number of bits per word and total number of words. For large memories such as the control store, all address decoding should be provided by the module. Also, in order to improve reliability, some kind of error detection scheme such as parity generating and checking should also be provided.

Parameters :

R1 Register file or memory size (words)

R2 Word size (bits)

R3 Error detection scheme - Parity or Hamming code

R4 Number of parity bits

R5 Odd or even parity

8. EVALUATION OF THE SET OF MODULES

8.1. Evaluation of the high level modules

Using the high level modules proposed in the previous section, 82% of the S-1 and 87% of the VAX can be built. The two tables given below provide useful statistics about how important each module is. A part list for each module used in both machines is given in Appendix B.

Module	#chips	% S-1	#chips	% VAX
Cache	760	13.28	193	7.74
Arithmetic Logic Unit	545	9.53	113	4.53
Shifter	141	2.46	106	4.25
Microsequencer and Control Store	801	14.00	274	11.00
Programmable Logic Array	1460	25.52	632	25.36
Total	3707	64.79	1318	52.88

Table 8

Module	#transistors	% S-1	%transistors	% VAX
Cache	1634094	27.30	534696	36.62
Arithmetic Logic Unit	171200	2.86	24432	1.67
Shifter	7191	0.12	7924	0.54
Microsequencer and Control Store	3045616	50.89	697540	47.78
Programmable Logic Array	33401	0.56	10745	0.73
Total	4891502	81.73	1275337	87.34

Table 9

8.2. Evaluation of Low Level Modules

The relative importance of each low level module is revealed by the statistics in table 10 and 11. Examining table 11 reveals that memory is the single most important ingredient in modern computer design. Multiplexer comes second. The high percentage of shift registers used in the VAX is due to the shifting of internal states of the processor to the front console for diagnotics purposes. The S-1 uses multiplexers to access the internal states of the processor from the front console. Also, the various VAX data types require alignment of data which involves shifting.

Both machines uses a lot of flip-flops and registers. Random logic gates comes fourth in both machines, if the shift registers in the VAX are discarded.

If we look at table 10, again discarding the shift registers in the VAX, the same four categories are the most important building blocks of the two computers. The comparators for the S-1 are built from exclusive-OR gates therefore leaving a zero chip count in that category.

8.3. Comparison of VAX and S-1 by chips

Comparison is done by grouping the functional blocks identified in section 4.2 for the S-1 and section 6.2 for the VAX into the various low level modules. The comparison is done in terms of chip counts. Top five most frequently used modules are given in column 4 and 7.

	S-1			VAX		
Module	chips	% S-1	Rank	chips	% VAX	Rank
Logic Gate	1639	28.65	2	741	29.74	1
Driver	49	0.86		128	5.14	
Flip-Flop/Latch	626	10.94	4	222	8.91	5
Decoder	30	0.52		34	1.36	
Multiplexer	1346	23.53	3	480	19.26	2
Comparator	0	0		25	1.00	
Error Detection/Correction	109	1.90		57	2.29	
Priority Encoder	12	0.21		12	0.48	
Shift Register	44	0.77		246	9.87	4
Counter	28	0.49		55	2.21	
ALU	170	2.97	5	27	1.08	
Register File/Memory	1668	29.16	1	407	16.33	3
Total	5721	100.00		2434	97.67	

Table 10

This does not represent 100 % of VAX because some of the low level modules of VAX have been discarded. (see section 7.2)

8.3.1. Register file/Memory Category

Since the register file/memory category represent a significant portion of both machines, it is subdivided into register files, programmable read only memory, and random access memory.

Parts	chips	% VAX	chips	% S-1
Register files	24	0.96	156	2.73
PROM	127	5.1	0	0
RAM	256	10.27	1512	26.43
Total	407	16.33	1668	29.16

8.4. Comparison of VAX and S-1 by transistors

Comparison is done in terms of the number of transistors in each module and the percentage by transistors of that module being used in both machines.

	S-1			VAX		
Module	Tr	% S-1	Rank	Tr	% VAX	Rank
Logic Gate	45821	0.77	5	15977	1.09	5
Driver	980	0.02		13498	0.92	
Flip-Flop/Latch	64191	1.07	3	21010	1.44	4
Decoder	1726	0.03		2260	0.15	
Multiplexer	102568	1.71	2	26152	1.79	3
Comparator	0	0		6942	0.48	
Error Detection/Correction	11635	0.19		8208	0.56	
Priority Encoder	2460	0.04		1104	0.08	
Shift Register	9328	0.16		41400	2.84	2
Counter	7194	0.12		7488	0.51	
ALU	55010	0.92	4	5756	0.39	
Register File/Memory	5683860	94.97	1	1307356	89.55	1
Total	5984773	100.00		1457151	99.80	

Table 11

8.4.1. Register file/Memory Category

Again, the register file/memory category is subdivided into register files, PROM's and RAM's.

