Every effort has been made to supply complete and accurate information. However, Sybex assumes no responsibility for its use; nor any infringements of patents or other rights of third parties which would result. No license is granted by the equipment manufacturers under any patent or patent rights. Manufacturers reserve the right to change circuitry at any time without notice.

Copyrighted material reprinted by permission.
INTRODUCTION

1. BRIEF HISTORY OF CPU DESIGN

2. BIT-SLICE PRINCIPLES

3. BIT-SLICE IN DETAIL - BUILDING WITH BIT-SLICE DEVICES

4. OTHER BIT-SLICE DEVICES

5. BIT-SLICE APPLICATIONS

6. DEVELOPMENT AIDS

7. CONCLUSION

8. APPENDICES

SYBEX
0. INTRODUCTION
INTRODUCTION

THE PURPOSES OF THIS COURSE:

1. TO EXPLAIN:
   - WHAT BIT-SLICE DEVICES DO
   - WHY BIT-SLICE DEVICES EXIST

2. TO DEMONSTRATE IN DETAIL THE PROCEDURE FOR
   DESIGNING WITH BIT-SLICE DEVICES

3. TO SURVEY THE BIT-SLICE DEVICES ON THE MARKET

4. TO SURVEY THE RANGE OF APPLICATIONS OF BIT-SLICE DEVICES.

SYBEX
BASIC CONCEPTS

This course explores topics in these areas:

1. Central processor architectures
2. Programming and microprogramming
3. Large scale integration (LSI)
4. Bit-slice architecture
5. Programmed logic arrays (PLA's)
6. Computer arithmetic
7. Emulation
8. Special function processor architectures

SYBEX
DEFINITIONS

CENTRAL PROCESSOR: THE PART OF A COMPUTER THAT TRANSFORMS DATA, MAKES DECISIONS.
- USUALLY COMPRISSES REGISTERS, ARITHMETIC-LOGICAL UNIT, AND CONTROL LOGIC

PROGRAM: THE DATA WHICH ULTIMATELY CONTROLS THE ACTIONS OF THE CENTRAL PROCESSOR
- USUALLY COMPRISSES SEQUENCE OF INSTRUCTION WORDS IN MEMORY

MICROPROGRAM: A FIXED SEQUENCE OF INSTRUCTION WORDS WHICH CONTROL THE GATES OF A PROCESSOR. A MICROPROGRAM SEQUENCE IS INVOKED BY A SINGLE INSTRUCTION OF THE (MACRO)PROGRAM.
RELATIONSHIP OF PROGRAM, MICROPROGRAM, AND CPU

PROGRAM

A-TYPE INSTRUCTION
(MAIN PROGRAM)

MICROSEQUENCE A

MICROPROGRAM

B-TYPE INSTRUCTION

MICROSEQUENCE B

A-TYPE INSTRUCTION

CENTRAL PROCESSING UNIT

RAM

RAM

ALU

SYBEX
DEFINITIONS (CONTINUED)

LARGE SCALE INTEGRATION: THE TECHNOLOGY OF ASSEMBLING CIRCUITS OF ONE THOUSAND OR MORE GATES ON A SINGLE CHIP (E.G., MICROCOMPUTERS, ROMS & RAMS, BIT-SLICE DEVICES)

BIT-SLICE ARCHITECTURE: THE INTERCONNECTIONS OF LOGICAL ELEMENTS WITHIN THE BIT-SLICE CHIP

PROGRAMMED LOGIC ARRAY: AN LSI DEVICE FOR REPLACING DISCRETE LOGIC NETWORKS BY PERFORMING THEIR LOGICAL EQUIVALENTS THROUGH MATRIX MAPPING

COMPUTER ARITHMETIC: THE ALGORITHMS (PROGRAM STEPS) BY WHICH COMPUTERS PERFORM NUMERICAL CALCULATIONS
EMULATION: THE PROGRAMMING OF A COMPUTER TO MAKE IT INTERPRET THE INSTRUCTION SET OF ANOTHER COMPUTER.

- MOST OFTEN: MICROPROGRAMMED EMULATION - A MAJOR APPLICATION OF BIT-SLICE DEVICES

SPECIAL FUNCTION PROCESSES: COMPUTERS DESIGNED EXCLUSIVELY FOR SPECIFIC APPLICATIONS (E.G., I/O DEVICE CONTROLLERS, TEXT HANDLERS)

SYBEX
1. BRIEF HISTORY OF CPU DESIGN
CPU DESIGN EVOLUTION

BIT - SLICE TECHNOLOGY

A STEP IN CPU DESIGN EVOLUTION?

OR

A SEMICONDUCTOR INDUSTRY BY - PRODUCT?
DEFINITIONS -
HORIZONTAL STRUCTURES ARE CIRCUITS FORMING THE DATA WORD CONTROLLING REGISTERS ADDERS COMPARATORS MEMORY REGISTERS

VERTICAL STRUCTURES ARE CIRCUITS FORMING THE DATA PATH CONTROLLING BUSSES ADDRESS SELECTION MULTIPLEXING

SYBEX
EVOLUTION

CPU DESIGN HAS EVOLVED FROM

HORIZONTAL DEVELOPMENT

to

VERTICAL DEVELOPMENT

SYBEX
HORIZONTAL MACHINE ORGANIZATION

SYBEX
EVOLUTION

CPU DESIGN FOR MATHEMATICAL APPLICATIONS
EMPHASIZED HORIZONTAL ENHANCEMENTS

1. WIDER DATA WORDS
2. POWERFUL ARITHMETIC FUNCTIONS
3. FASTER ALU PROPAGATE TIMES
   TO COMPENSATE FOR WIDER DATA

VERTICAL ORGANIZATION IS THE SAME IN MOST MACHINES
FROM PDP - 1 TO CDC 7600

SYBEX
EVOLUTION

NEW APPLICATIONS BROUGHT NEW CPU DESIGN PHILOSOPHY

DATA STRUCTURE APPLICATIONS

FILE SYSTEMS
ASSEMBLERS
COMPILERS
TEXT EDITORS
COMMUNICATION SYSTEMS
NON-ARITHMETIC LANGUAGES: LISP
EVOLUTION

DATA STRUCTURE PROBLEMS ARE

WORD ORDER TRANSFORMATIONS...

WORD ORDER TRANSFORMATIONS ARE ACCOMPLISHED BY

DATA PATH MANIPULATIONS...

THEY ARE A FUNCTION OF THE VERTICAL STRUCTURE OF THE CPU

SYBEX
EVOLUTION

EXAMPLES OF DATA STRUCTURING PROBLEMS:

PARSING
FILE MOVEMENT
FILE EDITING
LINK - LIST BUILDING
STACK MANIPULATION
STRING EDITING

SYBEX
EVOLUTION

FUNCTIONAL DEMANDS CHANGED →

CPU STRUCTURAL DESIGN EVOLVED

HORIZONTAL → VERTICAL

EXAMPLE: FORTRAN → LISP

IBM 7090 → DEC PDP - 6
PDP - 6 FEATURES

DOUBLE ADDRESSING
(REGISTER TO REGISTER OPERATIONS)

20\textsubscript{B} ACCUMULATORS

AUTOMATED STRING MOVES
(VARIABLE LENGTH BYTE OPERATIONS)

PDP - 6 ARCHITECTURE IS PRECURSOR OF BIT SLICE ARCHITECTURE
EVOLUTION - MACHINE POPULATION

SATURATION OF "NUMBER CRUNCHER" MARKET
GROWTH OF MINI-COMPUTER MARKET
RAPID ENHANCEMENT OF MINI-COMPUTERS
  1. AUGMENTED REGISTER ADDRESSING (REG-TO-REG OPS)
  2. AUGMENTED BYTE AND VECTOR HANDLING (BLTR)
  3. CONTEXT SWITCHING
  4. FASTER CYCLE TIMES FOR NON ARITHMETIC TRANSFER

BUT LITTLE OR NO ARITHMETIC AUGMENTATION
STANDARDIZED LENGTH 12 OR 16 BITS
EVOLUTION: SUMMARY

THE COMPUTER MARKET NOW DEMANDS VERTICAL POWER AND FLEXIBILITY AT THE EXPENSE OF HORIZONTAL POWER.

<table>
<thead>
<tr>
<th>VERTICAL</th>
<th>HORIZONTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGISTER SPACE</td>
<td>WORD WIDTH</td>
</tr>
<tr>
<td>COMPREHENSIVE BUSSING</td>
<td>ARITHMETIC FUNCTIONS</td>
</tr>
<tr>
<td>ITERATIVE DATA TRANSFERS</td>
<td>FAST PROPAGATE TIMES</td>
</tr>
<tr>
<td>TRANSFER CYCLE TIME</td>
<td></td>
</tr>
</tbody>
</table>
2. BIT-SLICE PRINCIPLES
BIT SLICE PRINCIPLES

LARGE SCALE INTEGRATION BROUGHT TWO DEVELOPMENTS

1. TOTALLY INTEGRATED MICRO COMPUTERS
2. INTEGRATED BLOCKS OF ARCHITECTURE FOR LARGER COMPUTERS

LIMITATIONS OF CURRENT LSI

1. LOWER DENSITY FOR HIGH SPEED LOGIC
2. HIGHER DISSIPATION FOR HIGH SPEED LOGIC
3. PIN COUNT LIMITS
WHY THE BIT SLICE?

LSI DESIGNERS HAD TWO CHOICES

HORIZONTAL PARTITIONING
WIDE REGISTERS
WIDE MULTIPLEXERS
WIDE ALUs

VERTICAL PARTITIONING
BIT WIDE SLICES OF THE COMPLETE DATA PATH OF CPU

SYBEX
WHY THE BIT SLICE?

HORIZONTAL LSI COMPONENTS (HYPOTHETICAL)

1. 16 BIT DUAL PORT REGISTER

~100 GATES
48 DATA PINS
3 CONTROL PINS
2 POWER PINS

2 GATES PER PIN

53 PINS

SYBEX
WHY THE BIT SLICE?

OTHER HYPOTHETICAL HORIZONTAL LSI COMPONENTS

2. 16 BIT FAST ALU  ~400 GATES / 53 PINS: ~8 GATES / PIN
3. 16 BIT  3:1 MULTIPLEXER  <100 GATES / 68 PINS:  1½ GATES / PINS

THESE ARE NOT TECHNICALLY LSI

HORIZONTAL STRUCTURES ARE PIN LIMITED TO MSI

SYBEX
HYPOTHETICAL ONE-BIT-WIDE SLICE
WHY THE BIT SLICE?

ASSUME 16 REGISTERS:

300 GATES / 23 PINS
OR
13 GATES PER PIN

MULTIPLYING SLICES ADDS 2 PINS PER SLICE:

EXTERNAL INPUT, EXTERNAL OUTPUT

MULTIPLYING SLICES ADDS 200 GATES PER SLICE

EXAMPLE 8 BIT SLICE 1700 GATES
39 PINS 43 GATES / PIN

SYBEX
# OTHER FACTORS FAVORING VERTICAL INTEGRATION

| Speed: | External connections increase  
CAPACITANCE \( \rightarrow \) decrease speed |
|---|---|
| Dissipation: | \( T^2 L \) compatible drive required external  
To chip -- 10mW (5V 2mA)  
Internal dissipation per gate typically  
1.2 - 1.4mW: Low power SHOTTKY  
100 microwatts: \( I^2 L \) (TI SBP0400) |
| Bit Slice Strategy: | Maximize internal vertical interconnections  
At the expense of external horizontal interconnections |
BIT-SLICE DEVICES

1974: THE COMMERCIAL BIRTH YEAR OF BIT-SLICE

NATIONAL SEMICONDUCTOR         IMP SERIES
INTEL                            3000 SERIES

SINCE THEN

MONOLITHIC MEMORIES     5700/6700 SERIES
ADVANCED MICRO DEVICES   2900 SERIES
MOTOROLA 10800
TEXAS SBP0400

AND SECOND SOURCES IN U.S. AND EUROPE

SYBEX
BIT SLICE APPLICATIONS

BIT SLICE DEVICES ARE TO DESIGNERS IN 1976 WHAT MSI WAS TO DESIGNERS IN 1973

BIT SLICE DEVICES ARE FOUND IN PRODUCTION MODELS OF PROTOTYPES OF

POPULAR MINI COMPUTERS
FLOATING POINT ADD-ONS
HIGH SPEED I/O ADAPTERS: DISK & COMMUNICATIONS
SPECIAL FUNCTION BOXES: FFT, NAVIGATION
SPECIAL ARCHITECTURES: DISTRIBUTED PROCESSOR DESIGNS

SYBEX
A BIT SLICE DEVICE IS NOT A MICROPRESSOR

<table>
<thead>
<tr>
<th></th>
<th>BIT-SLICE</th>
<th>LSI MICRO</th>
<th>BIT-SLICE ADD-ONs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SHIFT</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>REGISTERS</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>REGISTER SELECTION</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>INPUT TO BUS</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OUTPUT FROM BUS</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MICRO INSTRUCTION SEQUENCING</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MEMORY ADDRESSING</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ALU + SHIFT CONTROL</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>I/O CONTROL</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