Parts	Transistors	% VAX	Transistors	% S-1
Register files	14784	1.01	523080	8.74
PROM	335892	23.0	0	0
RAM	956680	65.53	5160780	86.23
Total	1307356	89.54	5683860	94.97

9. CONCLUSION

Analysis of the two machines shows that computers have not been built in a very structured or regular manner. The S-1, hierarchical as it appears, has only 14 higher level macros that have been called more than once by higher still macros. Future VLSI computers must employ a more regular design to overcome the increasing complexity and lack of design tools [PaSe]. The modules identified suggest the importance of this design direction. The Microsequencer and the PLA are the first steps towards regular designs. Computer designers have already noticed memory intensive designs as the viable approach to complex computing system realization. However, memory intensive design approaches should not be abused in VLSI computer design. Chip area is still precious. Some systems of logic equations could be minimized with very little effort, which makes PLA realization more attractive than ROM realization.

By studying the two computers, two sets of modules have been identified. The set of low level modules will represent virtually 100 % of both machines. The set of high level modules is provided to lessen the design time by assembling frequently used structures in modern computer architecture like the cache module and the microsequencer module. The identification of these modules is the first step towards developing the tools that will cope with the expanding potential of VLSI.

10. REFERENCES

[AlHo80] Alsing, C.J., Holberger, K.D., Holland, C.J., Rasala, E.J., and Wallach, S.J., "Minicomputer Fills Mainframe's Shoes", Electronics, May 22, 1980, p. 130.

[BeMu78] Bell, C.J., Mudge, J.C.,and McNamara, J.E.,"Computer Engineering", Digital Equipment Corporation, pp. 409-428.

[DEC78a] VAX 11/780 TB/CACHE/SBI Control Technical Description, Digital Equipment Corporation,Document No. EK-MM780-TD.

[DEC78b] VAX 11/780 System Maintenance Guide, Digital Equipment Corporation, Document No. EK-11780-PG-001.

[DEC79] VAX 11/780 Central Processing Unit Technical Description, Digital Equipment Corporation, Document No. EK-KA780-TD.

[Fa77] "ECL Data Book", Fairchild Camera and Instrument Corporation, Mountain View, California.

[He72] Helliwell, D., "The Stanford University Drawing System", Stanford Artificial Intelligence Laboratory, Palo Alto, California, 1972.

[McWi78a] McWilliams, T.M.,and Widdoes, L.C.. "SCALD: Structured Computer-Aided Logic Design", Proc. Ann. Design Automation Conf., 15th, Las Vegas, 1978 (IEEE, ACM, New York, 1978), p.271.

[McWi78b] McWilliams, T.M.,and Widdoes, L.C., "The SCALD Physical Design Subsystem", Proc. Ann. Design Automation Conf., 15th, Las Vegas, 1978.

[Mo78] Technical Information Center, Motorola Inc., "MECL High Speed Integrated Circuits", Motorola Inc., Phoenix, Arizona.

[Na78] "Linear Databook", National Semiconductor Inc., Santa Clara, California.

[PaSe80] Patterson, D.A.,and Sequin, C.H., "Design Considerations for Single-Chip Computers of the Future", IEEE Transaction on Computers, Vol. C-29, No. 2, February 1980, p. 108.

[TI76] The Engineering Staff of Texas Instruments Inc. Semiconductor Group, "The TTL Data Book for Design Engineers", 2nd Edition, Texas Instruments Inc.

[Wi80] Widdoes, L.C., "The S-1 Project: Developing High-Performance Digital Computers", IEEE Computer Society COMPCON Spring 1980 Meeting, San Francisco, California, February, 1980.

APPENDIX A

Error Detection Correction Circuits

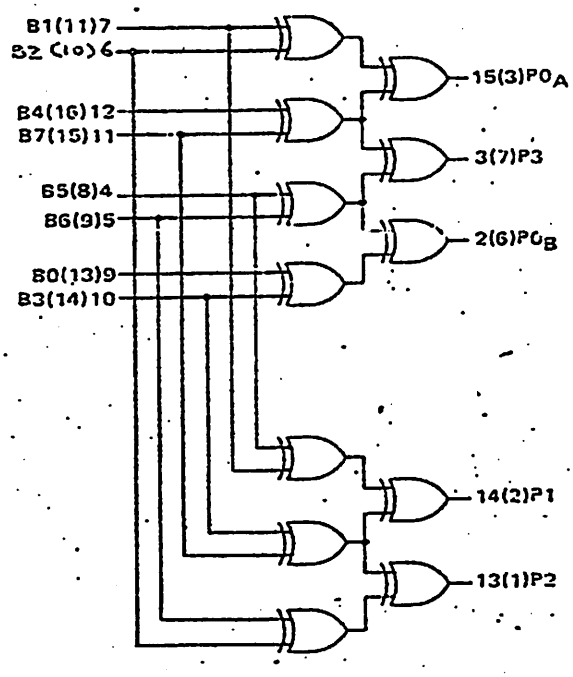
This is a copy of the data sheet of the MC10163 Error Detection/ Correction Circuit from the Motorola MECL Data Book[Mo78]. It gives an example of using the MC10163 together with the MC10160 Parity Generator/Checker to do single error correction, double error detection for a 64 bit data word.

MC10163/MC10563
MC10193/MC10593

ERROR DETECTION - CORRECTION CIRCUITS

The MC10163/MC10563 and the MC10193/MC10593 are error detection and correction circuits. They are building blocks designed for use with memory systems. They offer economy in the design of error detection/correction subsystems for main-frame and add-on memory systems. For example, using eight MC10163's together with eight 12-bit parity checkers (MC10160), single-bit error detection/correction and double-bit error detection can be done on a word of 64-bit length. Only eight check bits (B0–B7) need be added to the word. A useful feature of this building block is that the MC10193/MC10593 option generates the parity of all inputs to the block. Thus, if the MC10193 is applied in a byte sequence, individual byte parity is automatically available.

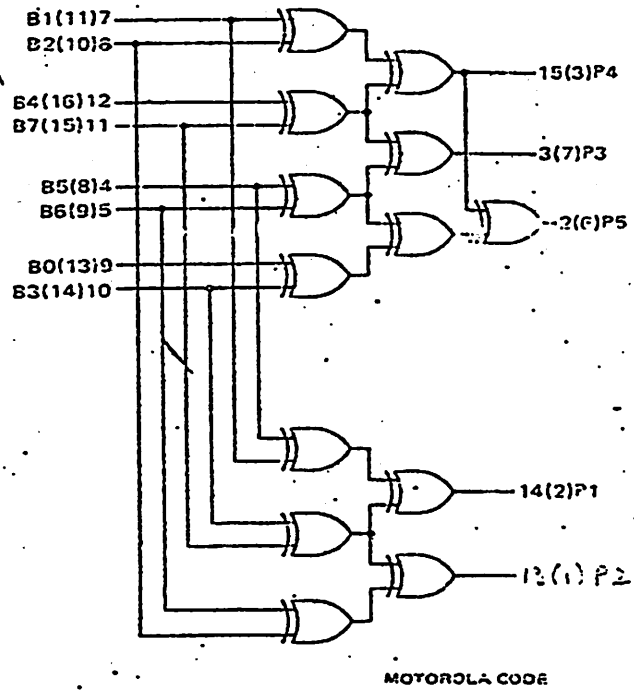
MC10163/MC10563 LOGIC DIAGRAM



IBM CODE

$P0_A$ = B1, B2, B4, B7
$P0_B$ = B0, B3, B5, B6
P1 = B1, B3, B5, B7
P2 = B2, B3, B6, B7
P3 = B4, B5, B6, B7
P_D = 520 mW typ/pkg (No Load)
t_{pd} = 5.0 ns typ

MC10193/MC10593 LOGIC DIAGRAM



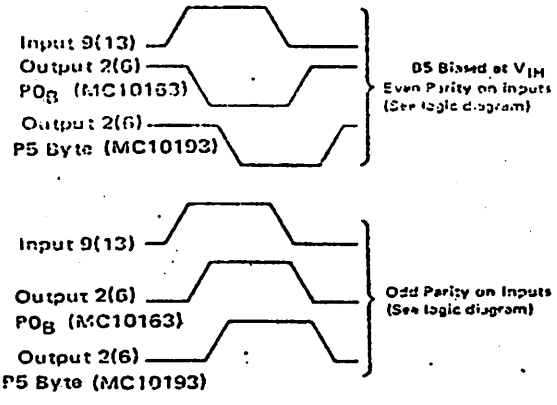
MOTOROLA CODE

P1 = B1, B3, B5, B7
P2 = B2, B3, B6, B7
P3 = B4, B5, B6, B7
P4 = B1, B2, B4, B7
P5 = Byte (B0, 1, 2, 3, 4, 5, 6, 7)
t_{pd} = 7.5 ns typ (to P5)
= 5.0 ns typ (to P1–P4)

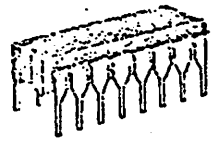
V_{CC1} = Pin 1(5)
V_{CC2} = Pin 16(4)
V_{EE} = Pin 8(12)

Numbers at ends of terminals denote pin numbers for L and P packages.
Numbers in parenthesis denote pin numbers for F package.