SYBEX
BIT SLICE PRINCIPLES: SUMMARY

BIT-SLICE DEVICES

ARE VERTICAL STRUCTURES:
2 OR 4/NTHS OF THE DATA PATH.
CIRCUITRY OF AN N-BIT PROCESSOR

HAVE NO CONTROL LOGIC -
CONTROL SIGNALS ARE SUPPLIED FROM EXTERNAL CONTROLLER CHIPS

ARE INTENDED FOR FAST APPLICATIONS
CYCLE SPEEDS BETTER THAN SHOTTKY MSI EQUIVALENTS

ARE NOT INTENDED FOR MINIMUM COST APPLICATIONS
MUCH EXTERNAL LOGIC REQUIRED
BIT-SLICE SYSTEM

DATA (IN)

ADDRESS

DATA (OUT)

INSTRUCTION REGISTER

DECODER

CONTROL UNIT

MICROPROGRAM

N-BIT SLICE (1)

N-BIT SLICE (2)

N-BIT SLICE (P)

CONTROL

N x P BIT-ALU

SYBEX
3. BIT-SLICE IN DETAIL:
BUILDING WITH BIT-SLICE DEVICES
A BIT SLICE DEVICE IN DETAIL

ADVANCED MICRO DEVICES 2901

FEATURES

4 BIT SLICE
16 WORD TWO PORT RAM
8 FUNCTION ALU
INDEPENDENT SHIFT
FOUR STATUS FLAGS
9-BIT CONTROL MICRO-INSTRUCTION

BIPOLAR: LOW POWER SHOTTKY IMPLEMENTATION
MIN 105 NS CLOCK FOR COMMERCIAL VERSION
AMD 2901 BLOCK DIAGRAM
**ALU INPUT SOURCES**

<table>
<thead>
<tr>
<th>R BUS</th>
<th>ABR</th>
<th>S BUS</th>
<th>ABR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. RAM A-LATCH</td>
<td>A</td>
<td>1. RAM A-LATCH</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1 OF 16 REGISTERS)</td>
<td></td>
</tr>
<tr>
<td>2. DIRECT DATA INPUTS</td>
<td>D</td>
<td>2. RAM B-LATCH</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1 OF 16 REGISTERS)</td>
<td></td>
</tr>
<tr>
<td>AND INHIBIT</td>
<td>Ø</td>
<td>3. Q - REGISTER</td>
<td>Q</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(FOR ON BUS)</td>
<td>Ø</td>
</tr>
</tbody>
</table>

SYBEX
AMD 2901 - BUS STRUCTURE

ALU OUTPUT - BUS DESTINATIONS

1. CHIP OUTPUT (Y BUS) MULTIPLEXER
2. Q REGISTER MULTIPLEXER
3. SHIFTER/RAM MULTIPLEXER

OTHER PATHS

1. A-LATCH TO Y-MULTIPLEXER
2. Q-REGISTER OUTPUT TO Q REGISTER MULTIPLEXER

INPUTS
\begin{align*}
\text{SHIFT LEFT} \\
\text{SHIFT RIGHT}
\end{align*}

SYBEX
AMD 2901 - FLAGS AND STATUS OUTPUT

OTHER DATA OUTPUT

1. CARRY GENERATE G
2. CARRY PROPAGATE P
3. CARRY OUT C
4. OVERFLOW (Cn+3 ← Cn+4) OVR
5. F-BUS = 0 F = 0
6. F-BUS MSB (SIGNBIT) F3
7. RAM SHIFT LEFT OUT (SHARED)
8. RAM SHIFT RIGHT OUT (SHARED)
9. Q-REG SHIFT LEFT OUT (SHARED)
10. Q-REG SHIFT RIGHT OUT (SHARED)
OTHER DATA INPUT

1. CARRY IN Cn

2. RAM SHIFT LEFT IN (SHARES WITH RAM SHIFT RIGHT OUT)

3. RAM SHIFT RIGHT IN (SHARES WITH RAM SHIFT LEFT OUT)

4. Q REG SHIFT LEFT IN (SHARES Q SHIFT RIGHT OUT)

5. Q REG SHIFT RIGHT IN (SHARES Q SHIFT LEFT OUT)

SYBEX
AMD 2901 - CHIP CONTROL

AREAS OF CONTROL

1. ALU INPUTS R AND S
2. ALU FUNCTIONS
3. Y BUS MUX, Y BUS OUTPUT ENABLE
4. RAM SHIFT MULTIPLEXER
5. RAM A ADDRESS
6. RAM B ADDRESS
7. Q-REGISTER MULTIPLEXER

PLUS CLOCK

SYBEX
AMD 2901 - CONTROL WORD

CONTROL WORD FORMAT 9 BIT

<table>
<thead>
<tr>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESTIN.</td>
<td>F(ALU)</td>
<td>SOURCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. ALU SOURCE CONTROL

<table>
<thead>
<tr>
<th>I_2</th>
<th>I_1</th>
<th>I_0</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Q</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Q</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ø</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>D</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>D</td>
<td>Q</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>D</td>
<td>Ø</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SYBEX
### AMD 2901 - CONTROL WORD

#### 2. ALU FUNCTION CONTROL

<table>
<thead>
<tr>
<th>$I_5\ I_4\ I_3$</th>
<th>ALU FUNCTION</th>
<th>SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R PLUS S</td>
<td>R + S</td>
</tr>
<tr>
<td>1</td>
<td>S MINUS R</td>
<td>S - R</td>
</tr>
<tr>
<td>2</td>
<td>R MINUS S</td>
<td>R - S</td>
</tr>
<tr>
<td>3</td>
<td>R OR S</td>
<td>R $\lor$ S</td>
</tr>
<tr>
<td>4</td>
<td>R AND S</td>
<td>R $\land$ S</td>
</tr>
<tr>
<td>5</td>
<td>R AND S</td>
<td>R $\land$ S</td>
</tr>
<tr>
<td>6</td>
<td>R X OR S</td>
<td>R $\lor$ S</td>
</tr>
<tr>
<td>7</td>
<td>R X NOR S</td>
<td>(R $\lor$ S)</td>
</tr>
</tbody>
</table>
# AMD 2901 - CONTROL WORD

## 3. DESTINATION CONTROL

<table>
<thead>
<tr>
<th>I_a,7,e</th>
<th>RAM SHIFT</th>
<th>RAM LOAD</th>
<th>Q SHIFT</th>
<th>Q LOAD</th>
<th>Y OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>NO</td>
<td>F_{a-3}</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>NO</td>
<td>F_{a-3}</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NO</td>
<td>F_{a-3}</td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NO</td>
<td>F_{a-3}</td>
<td></td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>LEFT</td>
<td>F_{a-3}, RLI</td>
<td>LEFT</td>
<td>Q_{a-3}, QLI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F_a \rightarrow RLO)</td>
<td>(Q_a \rightarrow QLO)</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LEFT</td>
<td>F_{a-3}, RLI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F_a \rightarrow RLO)</td>
<td></td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>RIGHT</td>
<td>F_{a-2}, RRI</td>
<td>RIGHT</td>
<td>Q_{a-2}, QRI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F_2 \rightarrow RRO)</td>
<td>(Q_2 \rightarrow QRO)</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RIGHT</td>
<td>F_{a-2}, RRI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F_3 \rightarrow RRO)</td>
<td></td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

**SYBEX**
AMD 2901 - ALU FUNCTIONS

By control of $I_5 - I_{15}$ and $CN$, the AM 2901 performs the following arithmetic functions in one step.

<table>
<thead>
<tr>
<th>$I_5-3, I_{2-0}$</th>
<th>$CN = 0$</th>
<th>$CN = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ADD A + Q</td>
<td>ADD PLUS ONE A + Q + 1</td>
</tr>
<tr>
<td>1</td>
<td>A + B</td>
<td>A + B + 1</td>
</tr>
<tr>
<td>5</td>
<td>D + A</td>
<td>D + A + 1</td>
</tr>
<tr>
<td>6</td>
<td>D + Q</td>
<td>D + Q + 1</td>
</tr>
<tr>
<td>2</td>
<td>PASS Q</td>
<td>INCREMENT Q + 1</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>B = 1</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>A = 1</td>
</tr>
<tr>
<td>7</td>
<td>D</td>
<td>D = 1</td>
</tr>
<tr>
<td>1 2</td>
<td>DECREMENT Q - 1</td>
<td>PASS Q</td>
</tr>
<tr>
<td>1 3</td>
<td>B - 1</td>
<td>B</td>
</tr>
<tr>
<td>1 4</td>
<td>A - 1</td>
<td>A</td>
</tr>
<tr>
<td>2 7</td>
<td>D - 1</td>
<td>D</td>
</tr>
</tbody>
</table>

SYBEX
### AMD 2901 - ALU FUNCTIONS CONTINUED

<table>
<thead>
<tr>
<th>(I_{5-3}, I_{2-0})</th>
<th>(C_N)</th>
<th>(C_N = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 2</td>
<td>ONES COMPL.</td>
<td>- Q - 1</td>
</tr>
<tr>
<td>2 3</td>
<td></td>
<td>TWOS COMP. -Q</td>
</tr>
<tr>
<td>2 4</td>
<td></td>
<td>(NEGATE) -B</td>
</tr>
<tr>
<td>1 7</td>
<td></td>
<td>B - 1</td>
</tr>
<tr>
<td>1 0</td>
<td>SUBTRACT</td>
<td>Q - A - 1</td>
</tr>
<tr>
<td>1 1</td>
<td>ONES COMPL.</td>
<td>B - A - 1</td>
</tr>
<tr>
<td>1 5</td>
<td></td>
<td>A - D - 1</td>
</tr>
<tr>
<td>1 6</td>
<td></td>
<td>Q - D - 1</td>
</tr>
<tr>
<td>2 0</td>
<td></td>
<td>A - Q - 1</td>
</tr>
<tr>
<td>2 1</td>
<td></td>
<td>A - B - 1</td>
</tr>
<tr>
<td>2 5</td>
<td></td>
<td>D - A - 1</td>
</tr>
<tr>
<td>2 6</td>
<td></td>
<td>D - Q - 1</td>
</tr>
</tbody>
</table>
AMD 2901 CONTROL

9 BIT CONTROL WORD PROVIDES ALL CHIP CONTROL
EXCEPT RAM ADDRESSING

RAM ADDRESSING

WRITE (B)  B-BUS  A-BUS  READ
AMD 2901 TIMING

Timing is provided by a single clock 105ns

- \( \geq 105 \text{ ns} \)
- \( >30 \text{ ns} \)
- \( \geq 30 \text{ ns} \)

- \( C_p \) < 30
- RAM OUT GOOD
- LATCH OUTPUT GOOD < 40
- 50 - 85ns
- ALU DATA AVAILABLE
- RAM WRITE ENABLE
- Q REG WRITE ENABLE
- Q REG DATA GOOD

* Depends on function and data
AMD 2901: TIMING CONTINUED

EXTERNAL SIGNAL TIMING REQUIREMENTS

CP
Ao-3
Bo-3
Io-8
Do-3
Cn

• RAM ADDRESSES ARE THE CRITICAL TIMING SIGNAL

SYBEX
### AMD 2901: TIMING TABLES

#### CYCLE TIME AND CLOCK CHARACTERISTICS

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Am2901DC</th>
<th>Am2901DM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Read Modify-Write Cycle</td>
<td>105 ns</td>
<td>120 ns</td>
</tr>
<tr>
<td>Time from selection of A, B registers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>to end of cycle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Clock Frequency to Shift Q</td>
<td>9.5 MHz</td>
<td>8.3 MHz</td>
</tr>
<tr>
<td>Register (50% duty cycle)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum Clock LOW Time</td>
<td>30 ns</td>
<td>30 ns</td>
</tr>
<tr>
<td>Minimum Clock HIGH Time</td>
<td>30 ns</td>
<td>30 ns</td>
</tr>
<tr>
<td>Minimum Clock Period</td>
<td>105 ns</td>
<td>120 ns</td>
</tr>
</tbody>
</table>

#### GUARANTEED OPERATING CONDITIONS

Tables I, II, and III define the timing requirements of the Am2901 in a system. The Am2901 is guaranteed to function correctly over the operating range when used within the delay and setup time constraints of these tables for the appropriate device type. The tables are divided into three types of parameters: clock characteristics, combinational delays from inputs to outputs, and setup and hold time requirements. The latter table defines the time prior to the end of the cycle (i.e., clock LOW-to-HIGH transition) that each input must be stable to guarantee that the correct data is written into one of the internal registers. The performance of the Am2901 within the limits of these tables is guaranteed by the testing defined as "Group A, Subgroup 9" Electrical Testing. For a copy of the tests and limits used for subgroup 9, contact Advanced Micro Devices' Product Marketing.