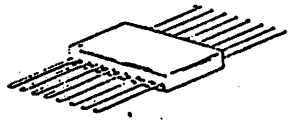
SWITCHING TIME WAVEFORMS @ 25°C



P SUFFIX
PLASTIC PACKAGE
CASE 648
MC10163 and
MC10193 only



L SUFFIX
CERAMIC PACKAGE
CASE 620



F SUFFIX
CERAMIC PACKAGE
CASE 650
MC10563 and
MC10593 only

Characteristic	Symbol	−55°C Min	−55°C Max	−30°C Min	−30°C Max	+25°C Min	+25°C Max	+85°C Min	+85°C Max	+125°C Min	+125°C Max	Unit
Power Supply Drain Current	I_E	–	137	–	137	–	125	–	137	–	137	mAdc
Input Current	I_{inH}											μAdc
Pins 4,6,10		–	375	–	350	–	220	–	220	–	220	
Pins 5,7,9,11,12		–	450	–	425	–	265	–	265	–	265	
Switching Times												ns
Propagation Delay	t_{pd}											
MC10163/MC10563		1.3	7.0	1.3	6.8	1.5	6.5	1.5	7.1	1.5	7.5	
MC10193/MC10593 B to P1–P4		1.3	7.1	1.3	6.8	1.5	6.5	1.5	7.1	1.5	11	
B to P5		1.8	9.1	1.8	8.9	2.0	8.5	2.0	9.2	2.0	10	
Rise Time, Fall Time (20% to 80%)	t_+, t_-											ns
MC10163/MC10563		1.1	4.4	1.1	4.2	1.1	3.9	1.1	4.4	1.1	4.5	
MC10193/MC10593		1.1	4.3	1.1	4.2	1.1	3.9	1.1	4.4	1.1	4.6	

−55°C and +125°C test values apply to MC105xx devices only.

MC10163/MC10563 APPLICATIONS INFORMATION

The MC10163/MC10563 is a building block for generating the modified Hamming single-error-correction, double-error-detection (SEC-DED) code used in the IBM370/145 memory. While the MC10163 can also be used for generating other patterns, it is optimized for generating the pattern shown in the H matrix of Figure 1.

When writing into a memory, the MC10163 is used to generate the eight check bits (C0–C32, CT) which are stored with the 64 data bits (B0–B63). These check bits are generated by taking the parity of all data bits marked with an X in the appropriate row of the H matrix. (C0, C1, C32, CT, are even parity; C2, C4, C8, C17, are odd parity.) To generate these check bits with the building blocks, eight MC10163's and eight MC10160 parity checkers are used. One MC10163 is connected to each byte of data and the outputs of these building blocks are connected to the eight MC10160 parity checkers, one for each check bit. Figure 2 shows which connections are required (i.e., C0 is the even parity of output $P0_A$ of the MC10163 on the "zero" byte of data, output $P0_B$ of the "zero" byte, ..., $P0_B$ of the "three byte and data bit 32.)

During the memory read operation, the fetched check bits previously generated (as described) are exclusive-ORed with newly generated C0–C32 to generate syndrome bits S0–S32. Syndrome ST is a special case where ST is the even parity of all eight fetched check bits and all 64 fetched data bits. For determining the type and location of an error:

1. If all syndromes (S0–S32 and ST) are false, there is no error.
2. If ST is true and S0–S32 are false, the CT is in error.
3. If ST is false and one or more of S0–S32 is true, an uncorrectable error has occurred.
4. If ST is true and one or more of S0–S32 is true, simply add the S1–S32 bits to get the binary location of the error (S1 has weight 1, S2 weight 2, S4 weight 4, etc.)

Data bits B0 and B32 are special cases of this location technique: B0 is in error if ST, S0, and S32 are true; B32 is in error if ST, S0, S1, and S32 are true.