#### TABLE II

**MAXIMUM COMBINATIONAL PROPAGATION DELAYS** *(all in ns, CL ≤ 15pF)*

<table>
<thead>
<tr>
<th>From Input</th>
<th>Am2901DC</th>
<th>Am2901DM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>A, B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1345</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1678</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OE Enable/Disable</td>
<td>40/25</td>
<td></td>
</tr>
<tr>
<td>A Latching</td>
<td>ALU (1 = 2xx)</td>
<td>80</td>
</tr>
</tbody>
</table>

#### TABLE III

**SET-UP AND HOLD TIMES** *(minimum cycles from each input)*

Set-up and hold times are defined relative to the clock LOW-to-HIGH edge. Inputs must be steady at all times from the set-up time prior to the clock until the hold time after the clock. The set-up times allow sufficient time to perform the correct operation on the correct data so that the correct ALU data can be written into one of the registers.

<table>
<thead>
<tr>
<th>From Input</th>
<th>Notes</th>
<th>Am2901DC</th>
<th>Am2901DM</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B Source</td>
<td>2, 3, 4</td>
<td>105</td>
<td>120</td>
</tr>
<tr>
<td>B Dest.</td>
<td>2, 4</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>D</td>
<td>100</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>CN</td>
<td>55</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>'012</td>
<td>85</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>'345</td>
<td>70</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>'878</td>
<td>4</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>RI, LI (RAM or Q)</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

**Notes:**
1. See Figure 11 and 12.
2. If the B address is used as a source operand, allow for the "A, B source" set-up time. If it is used only for the dest-ination address use the "B dest." set-up time.
3. Where two numbers are shown, both must be met.
4. "lpwl." is the clock LOW time.
AMD 2901: OTHER DATA

**E L E C T R I C A L**

1. **5.0 V O L T S U P P L Y**
2. **185 M A C U R R E N T T Y P I C A L - 280 M A X (1.4 W A T T S)**

**T E M P E R A T U R E**

**C O M M E R C I A L** 0°C TO +70°C / 4.75V TO 5.25V / 105ns

**M I L I T A R Y** -55°C TO +125°C / 4.50 TO 5.50V / 120ns

**P H Y S I C A L**


2.000 I N C H E S M Y .580 I N C H E S
BIT SLICE APPLICATION: CASCADING 2901s

SIMPLE CASCADING

RIPPLE CARRY
SHIFT OFF ENDS OR END AROUND SHIFT

EXAMPLE: 16 BIT CPU

RS'16  \[ S \]
QS'16  \[ C \]

\[ \begin{array}{c}
\text{ABI} \\
\text{CL}_6 \\
\text{CO}_1 \\
\text{CO}_0 \\
\text{RS} \\
\text{QS} \\
\text{ABI} \\
\text{Cl}_1 \\
\text{Cl}_0 \\
\end{array} \]

\[ \begin{array}{c}
\text{DO} - 3 \\
\text{D4} - 7 \\
\text{D8} - 11 \\
\text{D12} - 15 \\
\end{array} \]

\[ \begin{array}{c}
\text{Y0} - 3 \\
\text{Y4} - 7 \\
\text{Y8} - 11 \\
\text{Y12} - 15 \\
\end{array} \]

\[ \begin{array}{c}
\text{RS} = \text{RAM L/O/R I} \\
\text{RS'} = \text{RAM R/O/L I} \\
\text{QS} = \text{QREG L/O/R I} \\
\text{QS'} = \text{QREG R/O/L I} \\
\text{ABI} = A0-3 B0-3 I0-8 \\
\end{array} \]

SYBEX
SIMPLE CASCADE — TIMING IMPLICATIONS

CLOCK RATE MUST ACCOMODATE WORST CASE TIMING
FULL 16 BIT RIPPLE CARRY

CRITICAL TIMES

<table>
<thead>
<tr>
<th>TIME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cn + 4</td>
<td>100ns</td>
</tr>
<tr>
<td>Cn + 4</td>
<td>80ns</td>
</tr>
<tr>
<td>Cn + 4</td>
<td>30ns</td>
</tr>
<tr>
<td>RAM RO/LO (LAST OUTPUT)</td>
<td>55ns</td>
</tr>
</tbody>
</table>

SYBEX
AMD 2901 SIMPLE CASCADE TIMING

SHIFT OF END - CLOCK RATE

REGISTER SET UP + CARRY DELAYS + LAST OUTPUT DELAY

20 + 140 + 55 = 215ns

END AROUND SHIFT

ADD SHIFT IN + GATING

215 + 30 + 15 = 260ns

MINIMUM CLOCK FOR A PRACTICAL 16-BIT RIPPLE CPU

CP = 260ns

SYBEX
AMD 2901: LOOK AHEAD CASCADE

GENERATE AND PROPAGATE OUTPUTS

DESIGNED FOR STANDARD LOOK AHEAD
AM 2902/ 74S182

SIMULTANEOUS WITH CN+4 OUTPUT

CONFIGURATION FOR 16 BIT CPU

SYBEX
CRITICAL TIMES

CI TO AM2902 C, C₁, C₂, C₃ = 10ns
ABI TO G,P = 100ns
CLK TO G,P = 80ns
CI TO RAM RO/LO = 55ns
SHIFT IN SETUP = 30ns
AMD 2901 LOOK AHEAD CASCADE TIMING

MINIMUM CLOCK WITH LOOK - AHEAD/ WITHOUT SHIFT AROUND

\[ CP = 165\text{ns} \]

MINIMUM CLOCK WITH LOOK - AHEAD AND SHIFT AROUND

\[ CP = 210\text{ns} \]

LOOK AHEAD CIRCUITRY GAINS 50\text{ns} PER CYCLE FOR 16 BIT CPU

GAINS INCREASE AS WORD WIDTH INCREASES

EXAMPLE: 48 BIT RIPPLE \[ CP = 500\text{ns} \]

48 BIT LOOK - AHEAD \[ CP = 240\text{ns} \]

SYBEX
### AMD 2901 Ripple Versus Look-Ahead

<table>
<thead>
<tr>
<th></th>
<th>Ripple</th>
<th>Look-Ahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 - Bit Simple Shift</td>
<td>215</td>
<td>165</td>
</tr>
<tr>
<td>16 - Bit End Around</td>
<td>260</td>
<td>210</td>
</tr>
<tr>
<td>48 - Bit Simple Shift</td>
<td>455</td>
<td>195</td>
</tr>
<tr>
<td>48 - Bit End Around</td>
<td>500</td>
<td>240</td>
</tr>
</tbody>
</table>

**Formulae for n Bit Simple Shift**

Ripple: \( CP = 155 + 30 \left( \frac{N}{4} - 2 \right) \) ns

Look Ahead: \( CP = 155 + 10L \) ns

Where \( L = \) length of 74S182's cascade

Add 30 plus gating for end around shift

**SYBEX**
USING THE BIT-SLICE DEVICE

AN ARRAY OF BIT SLICE DEVICES FORMS A PROCESSING UNIT.

AN ALGORITHM IS A SEQUENCE OF STEPS PERFORMED BY THE PROCESSING UNIT.

ALGORITHMS MAY ORIGINATE FROM

MACRO-PROGRAMS
MICRO-PROGRAMS
LOGIC

THE INSTRUCTION AND REGISTER ADDRESS LOGIC OF THE BIT SLICE ARRAY

DETERMINE THE FINAL FORMAT OF CONTROL FOR A GIVEN ALGORITHM.
A MULTIPLY ALGORITHM FOR A 16-BIT 2901 CPU

CONFIGURATION - 16 BITS (WITH OR WITHOUT LOOKAHEAD)
# MULTIPLY ALGORITHM

<table>
<thead>
<tr>
<th>EXIT</th>
<th>SOURCE</th>
<th>FUNCTION</th>
<th>DESTINATION</th>
<th>A</th>
<th>B</th>
<th>LOAD MULTIPLIER R 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>J→D</td>
<td>D, 0</td>
<td>D V 0</td>
<td>B</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K→D</td>
<td>D, 0</td>
<td>D V 0</td>
<td>B</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0, A</td>
<td>0 V A</td>
<td>Q</td>
<td>0</td>
<td></td>
<td>MOVE J TO Q-REG</td>
</tr>
<tr>
<td></td>
<td>0, B</td>
<td>0 A B</td>
<td>B</td>
<td>3</td>
<td></td>
<td>CLEAR R 3</td>
</tr>
</tbody>
</table>

**MULTIPLY ALGORITHM**

<table>
<thead>
<tr>
<th>MULTTEST</th>
<th>ENA</th>
<th>0, B</th>
<th>A, B</th>
<th>B/A + B, SHIFT LEFT</th>
<th>B</th>
<th>1</th>
<th>3</th>
<th>ADD MULTIPLICAND TO R 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IF LSB OF Q IS ONE. THEN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHIFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SHIFT LEFT COMBINED PRODUCT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LEFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R 3 (NOTE TIMING QLO)</td>
<td></td>
</tr>
</tbody>
</table>

**REPEAT 15 TIMES**

<table>
<thead>
<tr>
<th>MULTTEST</th>
<th>ENA</th>
<th>0, B</th>
<th>A, B</th>
<th>B - 0/A, SHIFT LEFT</th>
<th>B</th>
<th>1</th>
<th>3</th>
<th>TRIAL SUBTRACT AND GATING</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OF SIGN INTO MSB OF R 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHIFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LEFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| MULTTEST | ENA       | 0, Q | 0 V Q | B | 2 | STORE PRODUCT L.O FROM |
|----------|-----------|------|-------|---|---|Q TO R 2 |

* I 2,1,p = φ, I 1,v (MULTIEN A QLO), 1 = φ 1/φ 11

**SYBEX**
BIT-SLICE ARRAY: SEQUENCING

MULTIPLY ALGORITHM: 21 INSTRUCTION STEPS
(16 BIT RIPPLE 260 NS  16 BIT LOOK AHEAD  210 NS)

SEQUENCE DATA IS CALLED

MICRO SEQUENCE
MICRO PROGRAM
MACRO PROGRAM

FOR THE MULTIPLY CONFIGURATION 21 x 18 BITS.
VARIOUS METHODS OF APPLYING SEQUENCE DATA TO DEVICE INPUTS

LOGIC

![Logic Diagram]

ROM

![ROM Diagram]
BIT SLICE ARRAY SEQUENCING TECHNIQUES

MEMORY AND LOGIC

MEMORY, LOGIC AND ROM

SYBEX
BIT SLICE ARRAY SEQUENCING TECHNIQUES

LOGIC ONLY (HARDWARE PROCESSOR)

FAST, EXPENSIVE, INFLEXIBLE, FROZEN

ROM ONLY (MICROSEQUENCED PROCESSOR)

FAST, INEXPENSIVE, FLEXIBLE, FROZEN

MEMORY AND LOGIC (PROGRAMMED PROCESSOR)

SLOW, EXPENSIVE, FLEXIBLE, PROGRAMABLE

MEMORY, LOGIC, AND ROM (MICROPROGRAMMED PROCESSOR)

FAST, INEXPENSIVE (?), FLEXIBLE, PROGRAMABLE

--THE LAST CATEGORY COVERS ALMOST EVERY MACHINE MARKETED SINCE ROMS AND MSI BECAME COMMERCIAL

SYBEX
BIT SLICE MICROPROGRAM SEQUENCERS

CONTROL ROM INSTRUCTION SOURCE
MAY CONTROL BIT SLICE STATUS AND SHIFT LINES
MAY CONTROL ANCILLARY LOGIC
SIMPLEST FORM

LOADS ADDRESS REGISTER FROM EXTERNAL DATA
INCREASES ADDRESS REGISTER

SYBEX
AMD 2909 MICRO SEQUENCER

4 BIT SLICE CASCADABLE TO ANY WIDTH

ADDRESS REGISTER
FOUR DEEP PUSH DOWN STACK WITH PUSH/POP CONTROL
MICRO PROGRAM COUNTER REGISTER
ADDRESS INCREMENTER

DIRECT ADDRESS REGISTER INPUT
MULTIPLEXER DATA INPUT
OR'ED DATA INPUT
28 PIN PACK

AMD 2911 SAME EXCEPT NO OR'ED INPUTS, DATA AND ADDRESS INPUTS SHARED
20 PIN PACK
The sequencer also controls the microprogram to itself.