FIGURE 1 – 370/145 PATTERN

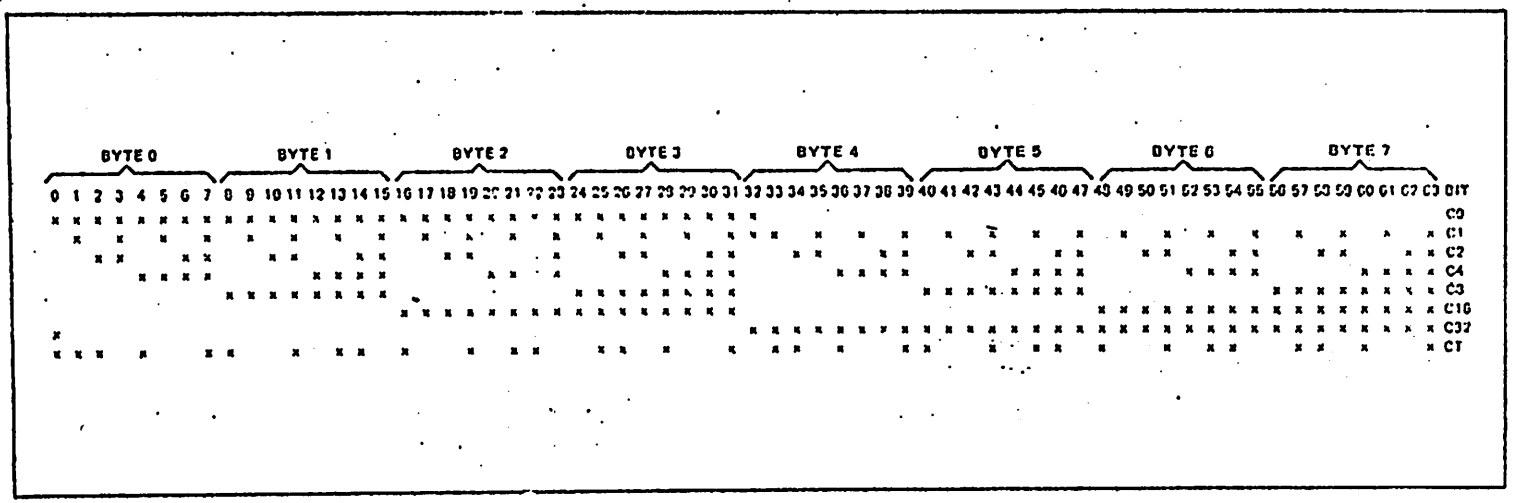


FIGURE 2 – 370/145 PATTERN GENERATION

C0 = $P0_{A0}$	$P0_{30}$	$P0_{A1}$	$P0_{B1}$	$P0_{A2}$	$P0_{B2}$	$P0_{A3}$	$P0_{B3}$	B(32)
C1 = P10	P11	P12	P13	P14	P15	P16	P17	B(32)
C2 = P20	P21	P22	P23	P24	P25	P26	P27	
C4 = P30	P31	P32	P33	P34	P35	P36	P37	
C8 = $P0_{A1}$	$P0_{B1}$	$P0_{A3}$	$P0_{B3}$	$P0_{A5}$	$P0_{B5}$	$P0_{A7}$	$P0_{B7}$	
C16 = $P0_{A2}$	$P0_{A2}$	$P0_{A3}$	$P0_{B3}$	$P0_{A6}$	$P0_{B6}$	$P0_{A7}$	$P0_{B7}$	
C32 = $P0_{A4}$	$P0_{B4}$	$P0_{A5}$	$P0_{B5}$	$P0_{A6}$	$P0_{B6}$	$P0_{A7}$	$P0_{B7}$	B(0)
CT = $P0_{A0}$	$P0_{B1}$	$P0_{B2}$	$P0_{A3}$	$P0_{A4}$	$P0_{B5}$	$P0_{B6}$	$P0_{A7}$	B(0)