### Microsequencer Control Inputs

<table>
<thead>
<tr>
<th>S1 S0</th>
<th>Address Source Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>µPC</td>
</tr>
<tr>
<td>1</td>
<td>AR</td>
</tr>
<tr>
<td>2</td>
<td>STK 0</td>
</tr>
<tr>
<td>3</td>
<td>DI</td>
</tr>
</tbody>
</table>

### Stack Control

<table>
<thead>
<tr>
<th>FE, PUP</th>
<th>Stack Control Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NO CHANGE</td>
</tr>
<tr>
<td>0 1</td>
<td>INCREMENT STKPTR, PUSH µPC → STK0</td>
</tr>
<tr>
<td>0 0</td>
<td>DECREMENT STKPTR (POP)</td>
</tr>
</tbody>
</table>

### Enable Address Register

SYBEX
EVERY MICROINSTRUCTION MUST CONTAIN:

1. BIT SLICE ARRAY CONTROL
2. ARRAY ANCILLARY LOGIC CONTROL
   SHIFT END AROUND
   MULTIPLY TEST ENABLE
3. MICRO SEQUENCER CONTROL
4. MICROPROGRAM BRANCH ADDRESSING
5. SEQUENCER ANCILLARY LOGIC CONTROL
   CONDITIONAL TESTS
   MICRO LOOP COUNTER
6. PROCESSOR BUS CONTROL
   MEMORY
   I/O
   INSTRUCTION DECODE

SYBEX
MICROPROGRAMMED 16-BIT PROCESSOR WITH AMD 2901/2909

DATA

X-BUS DATA

ADDRESS

X-BUS ADDRESS

DATA OUT BUS - Y

DATA IN BUS - D

OP DC

INT 1 VECTOR

INT 0 VECTOR

2909 x 2

2909 x 4

ROM 256 WORDS X 48 BITS

BUFFER

COND LOGIC

MISG CONTROL

BRANCH / LITERAL

S1, S0, FE, PUP

LOOPCTR

2901 x 4

D BUS, Y BUS

B/L, BR, LPCTL
SELECTING MICROPROGRAM STORAGE - ROM

TWO CRITERIA

1. SIZE
2. SPEED

OTHER CRITERIA

1. PROM AVAILABLE
2. SECOND SOURCE

SIZE: EXAMPLE

IF TOTAL ROM SPACE REQUIRED 256 WORDS x 48 BITS
CHOICE 256 x 4 TWELVE REQUIRED
256 x 8 SIX REQUIRED (HALF BOARD SPACE)
BUT SLIGHT TIME PENALTY
ROM SELECTION AND USE

SPEED OF CHIP

ROMS/ SIGNETICS 825229 256x4 TA = 50ns
SIGNETICS 825214 256x8 TA = 60ns
PROMS/ SIGNETICS 825129 256x4 TA = 50ns
SIGNETICS 825114 256x8 TA = 60ns

SPEED IN SYSTEM (SERIAL)

+ FROM COMPLETION OF CYCLE

CHANGE ADDRESS
DELAY (WORST CASE FILE POP) ROM 2901 ARRAY CYCLE (-20ns)
+ 40NS ACCESS TO ABI TRUE 60NS (50NS) 190NS =290NS/(280NS)

SYBEX
BUFFERED OPERATION (PIPELINING)

INSTEAD OF SERIAL OPERATION OF THE SEQ→ROM→ARRAY

BUFFERED OPERATION

ARRAY AND SEQ/ROM CONTROL IN PARALLEL
ARRAY LAGS ONE STEP BEHIND SEQ/ROM
BUFFERED SYSTEM TIMING

* THROUGH BUFFER DELAY ~15ns (AM 2918) / ALLOW 20ns

ADVANTAGES
210ns VERSUS 290ns
NON-CRITICAL SEQUENCER AND ROM TIMINGS
ALLOWS MORE LOGIC IN CONTROL CIRCUITS
ALLOWS SLOWER ROMS

DISADVANTAGES
REQUIRES BUFFER
"INITIATE/EXECUTE OFFSET"

SYBEX
INITIATE/EXECUTE OFFSET PROBLEMS

THE BUFFERED ARCHITECTURE REQUIRES SEQUENCER/ROM TO INITIATE \( i + 1 \) WHILE ARRAY IS EXECUTING

(Note: The micro-instruction \( i \) in the buffer is conditioning the sequencer during the initiation of \( i + 1 \).)

Consequences:

1. If the outcome of instruction \( i \) must condition \( i + 1 \) address
   then \( i + 1 \) must be delayed one cycle
2. If a branch is to be executed in cycle after instruction \( i \)
   then the new address must be available to sequencer at beginning of \( i \)
3. If instruction \( i \) is last in a loop
   then loop counter must be decremented during \( i - 1 \)

SYBEX
## CONDITION TESTS - FOR 2901/2909 CPU

<table>
<thead>
<tr>
<th>CONDITIONS TO BE TESTED</th>
<th>TIME AVAILABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM ARRAY:</td>
<td></td>
</tr>
<tr>
<td>CARRY</td>
<td>END CYCLE i</td>
</tr>
<tr>
<td>OVR</td>
<td>END CYCLE i</td>
</tr>
<tr>
<td>SIGN</td>
<td>END CYCLE i</td>
</tr>
<tr>
<td>F=0</td>
<td>END CYCLE i</td>
</tr>
<tr>
<td>FROM LOOP CTR: CTR=0</td>
<td>BEGIN. CYCLE i</td>
</tr>
<tr>
<td>OTHERS:</td>
<td></td>
</tr>
<tr>
<td>BUS ACCESS</td>
<td>ASYNCHRONOUS</td>
</tr>
<tr>
<td>INTERRUPT</td>
<td>ASYNCHRONOUS</td>
</tr>
</tbody>
</table>

**SYBEX**
## CONDITION MICROINSTRUCTION SEQUENCING

An "ABORT" facility is required.

<table>
<thead>
<tr>
<th>CYCLE</th>
<th>EXECUTE</th>
<th>FETCH</th>
<th>CONDITION</th>
<th>ABORT NEXT CYCLE EXECUTE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>i</td>
<td>i+1</td>
<td>Ti (→ BR)</td>
<td>YES</td>
</tr>
<tr>
<td>2</td>
<td>ABORTED</td>
<td>i(BR)</td>
<td>-</td>
<td>(NO)</td>
</tr>
<tr>
<td>3</td>
<td>i(BR)</td>
<td>i(BR)+1</td>
<td>-</td>
<td>(NO)</td>
</tr>
<tr>
<td>1'</td>
<td>i</td>
<td>i+1</td>
<td>Fi (→ BR)</td>
<td>(NO)</td>
</tr>
<tr>
<td>2'</td>
<td>i+1</td>
<td>i+2</td>
<td>-</td>
<td>(NO)</td>
</tr>
</tbody>
</table>

**Note:**
- Conditions available end cycle 1/1'
- Abort decision made early in cycle 2/2'
- Execution of i+1 in progress during cycle 2

**SYBEX**
EXECUTION ABORT

DEFINITION

AN INSTRUCTION IS ABORTED IF PREVENTED FROM ALTERING REGISTER OR CONDITION STATES.

NOTE: ALL 2901/2909 REGISTERS AND FLAGS ENABLES ON CLOCK
(RAM LATCHES MAY BE DISREGARDED)

ABORT IS ACCOMPLISHED BY PREVENTING FOR GIVEN INSTRUCTION IMPLEMENTATION

SYBEX
EXECUTION ABORT TIMING

ABORT SIGNAL MUST STABILIZE BEFORE CLKHL

**CYCLE 2**
- CLK
- ABORT
- CLK'
- EXECUTE
- TEST CONDITION LATCHED/FLAGS
- FETCH

**CYCLE 3**
- EXECUTE ABORTED
- FETCH
- (BR) FETCH
- (BR)+1 FETCH

ADDRESS BR MUST BE AT 2909 MUX HERE

CHANGE \( i+1 \) TO \( i(BR) \) AT INSTRUCTION BUFFER OUTPUTS MUST OCCUR HERE

SYBEX
CONDITIONAL TEST IMPLEMENTATION USING 2909

TEST AND BRANCH IN INSTRUCTION (EXAMPLE P.63)

BRANCH ADDRESS AND TEST MUST BE LATCHED

DURING i + 1. (USE ADR REG IN 2909s FOR BR)
ALTERNATIVE TEST AND BRANCH INSTRUCTION \( \ell + 1 \)

Flags of instruction \( \ell \) are latched

Test field and branch address gated from buffer during \( \ell + 1 \) (use 2909 D\(_{0-7}\) inputs)
COMPARISON OF BRANCH/TEST IMPLEMENTATIONS

FAR-REACHING IMPLICATIONS:

BRANCH/TEST IN $i$

ADVANTAGES
- CLEANER, MORE COMPACT PROGRAM
- GREATER PARALLELISM (OVERLAP OF TEST AND EXECUTION OF $i+1$ IF TEST FAILS)
- CLEANER TIMING (BUFFERED ADDRESS LOOP)

DISADVANTAGES
- WIDER ROM WORD
- EXTRA CHIP FOR TEST LATCH

BRANCH/TEST IN $i+1$

ADVANTAGES
- ENCODED MICROINSTRUCTION SAVES ROM WIDTH
- LEAVES 2909 ADR. REGISTER FREE FOR LOOP IMPLEMENTATION
- LESS HARDWARE (NO TEST LATCHES - ONE CHIP)

DISADVANTAGES
- LESS COMPACT MICROPROGRAM
- LESS PARALLELISM
  1. NO OVERLAP OF TEST
  2. EXTRA INSTRUCTION FOR BRANCHES AND RETURNS
  3. EXTRA STEP FOR INITIALIZING LOOP RETURNS

SYBEX
COMPARISON OF MICRO INSTRUCTION FORMATS

BRANCH IN i

<table>
<thead>
<tr>
<th>I6-8</th>
<th>A7-3</th>
<th>B7-3</th>
<th>D, Y BUS</th>
<th>LOOP</th>
<th>SHIFT</th>
<th>5% LC</th>
<th>CTRL</th>
<th>COND</th>
<th>BRANCH ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

16 BIT LITERAL (Q=8) 8

48 BITS

BRANCH/TEST IN i+1

<table>
<thead>
<tr>
<th>I6-8</th>
<th>A7-3</th>
<th>B7-3</th>
<th>D, Y BUS</th>
<th>SHIFT</th>
<th>5% LC</th>
<th>CTRL</th>
<th>COND</th>
<th>EXECUTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

BRANCH

<table>
<thead>
<tr>
<th>I6-8</th>
<th>A7-3</th>
<th>B7-3</th>
<th>D, Y BUS</th>
<th>CTRL</th>
<th>COND</th>
<th>BRANCH ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>16</td>
<td>16 BIT LITERAL (Q=8) 8</td>
</tr>
</tbody>
</table>

32 BITS
COMPARISON: SUMMARY

1. BRANCH/TEST IN \( i \)
   - FASTER BUT MORE EXPENSIVE
   - FEWER ROM WORDS/WIDER FIELD

2. BRANCH/TEST IN \( i + 1 \)
   - CHEAPER BUT SLOWER
   - MORE WORDS/NARROWER FIELD

HOW TO DECIDE

- IF SPEED IS CRUCIAL -- CHOICE 1
- IF COST IS CRUCIAL: PROBABLY BIT SLICES AREN'T THE RIGHT TECHNOLOGY
- IF SPACE IS CRUCIAL THEN CHOICE 2
  - IF THE LACK OF MICROPROGRAM COMPACTNESS DOESN'T
  - FORCE THE ADDITION OF ANOTHER BANK OF ROMS

THE APPLICATION MUST DECIDE
CONDITIONAL LOGIC TIMING REQUIREMENTS

TIMING: ASSUME 210 NS CLOCK WITH $T_L = 50$

ASSUME 20 NS CLK-ABORT $\rightarrow$ CLK'

CONDITIONAL LOGIC MUST

1. GENERATE ABORT SIGNAL BEFORE CLK
   210 NS - $T_L$ - DELAY = 140 NS

2. GENERATE $S_0$ TO SELECT BRANCH ADDRESS
   IN TIME FOR: ROM OUT $\rightarrow$ BUFFER AT
   210 NS - (SEQUENCER DELAY + ROM DELAY)
   ($S_i$, $S_0$ $\rightarrow$ $Y_t$)
   210 NS - (20 NS + 60 NS) = 130 NS

ASSUME 20 NS LATCH/BUFFER DELAY AT START OF CYCLE

MAX CONDITIONAL LOGIC DELAY = 110 NS

SYBEX
### Conditional Tests

**Microinstruction Field**

<table>
<thead>
<tr>
<th>Cond</th>
<th>T/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TRUE (UNCONDITIONAL)</td>
</tr>
<tr>
<td>1</td>
<td>CARRY = 1</td>
</tr>
<tr>
<td>2</td>
<td>OVR = 1</td>
</tr>
<tr>
<td>3</td>
<td>F15 = 1</td>
</tr>
<tr>
<td>4</td>
<td>F0-15 = 0</td>
</tr>
<tr>
<td>5</td>
<td>F0-7 = 0 (LOWER BYTE)</td>
</tr>
<tr>
<td>6</td>
<td>F8-15 = 0 (UPPER BYTE)</td>
</tr>
<tr>
<td>7</td>
<td>BUS</td>
</tr>
</tbody>
</table>

**T/F**

- 0: Branch if condition true
- 1: Branch if condition false

*SYBEX*
### BRANCHING OPERATIONS DEFINITION

Assume branch/test in instruction (48 bit implementation)

I - Field B/L = 0 (branch)

<table>
<thead>
<tr>
<th>h#:</th>
<th>Symbol</th>
<th>2909 Action</th>
<th>S₁</th>
<th>S₀</th>
<th>FE</th>
<th>PUP</th>
<th>Abort</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>µPC TO ROM ADR</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>NO</td>
</tr>
<tr>
<td>1</td>
<td>BR</td>
<td>AR TO ROM ADR</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>YES</td>
</tr>
<tr>
<td>2</td>
<td>PUSH BR</td>
<td>AR TO ROM ADR</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>µPC TO STK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>POPRET</td>
<td>STKØ TO ROM ADR</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>YES</td>
</tr>
</tbody>
</table>

Plus loop return (specified in LP field)

- LPRET BRANCH/DQ-7 TO ROM ADR | 1  | 1  | 1  | x | NO!