Where for P_{NM}: N = MC10163 Output
M = Byte Number

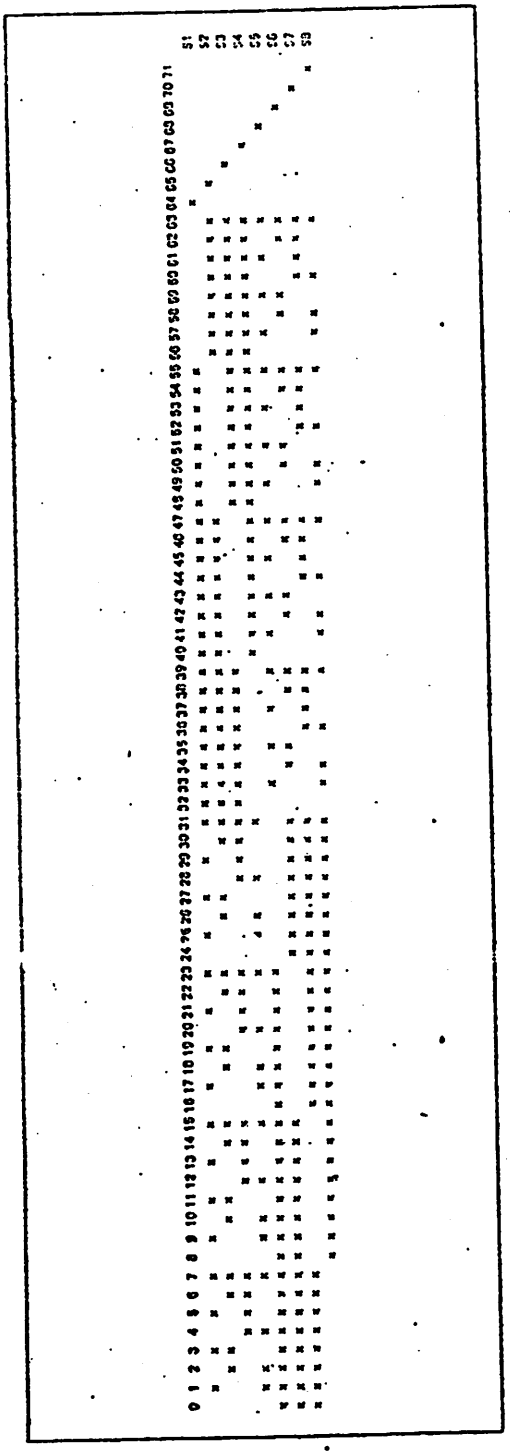


FIGURE 3 — MOTOROLA PATTERN EXAMPLE

The MC10193/MC10593 is a building block for generating modified Hamming SEC-DED codes. It can be used for any length data word and for a variety of codes. The MC10193 is optimized for codes organized on a byte repetitive basis and has the advantage of automatically supplying whole byte parity (P5 output). While it is possible to use a number of criteria for choosing a pattern, the pattern of Figure 3 was chosen on the basis of speed and ease of error location decode. As can be seen in the H matrix of Figure 3, the pattern is repetitive by byte with the various rows generated by only five combinations of bit parities within the bytes. For the 64 bit data word in the example of Figure 3, the eight check bits (B64 to B71) are generated by the odd parity of all data bits indicated by an X in the appropriate row. The syndromes S1 to S8 are generated by including the fetched check bits in the same generator that originally generated the check bits.

The pattern of Figure 3 is easily generated by using eight MC10193 devices, one for each data byte and eight MC10160 parity checkers, one for each syndrome/check bit. The connections of building blocks and parity checkers are shown in tabular form in Figure 4 and in schematic form in Figure 6.

Once the syndrome bits (S1 to S8) have been formed from fetched data (B0 to B63) and fetched check bits (B64 to B71), the determination of type and location of error is simply done:

1. If all syndromes are false, there is no error.
2. If one syndrome is true, the corresponding check bit is in error.
3. If more than one syndrome is true and the parity of all syndromes is even, a multiple (uncorrectable) error has occurred.
4. If more than one syndrome is true, and the parity of all syndromes is odd, a single error has occurred and is easily located by the circuit of Figure 5.

Figure 5 gives the error location circuit for the example pattern. The outputs EB0 to EB6 are a one-of-eight-high code giving the byte in error. Outputs EC0 to EC3 give the binary location of the bit in error within the located byte. Since this location process can occur simultaneously with the determination of error type described, the entire error correction sequence (using a toggling fetched data latch) takes less than 20 ns. This is because an error occurrence detector is a simple ORing of S1 to S8. The error locator has simultaneously located the error which is then corrected as through the error was a single (and therefore correctable) error. The parity of syndromes then determines if the error was indeed single, and interrupts the CPU if the error was an uncorrectable (multiple) error. Since uncorrec-

table data is unusable without special handling, the CPU would be interrupted anyway; therefore this automatic correction of any error as if it were single does not create any problems. This fast error correction technique allows single erorr correction on a non-interrupt basis with only a 20 ns memory system access time penalty.

These techniques can, of course, be extended to large or smaller data words.

FIGURE 4 – M2 PATTERN BUILDING BLOCK

S1 =	P10	P11	P12	P13	P54	P55	P56	B(64)
S2 =	P20	P21	P22	P23	P54	P55	P57	B(65)
S3 =	P30	P31	P32	P33	P54	P56	P57	B(66)
S4 =	P40	P41	P42	P43	P55	P56	P57	B(67)
S5 =	P14	P15	P16	P17	P50	P51	P52	B(68)
S6 =	P24	P25	P26	P27	P50	P51	P53	B(69)
S7 =	P34	P35	P36	P37	P50	P52	P53	B(70)
S8 =	P44	P45	P46	P47	P51	P52	P53	B(71)

Where for P_{NM}: N = MC10193 Output
M = Byte Number

FIGURE 5 – M2 PATTERN CORRECTION MATRIX

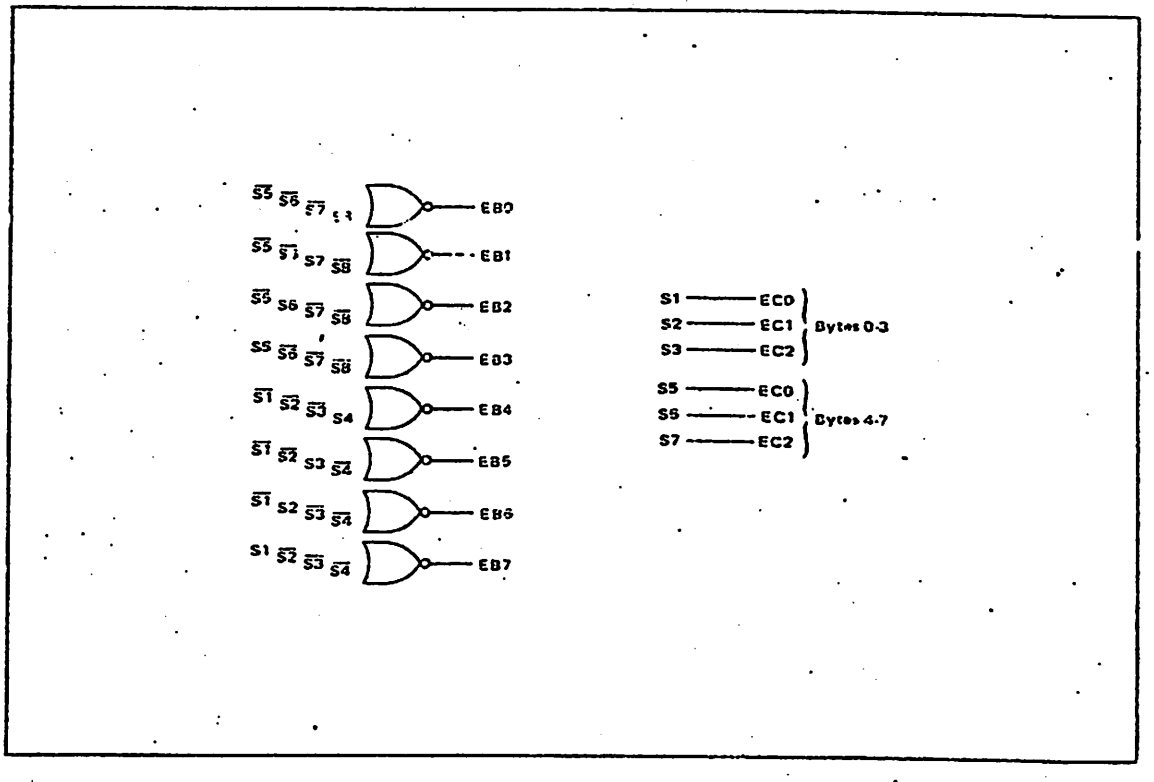
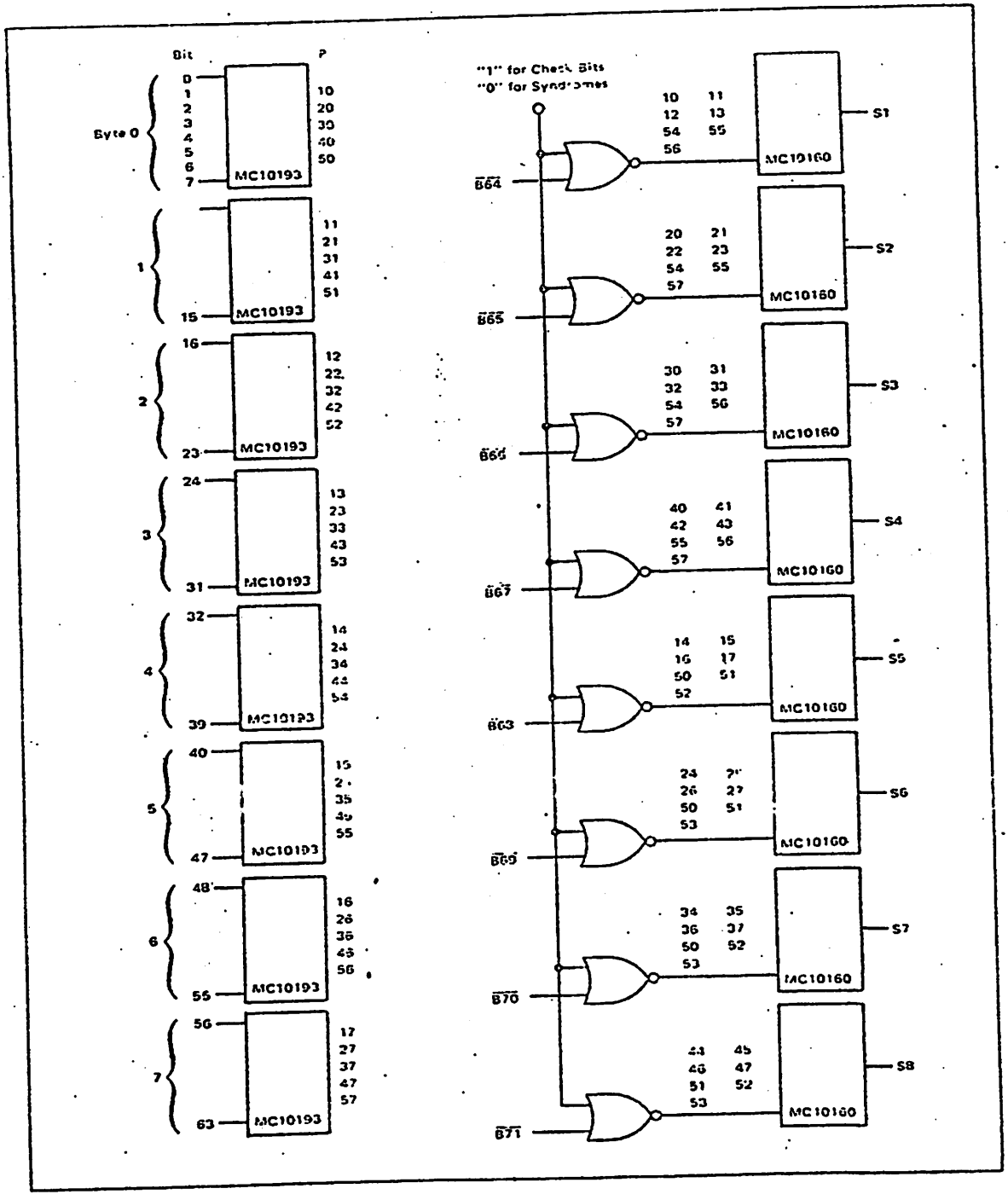