And interrupt (automated)

- INTERRUPT INT./ADR/DQ-7 TO ROM ADR | 1  | 1  | 0  | 1 | NO

Priority: loop return > literal/brops > interrupts

SYBEX
LOOP RETURN MICRO COMMANDS

LR \[LD, TD\]  
LD: LOAD LOOP COUNTER 0 = NOP/1 = LOAD
TD: TEST AND DECREMENT 0 = NOP/1 = TEST LPCTR ≠ Ø?

TIMING:
UNLIKE ALU TESTS, LOOP CTR ≠ Ø IS AVAILABLE AT START OF CYCLE
BRANCH ADDRESS IN INSTRUCTION BUFFER DURING
1. GATED INTO 2909 D INPUT IF COUNTER ≠ Ø
2. IGNORED OTHERWISE
NO ABORT IS REQUIRED
A CYCLE IS SAVED IN EACH ITERATION
ASSUME A MICROPROGRAM LOOP

\[ i_{u_p} \text{ /FIRST INSTRUCTION} \]

\[ : \]

\[ \iota_t \text{ /... TD = 1, BR, ADDR = } i_{u_p} \text{ /END OF LOOP} \]

\[ \iota_t + 1 \text{ /NEXT INSTRUCTION} \]

**EXECUTE**

**BUFFER**

**ADDRESS**

**FIELD**

**TD**

\[ \phi \]

\[ 1 \]

\[ \iota_t \]

\[ \iota_{t+1} \]

**LOOP CTR**

**FETCH**

**uPC**

**TEST/DECR**

**OCCURS HERE**

\[ \text{CTR } \neq \emptyset \rightarrow \text{BR} \]

**SYMEX**

\[ \iota_t \]

\[ \iota_t + 1 \]

\[ \iota_t + 1 \]

\[ 1 \]

\[ \phi \]

\[ \text{TEST, NO DECR OCCURS HERE} \]

\[ \text{CTR } = \emptyset \rightarrow \iota_t + 1 \]
BRANCH/TEST CONTROL LOGIC

SYBEX
BRANCH LOGIC IMPLEMENTATION

THE LOGIC IN THE BOX CAN BE PERFORMED BY

- 8 CHIPS DISCRETE LOGIC
- 1 ROM 512 x 8
- 1 PLA 14(16) x 48 x 8

<table>
<thead>
<tr>
<th>METHOD</th>
<th>SPEED</th>
<th>PINS*</th>
<th>DISSIPATION</th>
<th>COST</th>
<th>SPACE</th>
<th>EXPANSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 CHIPS L.S.</td>
<td>40 NS</td>
<td>8 x 16 = 128</td>
<td>8 x 50 mW = 400 mW</td>
<td>$8</td>
<td>4 IN²</td>
<td>NO</td>
</tr>
<tr>
<td>1 ROM 512 x 8</td>
<td>60 NS</td>
<td>24</td>
<td>700 mW</td>
<td>$15</td>
<td>1 IN²</td>
<td>NO</td>
</tr>
<tr>
<td>1 PLA (48)</td>
<td>50 NS</td>
<td>24/28</td>
<td>600 mW</td>
<td>$20?</td>
<td>1 IN²</td>
<td>YES (5 SIGNALS)</td>
</tr>
</tbody>
</table>

*RELIABILITY AND BOARD COMPLEXITY

**HIGHLY UNPREDICTABLE
P.L.A. REVIEW

P.L.A. IS A THREE STAGE ARRAY

FUNCTION STAGE 1: PRODUCTS OF INPUT TERMS

\[ P_n = K_{n,0} \cdot \phi \wedge K_{n,1} \cdot i_1 \wedge K_{n,2} \cdot i_2 \wedge \ldots \]

WHERE \( K_{n,m} = 1, \emptyset, -1 \) SUCH THAT \( 1 \cdot i = i \)
\[ \emptyset \cdot i = \text{undefined (i.e. no term)} \]
\[ -1 \cdot i = \overline{i} \]

\( n = 0, 1, 2, \ldots \)

\( K \) IS A MATRIX

\[ \begin{array}{cc}
  y & K_{x,y} \\
  x & \end{array} \]

WHICH MAPS ONTO

PRODUCT MATRIX

\[ P \]

SYBEX
AT EACH INTERSECTION AT LEAST ONE FUSE MUST BE BLOWN
STANDARD INPUTS \( i_y \): 14 or 16
PRODUCTS \( P_x \): 48 or 96

SYBEX
PLA STAGE 2: SUMMATION

FUNCTION STAGE 2: SUMMATION OF PRODUCT TERMS

\[ S_r = J_{r,\phi} \cdot P_{\phi} \lor J_{r,1} \cdot P_1 \lor J_{r,2} \cdot P_2 \lor \ldots \]

WHERE \( J_{r,s} = 1, 0 \) SUCH THAT \( 1 \cdot P = P \)

\( 0 \cdot P = \text{UNDEFINED (i.e. NO TERM)} \)

J IS A MATRIX

\[
\begin{array}{c|c}
    x & J_{x,z} \\
    \hline
    z & \end{array}
\]

WHICH MAPS ONTO

S' SUM MATRIX

P

NOTE: STAGE 3 FUNCTION -- NEGATION OF SUM TERMS

\[ S_r' = L_r \cdot S_r' \]

WHERE \( L_r = 1, -1 \) SUCH THAT \( 1 \cdot S' = S' \)

\( -1 \cdot S' = \overline{S'} \)

SYBEX
CIRCUIT OF SUMMATION (STAGE 2) MATRIX

STAGE 3
(NEGATION OF SUM TERMS)

STANDARD $S_Z = 8$
($P_X = 48\ or\ 96$)

SYBEX
PLA CHARACTERISTICS

THREE STANDARD CONFIGURATION

14 INPUTS 48 PRODUCTS 8 OUTPUTS (PLA, FPLA)
16 INPUTS 48 PRODUCTS 8 OUTPUTS (PLA, FPLA)
14 INPUTS 96 PRODUCTS 8 OUTPUTS (PLA ONLY)

AVAILABLE OC OR TRI-STATE OUTPUT, OUT ENABLE
RANGE OF SPEED 50 NS - 100 NS
DISSIPATION: EXAMPLE (SIGNETICS 82S101 16 x 48 x 8 FPLA 50 NS)
600 mW
24 OR 28 PIN PACKS

PROGRAMMING: SPECIFY J,K,L MATRICES BY PAPERTAPE OR CARDS
PLA: FACTORY MASKED
FPLA: FIELD PROGRAMMABLE WITH FUSE BLOWER

SYBEX
## PLA Mask Coding: Cards or Tape

### Program Table of Sample Device

<table>
<thead>
<tr>
<th>Input Variable</th>
<th>Output Function</th>
<th>Output Active Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prod. Term Present in f&lt;sub&gt;n&lt;/sub&gt;</td>
<td>Prod. Term Not Present in f&lt;sub&gt;n&lt;/sub&gt;</td>
</tr>
<tr>
<td>H, L</td>
<td>A</td>
<td>H</td>
</tr>
<tr>
<td>L, L</td>
<td>- (dash)</td>
<td>L</td>
</tr>
</tbody>
</table>

### Product Term

<table>
<thead>
<tr>
<th>No.</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>1</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>3</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>4</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
1. Enter 1 for unused inputs of used P-terms.
2. Enter 0 for unused outputs of used P-terms.

### Output Level

<table>
<thead>
<tr>
<th>Active Level</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Low</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

### TWX Tape Coding Format

Entries for the 3 Data Fields are determined in accordance with the following Table:

<table>
<thead>
<tr>
<th>Input Variable</th>
<th>Output Function</th>
<th>Output Active Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>H, L</td>
<td>Prod. Term Present in f&lt;sub&gt;n&lt;/sub&gt;</td>
<td>Prod. Term Not Present in f&lt;sub&gt;n&lt;/sub&gt;</td>
</tr>
<tr>
<td>L, L</td>
<td>- (dash)</td>
<td>A</td>
</tr>
</tbody>
</table>

**Notes:**
1. Enter 1 for unused inputs of used P-terms.
2. Enter 0 for unused outputs of used P-terms.
EITHER FUSE F1 OR F2 IS BLOWN

SYBEX
<table>
<thead>
<tr>
<th>STX</th>
<th>( \ast A )</th>
<th>( LLLLLLLLL )</th>
<th>( \ast P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ast P ) 00</td>
<td>LL----L--------H</td>
<td>( F )</td>
<td>( LLLLLLLLL )</td>
</tr>
<tr>
<td>( \ast P ) 01</td>
<td>LLLL LH--------</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 02</td>
<td>LL HLLH--------</td>
<td>( F )</td>
<td>( AA-----AA )</td>
</tr>
<tr>
<td>( \ast P ) 03</td>
<td>HL LL LH--------</td>
<td>( F )</td>
<td>( A---A--AA )</td>
</tr>
<tr>
<td>( \ast P ) 04</td>
<td>LH LL LH--------</td>
<td>( F )</td>
<td>( A---A--AA )</td>
</tr>
<tr>
<td>( \ast P ) 05</td>
<td>LH LLLH--------HL</td>
<td>( F )</td>
<td>( AA-----AA )</td>
</tr>
<tr>
<td>( \ast P ) 06</td>
<td>LH LLLH--------HL</td>
<td>( F )</td>
<td>( AA-----AA )</td>
</tr>
<tr>
<td>( \ast P ) 07</td>
<td>LH LLLH--------HL</td>
<td>( F )</td>
<td>( AA-----AA )</td>
</tr>
<tr>
<td>( \ast P ) 08</td>
<td>LH LLLH--------HL</td>
<td>( F )</td>
<td>( AA-----AA )</td>
</tr>
<tr>
<td>( \ast P ) 09</td>
<td>LH LL LH--HLLLL</td>
<td>( F )</td>
<td>( AAAA--AA )</td>
</tr>
<tr>
<td>( \ast P ) 10</td>
<td>LH LL LH--HLLLL</td>
<td>( F )</td>
<td>( AAAA--AA )</td>
</tr>
<tr>
<td>( \ast P ) 11</td>
<td>LH LLLHLLL L L</td>
<td>( F )</td>
<td>( AAAA--AA )</td>
</tr>
<tr>
<td>( \ast P ) 12</td>
<td>LL----L--------HL</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 13</td>
<td>LLHLLH--------</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 14</td>
<td>HLHLLH--------</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 15</td>
<td>HHHHLLH--------</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 16</td>
<td>LH HLLH--------H</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 17</td>
<td>LH HLLH--------H</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 18</td>
<td>LH HLLH--------H</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 19</td>
<td>LH HLLH--------H</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 20</td>
<td>LH HLLH HLLLL</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 21</td>
<td>LH HLLH HLLLL</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 22</td>
<td>LH HLLH HLLLL L H</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
<tr>
<td>( \ast P ) 23</td>
<td>LH HLLH HLLLL L L</td>
<td>( F )</td>
<td>( AAAAAAA )</td>
</tr>
</tbody>
</table>

ETX
## FPLA'S

<table>
<thead>
<tr>
<th></th>
<th>INPUTS</th>
<th>PRODUCT TERMS</th>
<th>OUTPUTS</th>
<th>PINS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD</td>
<td>16</td>
<td>48</td>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>INTERSIL</td>
<td>14</td>
<td>48</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>MMI</td>
<td>14</td>
<td>48</td>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>SIGNETICS</td>
<td>16</td>
<td>48</td>
<td>8</td>
<td>24</td>
</tr>
</tbody>
</table>
CURRENT PLA'S

- HUGHES
- INTERSIL
- MMI
- NATIONAL SEMICONDUCTOR
- SIGNETICS

SYBEX
- SEQUENCING

- ALU CONTROL = MICROINSTRUCTION DECODING

- CODE CONVERSION

- ASC II STANDARD INTERFACE BUS (IEEE 488-1975)

- FAST DECODING (50-100 NS)

- ROM PATCHES
BRANCH CONTROL PLA PRODUCT TERMS

BRANCH & LITERAL PRIOR

CONDOUT (MUX)
CONT/F
BR 0 = NOP
1 = BR
2 = PUSH BR
3 = POPPET

BL
(BL = \emptyset BRANCH
= 1 LITERAL/NB)

TD
(TD = 1 ∧ LC ≠ \emptyset)
L.R. PRIORITY

INTRO_{\text{HIGHEST}} PRIORITY

INTRO 1

INTRO (OP. AVAIL)

PRODUCT TERMS (18)
**BRANCH CONTROL: PLA SUM TERMS**

<table>
<thead>
<tr>
<th>PRODUCT TERMS</th>
<th>LITERAL</th>
<th>INTRQ2</th>
<th>INTRQ4</th>
<th>POPRET(F)</th>
<th>PUSHBR(F)</th>
<th>BR(F)</th>
<th>POPRET(T)</th>
<th>PUSHBR(T)</th>
<th>BR(T)</th>
<th>NOOP</th>
<th>LOOPRET</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CONTROL AND DATA TO 2909 SEQUENCER

INSTRUCTION BUFFER (AM 2918)

INTERRUPTS
INT0: SYSTEM
INT1: I/O
MICROPROGRAMMED 16 BIT PROCESSOR WITH AMD 2901/2909

DATA

X-BUS DATA

ADDRESS

X-BUS ADDRESS

DATA OUT BUS - Y

DATA IN BUS - D

REG

2901 x 4

ROM

256 WORDS

x

48

BITS

BUFFER

COND

LOGIC

MISC

CONTROL

LOOPCIR

BRANCH / LITERAL

ROM ADDR

S1, S0, FE, PUP

INT 1 VECTOR

INT 0 VECTOR

OP. DC

REG

2909

x 2

2909

x 2

ROM ADDR

D BUS, Y BUS

B/L, BR, LPE, CTL
# BUS CONTROL

<table>
<thead>
<tr>
<th>D BUS SOURCES</th>
<th>CONTROLED BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>LITERAL ← D</td>
<td>B/L = 1</td>
</tr>
<tr>
<td>X BUS DATA ← D</td>
<td>D BUS₉ = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D BUS DESTINATIONS</th>
<th>ALWAYS ENABLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>D → OP DECODE</td>
<td>D BUS₁ = 1</td>
</tr>
<tr>
<td>D → A/B REG. REGS</td>
<td>I₀,₁,₂</td>
</tr>
<tr>
<td>D → 2901s</td>
<td>LC₀ = 1</td>
</tr>
<tr>
<td>D → LOOP CTR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Y BUS DESTINATIONS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Y → X BUS DATA</td>
<td>Y BUS₉ = 1</td>
</tr>
<tr>
<td>Y → X BUS ADDR/RD</td>
<td>Y BUS₁,₂ = 01</td>
</tr>
<tr>
<td>Y → X BUS ADDR/WR</td>
<td>Y BUS₁,₂ = 10</td>
</tr>
<tr>
<td>Y → X BUS ADDR/OPR *</td>
<td>R BUS₁,₂ = 11</td>
</tr>
</tbody>
</table>

*NOTE: CAUSES PAUSE → INT3 WITH OP*
D TO X - BUS LOGIC: DATA TRANSCEIVER

AM 2907  QUAD BUS TRANSCEIVER
TRI STATE RECEIVED OUTPUTS
LATCHED OPERATION
PARITY CHECK / GENERATE

LS  400 MW / 35 NS
20 PIN PACK

SYBEX
INT 1: I/O INTERRUPT GENERATION

INT1 IS GENERATED BY I/O REQUESTS

WHEN INT1 IS ACKNOWLEDGED THE INT0 VECTOR IS GATED TO SEQUENCER D INPUTS, \( \mu PC \) IS STACKED

THE INTERRUPT VECTOR IS THE ADDRESS OF THE MICROPROGRAM SEQUENCE TO SERVICE THE INTERRUPT

SINGLE LEVEL INTERRUPT SIMPLY FORCES FIXED ADDRESS

PRIORITIZED INTERRUPTS MUST BE SORTED
VECTORS MUST BE GENERATED

\[
\text{INT 1 VECTOR} \quad \begin{array}{c}
\text{ROM} \\
32 \times 8 \\
(TS)
\end{array} \quad \text{PRIORITY LEVEL}
\]

\[
\text{INT SELECT} \quad \{ \text{SIGNETICS} \} \\
\text{82S123}
\]

SYBEX
VECTORED PRIORITY INTERRUPT ENCODER AM 2914

8 LEVELS
MASKING REGISTER
STATUS FOR LOWEST PERMISSIBLE LEVEL

VECTOR OUTPUT
CASCADEABLE FOR 8 LEVELS
70 NS DELAY

SYBEX
SUMMARY

16 BIT PROCESSOR EXPLOITS MAXIMUM CAPABILITIES OF THE AMD 2900 BIT SLICE SERIES

MINIMUM CYCLE TIME
POWERFUL MICROINSTRUCTION
HIGH DEGREE OF PARALLELISM
LOOK AHEAD
FETCH PIPELINE
BRANCH PREPARATION (SINGLE INSTRUCTION INTERRUPT SERVICE)
CONDITION TESTING
LOOP COUNTING (SINGLE INSTRUCTION LOOPS)

POWERFUL DATA PATHS
SIGNIFICANT AUTOMATISM
OP CODE VECTORING
LOOP CONTROL
INTERRUPTS \{ SYSTEM
I/O

~ 50 CHIPS

SYBEX
NEW AMD CHIPS

- AM 2901A SCHEDULED (76Q4)
  IMPROVED SWITCHING SPEED

- NEW CONTROL CIRCUITS: (76Q4)
  - 29811 FOR SELECT LINES AND STACK CONTROL OF 2911
    CONTAINS LOOP COUNTER
  - 29803 CONTROLS OR INPUTS ON 2909 = MULTI-WAY BRANCHING
    (UP TO 4 TEST LINES)

- NEW 2-PORT 16 x 4 RAMS 29704, 29705 (76Q4)

- 2910 MICROPROGRAM CONTROLLER
JUMPS
  TO ZERO
  TO ADDRESS
  REPEAT IF COUNTER ≠ 0
  TO SUBROUTINE (1 of 2)
  TO EXTERNAL ADDRESS
  TO MAP ADDRESS

COUNTER OPERATIONS
  LOAD
  TEST AND JUMP

STACK OPERATIONS
  PUSH PC
  RETURN
  JUMP AND POP
- 12-BIT MICROPROGRAM ADDRESS GENERATOR

- 4 LEVEL STACK

- 12-BIT LOOP COUNTER (TEST/DECREMENT)

- 16 PAIRS OF INSTRUCTIONS:
  - 4-BIT PAIR SELECT
  - CONDITION CODE CC + LOOP COUNTER SELECTS ONE OF PAIR

- 40-PIN DIP
4. OTHER BIT-SLICE DEVICES

SYBEX
THE BIT-SLICE FAMILY TREE

1966
1973
1975
1976
1977

SYBEX
IMMEDIATE PREDECESSOR OF AMD 2901
ALMOST IDENTICAL ARCHITECTURE

Note: The numbers in parentheses are the numbers of signal lines.
## MM 6701 VERSUS AMD 2901

<table>
<thead>
<tr>
<th></th>
<th>6701</th>
<th>2901</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cycle</strong></td>
<td>185 NS (SPEC 175 NS)</td>
<td>105 NS</td>
</tr>
<tr>
<td>(6701 - 1: 90 NS)</td>
<td></td>
<td>A, B CLK</td>
</tr>
<tr>
<td><strong>Address Hold</strong></td>
<td>-10 NS/CLK</td>
<td>0 NS/CLK</td>
</tr>
<tr>
<td><strong>16 Bit Comparison</strong></td>
<td>185 NS (175)</td>
<td>145 NS</td>
</tr>
<tr>
<td>(W/O Shift Around)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flags Out</strong></td>
<td>C, OVR, F = 0, F = 1111</td>
<td>C, OVR, N, F = 0</td>
</tr>
<tr>
<td>(6701 - 1 HAS N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bussing</strong></td>
<td>A OR B OR F</td>
<td>A OR F</td>
</tr>
<tr>
<td>TO DATA OUT</td>
<td></td>
<td>TO DATA OUT</td>
</tr>
<tr>
<td>B→D→B</td>
<td></td>
<td>A→D→B</td>
</tr>
<tr>
<td><strong>Always Load B And/Or Q</strong></td>
<td>ALWAYS LOAD B AND/OR Q</td>
<td>LOAD B AND/OR Q OR NO LOAD</td>
</tr>
</tbody>
</table>

(MISCELLANEOUS ALU DIFFERENCES)
### A.C. Characteristics

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SYMBOL</th>
<th>CONDITIONS</th>
<th>S.V. ± 10%</th>
<th>S.V. ± 5%</th>
<th>S.V. ± 0%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td></td>
<td>MAX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Timing

**Figure 58**

- **Base Time Delay due to High to Low Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 59**

- **Base Time Delay due to Low to High Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 60**

- **Timing of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 61**

- **Base Time Delay due to High to Low Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 62**

- **Base Time Delay due to Low to High Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 63**

- **Timing of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 64**

- **Base Time Delay due to High to Low Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 65**

- **Base Time Delay due to Low to High Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 66**

- **Timing of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 67**

- **Base Time Delay due to High to Low Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 68**

- **Base Time Delay due to Low to High Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 69**

- **Timing of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 70**

- **Base Time Delay due to High to Low Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 71**

- **Base Time Delay due to Low to High Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 72**

- **Timing of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 73**

- **Base Time Delay due to High to Low Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 74**

- **Base Time Delay due to Low to High Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 75**

- **Timing of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 76**

- **Base Time Delay due to High to Low Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 77**

- **Base Time Delay due to Low to High Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 78**

- **Timing of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 79**

- **Base Time Delay due to High to Low Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)

**Figure 80**

- **Base Time Delay due to Low to High Transition of Clock**
  - Time Delay:
  - MAX: 70 ns (Typical)
  - MIN: 60 ns (Typical)
  - NON-Delayed:
    - MAX: 50 ns (Typical)
    - MIN: 40 ns (Typical)
# MM 6701 MICRO INSTRUCTIONS

## INSTRUCTIONS IN THE 37 x 9 ROM - POSITIVE LOGIC H = +3 V - INTERPRETATION

<table>
<thead>
<tr>
<th>ROM WORD</th>
<th>ALU Increment</th>
<th>ALU Output</th>
<th>TYPICAL USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

## INSTRUCTION MODIFIERS IN THE 8 x 8 ROM - POSITIVE LOGIC H = +3 V - INTERPRETATION

<table>
<thead>
<tr>
<th>ROM Word</th>
<th>B-A</th>
<th>Load Count</th>
<th>Out Count</th>
<th>Out Count</th>
<th>Out Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---

SYBEX
MM 5710/6710 MICROPROGRAM CONTROL UNIT

COMPLETE CONTROL IN ONE CHIP

CONDITIONAL BRANCHING
SHIFT CONTROL
1-LEVEL SUBROUTINE REGISTER
LOOP COUNTER
ADDRESS 512 WORDS MICRO CONTROL

SYBEX
### Table I - MCU Control Options

<table>
<thead>
<tr>
<th>Control Code</th>
<th>Address Field Destination</th>
<th>Control Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>None</td>
<td>Continue to next µinstruction</td>
</tr>
<tr>
<td>001</td>
<td>Control Counter</td>
<td>Continue to next µinstruction</td>
</tr>
<tr>
<td>010</td>
<td>None/CRAR (Cond. Jump)</td>
<td>Jump to next µinstruction if Control Counter ≠ 0, Decrement Control Counter</td>
</tr>
<tr>
<td>011</td>
<td>None/CRAR (Cond. Subr. Jump)</td>
<td>Subroutine Jump to next µinstruction if Control Counter ≠ 0, Decrement Control Counter</td>
</tr>
<tr>
<td>100</td>
<td>'None</td>
<td>Return from Subroutine</td>
</tr>
<tr>
<td>101</td>
<td>CRAR (Jump Subroutine)</td>
<td>Return from Subroutine when Control Counter Subroutine Latch = CRAR04</td>
</tr>
<tr>
<td>110</td>
<td>CRAR (Jump)</td>
<td>Jump to next µinstruction</td>
</tr>
<tr>
<td>111</td>
<td>CRAR (Jump Subroutine)</td>
<td>Subroutine Jump to next µinstruction</td>
</tr>
</tbody>
</table>

### Table III - Shift Control

<table>
<thead>
<tr>
<th>Control Code</th>
<th>Shifting Operation</th>
<th>Bidirectional Shift Lines Acting as Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F0 (SRO/SRI)</td>
</tr>
<tr>
<td>000</td>
<td>Arithmetic Shift Left</td>
<td>-</td>
</tr>
<tr>
<td>001</td>
<td>Arithmetic Shift Right</td>
<td>F3</td>
</tr>
<tr>
<td>100</td>
<td>Rotate Shift Left</td>
<td>-</td>
</tr>
<tr>
<td>101</td>
<td>Rotate Shift Right</td>
<td>F0</td>
</tr>
<tr>
<td>110</td>
<td>Rotate Shift Right</td>
<td>-</td>
</tr>
<tr>
<td>111</td>
<td>Rotate Shift Right</td>
<td>-</td>
</tr>
</tbody>
</table>

*High Impedance State
SC Contents of Carry Flip Flop

SYBEX
### Table II - Flag Status Control Options

<table>
<thead>
<tr>
<th>Control Code</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>None</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>Store C</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>Store N, V, Z</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>Store C, N, V, Z</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>Shift Flag Register into Q₀</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>Shift Flag Register out of Q₀</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>Instantaneous value of Q₃ to CRAR₀</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>Instantaneous value of Q₀ to CRAR₀</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>Stored value of C to CRAR₀</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>Stored value of N to CRAR₀</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>Stored value of V to CRAR₀</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>Stored value of Z to CRAR₀</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>Instantaneous value of C to CRAR₀</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>Instantaneous value of N to CRAR₀</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>Instantaneous value of V to CRAR₀</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>Instantaneous value of Z to CRAR₀</td>
</tr>
</tbody>
</table>

Code bit I₇ inverts the status of output line so that the condition is dependent upon C, etc. For the first six entries in the table if I₇ = 0 there is an unconditional branch to X,0. If I₇ = 1 an unconditional branch to X,1.
# COMPARISON 6710 VERSUS 2909