FIGURE 6 – SYNDROME AND CHECK BIT GENERATOR, M2 PATTERN



APPENDIX B

Evaluation of high level modules

The following tables are obtained by going through the S-1 Engineeing Drawings, the output of SCALD, the VAX blueprints, and the VAX CPU Technical Description. First, I located the various modules used in both machines. Then I compiled a part list for each occurrence of the five high level modules. Finally, I multiply the number of a particular integrated circuits used by the number of transistors in that chip to get the total transistor count.

High Level Modules in S-1

Module	Part #	# chips	% S-1	# Tr	% S-1
Cache	10101	4		112	
	10109	32		608	
	10113	68		4896	
	10145A	184		96416	
	10160	10		1000	
	10170	10		630	
	2110-1	296		1285824	
	MB7042	156		244608	
		760	13.28	1634094	27.30
ALU	10145A	194		101656	
	10158	2		136	
	10159	19		1292	
	10164	36		2736	
	10173	66		7128	
	10174	54		2754	
	10176	4		488	
	10179	11		803	
	10180	21		1491	
	10181	138		52716	
		545	9.53	171200	2.86
Shifter	10174	141	2.46	7191	0.12

Module	Part #	# chips	% S-1	# Tr	% S-1
Microsequencer and Control Store	10016	8		1944	
	10105A	2		42	
	10136	2		616	
	10145A	24		12576	
	10158	11		748	
	10161	2		106	
	10164	28		2128	
	10173	2		216	
	10174	4		204	
	10176	6		732	
	2110-1	688		2988672	
	MB7042	24		37632	
		801	14.00	3045616	50.89
PLA	AND/NAND	95		2848	
	OR/NOR	1296		28711	
	INVERT	12		288	
	AND-OR-INVERT	57		1554	
		1460	25.52	33401	0.56

High Level Modules in VAX

Module	Part #	# chips	% VAX	# Tr	% VAX
Cache	74S04	2		48	
	74S64	9		108	
	74S257	9		396	
	74S280	32		4608	
	74S373	3		576	
	8209	12		29472	
	93425A	114		495216	
	DC102	12		4272	
		193	7.74	534696	36.62
ALU	74S151	32		1664	
	74S153	23		1472	
	74S157	3		132	
	74S181	13		4030	
	74S182	5		250	
	74S194	4		624	
	74S283	9		1476	
	85S68	24		14784	
		113	4.53	24432	1.67
Shifter	74S153	29		1856	
	74S157	2		88	
	74S257	8		352	
	25S10	67		5628	
		106	4.25	7924	0.54

Module	Part #	# chips	% VAX	# Tr	% VAX
Microsequencer and Control Store	74S151	3		156	
	74S153	5		320	
	74S157	1		44	
	74S175	4		336	
	74S194	1		156	
	74S240	33		1980	
	74S251	1		52	
	74S253	6		288	
	74S280	15		2160	
	74S283	2		328	
	85S68	4		2464	
	93425A	99		430056	
	PROM (256 x 8)	100		259200	
		274	11.00	697540	47.78
PLA	AND/NAND	277		4208	
	OR/NOR	37		938	
	INVERTER	151		3600	
	AND-OR-INVERT	167		1999	
		632	25.36	10745	0.73

Acknowledgement

This work was supported by the Defense Advanced Research Projects Agency (DoD), ARPA Order 3803, monitored by the Naval Electronic System Command under Contract N00039-78-C-0013-0004.