<table>
<thead>
<tr>
<th>Functions</th>
<th>6710</th>
<th>2909</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADDRESS GATING</td>
<td>ADDRESS GATING</td>
</tr>
<tr>
<td></td>
<td>LOOP COUNTING</td>
<td>STACK CONTROL</td>
</tr>
<tr>
<td></td>
<td>FLAG TESTING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHIFT CONTROL</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>COMPLETE CONTROL</td>
<td>SLICE OF CONTROL</td>
</tr>
<tr>
<td></td>
<td>UNIT FOR SMALL SYSTEM</td>
<td>UNIT FOR LARGE SYSTEM</td>
</tr>
<tr>
<td>Speed: Cycle</td>
<td>100 NS MIN</td>
<td>60 NS MIN</td>
</tr>
<tr>
<td>Speed: Clock TO ADDROUT</td>
<td>25 NS</td>
<td>40 NS (FILE POP MODE)</td>
</tr>
<tr>
<td>Size</td>
<td>40 PIN</td>
<td>28/20 PIN</td>
</tr>
</tbody>
</table>

*SYBEX*
# OTHER MM 5700/6700 SERIES DEVICES

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5702/6702</td>
<td>4 BIT CPU SLICE WITH ADDITIONAL BIDIRECTIONAL I/O PORT (NO Q REGISTER)</td>
</tr>
<tr>
<td>5716/6716</td>
<td>PRIORITY INTERRUPT CONTROLLER</td>
</tr>
<tr>
<td>5717/6717</td>
<td>DMA CONTROLLER</td>
</tr>
<tr>
<td>5718/6718</td>
<td>I/O INTERFACE CONTROLLER</td>
</tr>
</tbody>
</table>

**SYBEX**
INTEL 3000 SERIES

3002 CENTRAL PROCESSING ELEMENT

2 BIT WIDE SLICE  3 INPUT CHANNELS
11 GENERAL REGISTERS  2 OUTPUT CHANNELS
ACCUMULATOR  100 NS CYCLE
MEMORY ADDRESS REGISTER  28 PIN PACK
INTEL 3002 CPE

16 BIT CYCLE TIME WITH LOOK AHEAD (NO SHIFT AROUND): 155 NS (INTEL 3003)

ADVANTAGES:
   SPEED
   MULTIPLE BUS INPUTS AND OUTPUTS
   BUFFERED OUTPUTS

DISADVANTAGES:
   2 BITS PER CHIP
   LIMITED GENERAL REGISTER ACCESS - ALU A - MUX ONLY
   COMPLICATED MICROINSTRUCTION SET
   LIMITED FLAG OUT (CARRY AND LOOK AHEAD)
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>lcv</td>
<td>Clock Cycle Time (2)</td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>2.2</td>
</tr>
<tr>
<td>lwp</td>
<td>Clock Pulse Width</td>
<td>33</td>
<td>20</td>
<td>20</td>
<td>2.2</td>
</tr>
<tr>
<td>lfs</td>
<td>Function Input Set-Up Time (F0 through F8)</td>
<td>80</td>
<td>40</td>
<td>20</td>
<td>2.2</td>
</tr>
<tr>
<td>los</td>
<td>F0 through F8</td>
<td>50</td>
<td>30</td>
<td>27</td>
<td>2.2</td>
</tr>
<tr>
<td>S5s</td>
<td>L1, C1</td>
<td>27</td>
<td>13</td>
<td>20</td>
<td>2.2</td>
</tr>
<tr>
<td>tsh</td>
<td>Data and Function Hold Time:</td>
<td>5</td>
<td>-2</td>
<td>15</td>
<td>2.2</td>
</tr>
<tr>
<td>tfh</td>
<td>F0 through F8</td>
<td>5</td>
<td>-4</td>
<td>15</td>
<td>2.2</td>
</tr>
</tbody>
</table>

- Propagation Delay to X, Y, RO from:
  - 1xf: Any Function Input | 37  | 52  | 2.2  |
  - 1xd: Any Data Input | 29  | 42  | 2.2  |
  - 1xt: Leading Edge of CLK | 40  | 80  | 2.2  |
  - 1xl: Leading Edge of CLK | 20  | 2.2  |

- Propagation Delay to CO from:
  - 1cl: Leading Edge of CLK | 20  | 2.2  |
  - 1ct: Leading Edge of CLK | 48  | 70  | 2.2  |
  - 1cf: Any Function Input | 42  | 66  | 2.2  |
  - 1cd: Any Data Input | 30  | 66  | 2.2  |
  - 1cc: CI (Triple Carry) | 14  | 25  | 2.2  |

- Propagation Delay to Aq, A1, D0, D1 from:
  - 1ol: Leading Edge of CLK | 5   | 32  | 80  | 2.2  |
  - 1oe: Enable Input ED, EA | 12  | 25  | 2.2  |

Ta = -0°C to 70°C; VCC = 5V ± 5%
## APPENDIX A  MICRO-FUNCTION SUMMARY

<table>
<thead>
<tr>
<th>GROUP</th>
<th>R. GROUP</th>
<th>MICRO-FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I</td>
<td>( R_n ) + (AC &amp; K) + CI \rightarrow R_n, AC</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>M + (AC &amp; K) + CI \rightarrow AT</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>AT_L + (R_L &amp; K_L) \rightarrow RO L + [R_H &amp; K_H] &amp; AT_H \rightarrow AT_H</td>
</tr>
<tr>
<td>1</td>
<td>I</td>
<td>K &amp; R_n \rightarrow MAR R_n + K + CI \rightarrow R_n</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>K &amp; M \rightarrow MAR M + K + CI \rightarrow AT</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>(AT &amp; K) + [(AT &amp; K) &amp; CI] \rightarrow AT</td>
</tr>
<tr>
<td>2</td>
<td>I</td>
<td>(AC &amp; K) &amp; 1 + CI \rightarrow R_n</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>(AC &amp; K) &amp; 1 + CI &amp; AT \rightarrow [see Note 1]</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>(I &amp; K) &amp; 1 + CI &amp; AT</td>
</tr>
<tr>
<td>3</td>
<td>I</td>
<td>R_n + (AC &amp; K) &amp; CI \rightarrow R_n</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>M + (AC &amp; K) &amp; CI &amp; AT</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>AT + (I &amp; K) &amp; CI &amp; AT</td>
</tr>
<tr>
<td>4</td>
<td>I</td>
<td>CI &amp; (R_n &amp; AC &amp; K) &amp; CO R_n + (AC &amp; K) &amp; AT</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>CI &amp; (M &amp; AC &amp; K) &amp; CO M &amp; (AC &amp; K) &amp; AT</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>CI &amp; (AT &amp; I &amp; K) &amp; CO AT + (I &amp; K) &amp; AT</td>
</tr>
<tr>
<td>5</td>
<td>I</td>
<td>CI &amp; (R_n &amp; K) &amp; CO K &amp; R_n &amp; R_n</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>CI &amp; (M &amp; K) &amp; CO K &amp; M &amp; AT</td>
</tr>
<tr>
<td></td>
<td>III</td>
<td>CI &amp; (AT &amp; K) &amp; CO K &amp; AT &amp; AT</td>
</tr>
</tbody>
</table>

### NOTES
1. 2's complement and minisub add 111 11 to perform subtraction of 000 01
2. \( R_n \) includes \( T \) and AC as source and designation registers in Group I micro functions
3. Standard arithmetic carry output values are generated in Group I, 2 and 3 instructions

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>I, K, M</td>
<td>Data on the I, K, and M buses, respectively</td>
</tr>
<tr>
<td>CI, LI</td>
<td>Data on the carry input and left input, respectively</td>
</tr>
<tr>
<td>CO, RO</td>
<td>Data on the carry output and right output, respectively</td>
</tr>
<tr>
<td>R_n</td>
<td>Contents of register ( n ) including ( T ) and AC (R Group I)</td>
</tr>
<tr>
<td>AC</td>
<td>Contents of the accumulator</td>
</tr>
<tr>
<td>AT</td>
<td>Contents of AC on I, as specified</td>
</tr>
<tr>
<td>MAR</td>
<td>Contents of the memory address register</td>
</tr>
<tr>
<td>L, H</td>
<td>As subscripts, designate low and high order bit, respectively</td>
</tr>
</tbody>
</table>

- \( \& \): 2's complement addition
- \( \& \): 2's complement subtraction
- Logical AND
- Logical OR
- EXCLUSIVE OR
- EXCLUSIVE NOR
### APPENDIX B  ALL-ZERO AND ALL-ONE K-BUS MICRO-FUNCTIONS

<table>
<thead>
<tr>
<th>F-GROUP</th>
<th>R-GROUP</th>
<th>K-BUS = 00 MICRO-FUNCTION</th>
<th>MNEMONIC</th>
<th>K-BUS = 11 MICRO-FUNCTION</th>
<th>MNEMONIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I</td>
<td>$R_n + CI - R_n, AC$</td>
<td>ILR</td>
<td>$AC + R_n + CI - R_n, AC$</td>
<td>ALR</td>
</tr>
<tr>
<td>0</td>
<td>II</td>
<td>$M + CI - AT$</td>
<td>ACM</td>
<td>$M + AC + CI - AT$</td>
<td>AMA</td>
</tr>
<tr>
<td>III</td>
<td></td>
<td>$AT_L = RO \ AT_H = AT_L\ LI = AT_H$</td>
<td>SRA</td>
<td>(See Appendix A)</td>
<td>(See Appendix A)</td>
</tr>
<tr>
<td>1</td>
<td>I</td>
<td>$R_n \rightarrow MAR$</td>
<td>LMI</td>
<td>$11 \rightarrow MAR$</td>
<td>DSM</td>
</tr>
<tr>
<td>II</td>
<td>I</td>
<td>$M \rightarrow MAR$</td>
<td>LMM</td>
<td>$11 \rightarrow MAR$</td>
<td>LDM</td>
</tr>
<tr>
<td>III</td>
<td>I</td>
<td>$\overline{AT} + CI - AT$</td>
<td>CIA</td>
<td>$AT - 1 + CI - AT$</td>
<td>DCA</td>
</tr>
<tr>
<td>2</td>
<td>I</td>
<td>$CI - 1 \rightarrow R_n$</td>
<td>CSR</td>
<td>AC - 1 + CI - R_n</td>
<td>SDR</td>
</tr>
<tr>
<td>II</td>
<td>I</td>
<td>$CI - 1 \rightarrow AT$</td>
<td>CSA</td>
<td>AC - 1 + CI - AT</td>
<td>SDA</td>
</tr>
<tr>
<td>III</td>
<td>I</td>
<td>(See CSA above)</td>
<td>-</td>
<td>(See CSA above)</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>II</td>
<td>$R_n + CI - R_n$</td>
<td>LN</td>
<td>$AC + R_n + CI - R_n$</td>
<td>ADR</td>
</tr>
<tr>
<td>III</td>
<td>II</td>
<td>(See ACM above)</td>
<td>-</td>
<td>(See ACM above)</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>II</td>
<td>$CI - CO 0 \rightarrow R_n$</td>
<td>CLR</td>
<td>CI :(R_n + AC) - CO : R_n : AC - R_n</td>
<td>ANR</td>
</tr>
<tr>
<td>III</td>
<td>II</td>
<td>$CI - CO 0 \rightarrow AT$</td>
<td>CLA</td>
<td>CI :(M + AC) - CO : M : AC - AT</td>
<td>ANM</td>
</tr>
<tr>
<td>(See CLA above)</td>
<td></td>
<td>(See CLA above)</td>
<td>-</td>
<td>CI :(AT + 1) - CO : AT + 1 - AT</td>
<td>ANI</td>
</tr>
<tr>
<td>5</td>
<td>II</td>
<td>(See CLR above)</td>
<td>-</td>
<td>CI : R_n - CO : R_n - R_n</td>
<td>TZR</td>
</tr>
<tr>
<td>III</td>
<td>II</td>
<td>(See CLA above)</td>
<td>-</td>
<td>CI : M - CO : M - AT</td>
<td>LTM</td>
</tr>
<tr>
<td>(See CLA above)</td>
<td></td>
<td>(See CLA above)</td>
<td>-</td>
<td>CI : AT - CO : AT - AT</td>
<td>TZA</td>
</tr>
<tr>
<td>6</td>
<td>II</td>
<td>$CI - CO R_n - R_n$</td>
<td>NOP</td>
<td>CI : AC - CO : R_n : AC - R_n</td>
<td>ORR</td>
</tr>
<tr>
<td>III</td>
<td>II</td>
<td>$CI - CO M - AT$</td>
<td>LMF</td>
<td>CI : AC - CO : M : AC - AT</td>
<td>ORM</td>
</tr>
<tr>
<td>(See NOP above)</td>
<td></td>
<td>(See NOP above)</td>
<td>-</td>
<td>CI : 1 - CO : 1 - AT - AT</td>
<td>ORI</td>
</tr>
<tr>
<td>7</td>
<td>II</td>
<td>$CI - CO R_n - R_n$</td>
<td>CMR</td>
<td>CI :(R_n + AC) - CO : R_n : AC - R_n</td>
<td>XNR</td>
</tr>
<tr>
<td>III</td>
<td>II</td>
<td>$CI - CO M - AT$</td>
<td>LCM</td>
<td>CI :(M + AC) - CO : M : AC - AT</td>
<td>XNM</td>
</tr>
<tr>
<td>III</td>
<td>II</td>
<td>$CI - CO \overline{AT} - AT$</td>
<td>CMA</td>
<td>CI :(AT + 1) - CO : AT + 1 - AT</td>
<td>XNI</td>
</tr>
</tbody>
</table>

4. The more general operations, CSR and SDR, should be used in place of the CSA and SDA operations, respectively.
512 INSTRUCTION ADDRESSING
CONDITIONAL TESTS
BRANCH CONTROL
<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
<th>FUNCTION</th>
<th>NEXT ROW</th>
<th>NEXT COL</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCC</td>
<td>Jump in current column</td>
<td>0 0 d₄ d₃ d₂ d₁ d₀</td>
<td>d₄ d₃ d₂ d₁ d₀</td>
<td>m₃ m₂ m₁ m₀</td>
</tr>
<tr>
<td>JZR</td>
<td>Jump to zero row</td>
<td>0 1 0 d₃ d₂ d₁ d₀</td>
<td>0 0 0 0 0 0</td>
<td>d₃ d₂ d₁ d₀</td>
</tr>
<tr>
<td>JCR</td>
<td>Jump in current row</td>
<td>0 1 1 d₃ d₂ d₁ d₀</td>
<td>m₈ m₇ m₆ m₅ m₄</td>
<td>d₃ d₂ d₁ d₀</td>
</tr>
<tr>
<td>JCE</td>
<td>Jump in column/enable</td>
<td>1 1 1 0 d₂ d₁ d₀</td>
<td>m₈ m₇ m₆ m₵</td>
<td>m₃ m₂ m₁ m₀</td>
</tr>
<tr>
<td>JFL</td>
<td>Jump/test F-latch</td>
<td>1 0 0 d₃ d₂ d₁ d₀</td>
<td>m₈ m₇ m₆ m₅ m₄</td>
<td>m₃ 0 1 f</td>
</tr>
<tr>
<td>JCF</td>
<td>Jump/test C-flag</td>
<td>1 0 1 0 d₂ d₁ d₀</td>
<td>m₈ m₇ m₆ m₅</td>
<td>m₃ 0 1 c</td>
</tr>
<tr>
<td>JZF</td>
<td>Jump/test Z-flag</td>
<td>1 0 1 1 d₂ d₁ d₀</td>
<td>m₈ m₇ m₆ m₅</td>
<td>m₃ 0 1 z</td>
</tr>
<tr>
<td>JPR</td>
<td>Jump/test PR-latches</td>
<td>1 1 0 0 d₂ d₁ d₀</td>
<td>m₈ m₇ m₆ m₵</td>
<td>p₃ p₂ p₁ p₀</td>
</tr>
<tr>
<td>JLL</td>
<td>Jump/test left PR bits</td>
<td>1 1 0 1 d₂ d₁ d₀</td>
<td>m₈ m₇ m₆ m₵</td>
<td>0 1 p₃ p₂</td>
</tr>
<tr>
<td>JRL</td>
<td>Jump/test right PR bits</td>
<td>1 1 1 1 d₁ d₀ m₈ m₇</td>
<td>1 d₁ d₀</td>
<td>1 1 p₁ p₀</td>
</tr>
<tr>
<td>JPX</td>
<td>Jump/test PX-bus</td>
<td>1 1 1 1 0 d₁ d₀ m₈ m₇ m₆ d₁ d₀</td>
<td>x₇ x₆ x₅ x₄</td>
<td></td>
</tr>
</tbody>
</table>

**SYMBOL MEANING**

- dₙ: Data on address control line n
- mₙ: Data in microprogram address register bit n
- pₙ: Data in PR latch bit n
- xₙ: Data on PX-bus line n (active LOW)
- f, c, z: Contents of F-latch, C-flag, or Z-flag, respectively

*SYBEX*
The following ten diagrams illustrate the jump set for each of the eleven jump and jump/test functions of the MCU. Location 341, indicated by the black square, represents one current row (row2) and current column (cols) address. The grey boxes indicate the microprogram locations that may be selected by the particular function as the next address.
MULTIMODE 8 BIT LATCH (2 REQUIRED FOR BI-DIRECTIONAL)

INTERRUPT CONTROL UNIT

4 BIT BI-DIRECTION BUS DRIVERS/RECEIVERS

Figure 1. 3214 Block Diagram.

SYBEX
4 BIT CPU SLICE IN I²L
10 REGISTERS
FACTORY PROGRAMMABLE PLA ON CHIP
1000 NS CLOCK: TYPICAL ALU PROPAGATE=500 NS
FUNCTIONS ON .85 V @ 150 mA  128 mW MAX
FULL MILITARY TEMPERATURE RANGE AVAILABLE
CAN OPERATE OVER SEVERAL ORDERS OF MAGNITUDE OF CURRENT/PROPAGATE

SBPD400
NORMALIZED PROPAGATION DELAY TIME

INJECTOR CURRENT

SYBEX
## REGISTER-ALU SLICE/MOS-P CHANNEL

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multiplexed Command Bus</strong></td>
<td>Four Phase Execution</td>
</tr>
<tr>
<td><strong>7 General Register</strong></td>
<td></td>
</tr>
<tr>
<td><strong>16 Word Stack</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Bipolar Compatible Outputs</strong></td>
<td></td>
</tr>
<tr>
<td><strong>330 ns Min Per Phase</strong></td>
<td><strong>1.3 µs Per Instruction</strong></td>
</tr>
</tbody>
</table>

![ schematic diagram ]

**SYBEX**
### 1.A Command Inputs

<table>
<thead>
<tr>
<th>TIME INTERVAL</th>
<th>NCB(3)</th>
<th>NCB(2)</th>
<th>NCB(1)</th>
<th>NCB(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>STACK</td>
<td></td>
<td>R BUS</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>COMP</td>
<td></td>
<td>R BUS</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>CTL</td>
<td></td>
<td>ALU</td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td>I/O</td>
<td></td>
<td>R BUS</td>
<td></td>
</tr>
</tbody>
</table>

### 1.B Command Codes

#### ALU FUNCTIONS

<table>
<thead>
<tr>
<th>NCB(1), (0) @ T5</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>AND</td>
</tr>
<tr>
<td>10</td>
<td>XOR</td>
</tr>
<tr>
<td>01</td>
<td>OR</td>
</tr>
<tr>
<td>00</td>
<td>ADD</td>
</tr>
</tbody>
</table>

#### CTL FUNCTIONS

<table>
<thead>
<tr>
<th>NCB(3), (2) @ T5</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>NONE</td>
</tr>
<tr>
<td>10</td>
<td>R BUS CONTROL</td>
</tr>
<tr>
<td>01</td>
<td>SHIFT LEFT</td>
</tr>
<tr>
<td>00</td>
<td>SHIFT RIGHT</td>
</tr>
</tbody>
</table>

#### A, B & R BUS ADDRESSES

<table>
<thead>
<tr>
<th>NCB(2), (1), (0)</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>ZEROES, FLAGS, STACK</td>
</tr>
<tr>
<td>110</td>
<td>R1</td>
</tr>
<tr>
<td>101</td>
<td>R2</td>
</tr>
<tr>
<td>100</td>
<td>R3</td>
</tr>
<tr>
<td>011</td>
<td>R4</td>
</tr>
<tr>
<td>010</td>
<td>R5</td>
</tr>
<tr>
<td>001</td>
<td>R6</td>
</tr>
<tr>
<td>000</td>
<td>R7</td>
</tr>
</tbody>
</table>

### R BUS CONTROL

<table>
<thead>
<tr>
<th>I/O (NCB(3) @ T7)</th>
<th>BYTE (SININ @ T5)</th>
<th>R BUS VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>OUTPUT OF SHIFTER</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>OUTPUT OF SHIFTER</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>OUTPUT OF I/O MUX</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>VALUE OF SIGN INPUT</td>
</tr>
</tbody>
</table>

Notes:
1. Commands are complemented signals.
2. See text and Tables II and III for addressing flags and stack. B bus only addresses zeroes.
3. Logic values shown are values which must be applied to NCB inputs to get indicated results.
CONTROL UNIT AND ROM

PROVIDES MULTIPLEXED INSTRUCTIONS FOR RALU
BRANCH CONTROL
SHIFT AND CARRY CONTROL
SUBROUTINING
MICROPROGRAMMABLE: 100 INSTRUCTIONS 23 BITS
STANDARD 8 AND 16 BIT EMULATIONS AVAILABLE
DRIVES UP TO 8 RALUS
CONTROLLERS ARE ARRAYABLE

TABLE I. IMP Microinstruction Word Formats

ARITHMETIC INSTRUCTIONS

| 0, 1, 7, 3, 6, 5, 8, 7, 9, 0, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22 |
|-----------------|---|---|---|---|---|
| CONTROL | A | B | ADP | SHIFT | CONTROL |

I/O INSTRUCTIONS

| 0, 1, 7, 3, 4, 5, 6, 7, 9, 0, 10, 1, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22 |
|-----------------|---|---|---|---|---|
| CONTROL | RDST | WR | JADD | ST | LOAD |

JUMP INSTRUCTIONS

| 0, 1, 7, 3, 4, 5, 6, 7, 9, 0, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22 |
|-----------------|---|---|---|---|---|
| JLT | JADD | JCOND | CONTROL |
SYBEX
## LOGIC LEVELS

<table>
<thead>
<tr>
<th>SIGNALS (Note 1)</th>
<th>TIME INTERVALS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T1</td>
</tr>
<tr>
<td><strong>CLOCK</strong></td>
<td></td>
</tr>
<tr>
<td>φ₁</td>
<td>MOS</td>
</tr>
<tr>
<td>φ₂</td>
<td>MOS</td>
</tr>
<tr>
<td>φ₃</td>
<td>MOS</td>
</tr>
<tr>
<td>φ₄</td>
<td>MOS</td>
</tr>
<tr>
<td><strong>COMMAND</strong></td>
<td></td>
</tr>
<tr>
<td>NCBI(10)</td>
<td>MOS</td>
</tr>
<tr>
<td>NCBI(11)</td>
<td>MOS</td>
</tr>
<tr>
<td>NCBI(12)</td>
<td>MOS</td>
</tr>
<tr>
<td>NCBI(13)</td>
<td>MOS</td>
</tr>
<tr>
<td><strong>DATA</strong></td>
<td></td>
</tr>
<tr>
<td>DI(01,11,12,13)</td>
<td>TTL</td>
</tr>
<tr>
<td>DI(04,15,16,17)</td>
<td>TTL</td>
</tr>
<tr>
<td><strong>CONTROL</strong></td>
<td></td>
</tr>
<tr>
<td>ENCTL</td>
<td>MOS</td>
</tr>
<tr>
<td>NFLEN</td>
<td>TTL</td>
</tr>
<tr>
<td><strong>MISC</strong></td>
<td></td>
</tr>
<tr>
<td>NCOND</td>
<td>TTL</td>
</tr>
<tr>
<td>HOSCH</td>
<td>MOS</td>
</tr>
<tr>
<td>LOCCH</td>
<td>MOS</td>
</tr>
</tbody>
</table>

### Notes:
- A positive true logic convention is used for all signals except clocks. Signal names beginning with N are complemented signals.
- HOSCH at T4 and T5 is in the TRI-STATE high impedance output mode of CRQM load drivers.
- "1" (OUT) means CRQM is driving this node to the logic "1" level during the defined interval. For I/O lines the logic state is defined as "in" or "out." Input or output nodes are defined only as "1" or "0."
MOTOROLA 10800 RALU 4 BIT (MSI)

50 NS ECL IMPLEMENTATION
1 ACCUMULATOR, 1 INPUT LATCH
BCD FUNCTIONS

FAIRCHILD MACRO LOGIC SERIES

9405 RALU 4 BIT (MSI)
8 REGISTERS
ONE ALU INPUT "BARE"
9404 DATA PATH 4 BIT
SHIFTING, MULTIPLEXING, MASKING
9407 DATA ACCESS 4 BIT
ADDITION, MULTIPLEXING
HOLDING REGISTERS

SYBEX
CONTROL ROM

MICROCONTROL MC10801

TIMING MC10802

REGISTER SLICE

ALU SLICE MC10800

MEMORY INTERFACE MC10803

ADDRESS DATA

SYBEX
- ASSOCIATED TO SMS 360 IV BYTE
- AVAILABLE IN CARD AND MODULE FORM FROM SMS. MODIFIED CHIPS AVAILABLE FROM SIGNETICS
- BIPOLAR DESIGN, 48-PIN DIP
  300 TO 600 NS/INSTRUCTION
- 8 REGISTERS (ACCUMULATORS). NO STACK POINTER.
- 13-BIT PROGRAM COUNTER = 8 K ADDRESSING
- ORIENTED TOWARDS FAST SIGNAL PROCESSING WITH SMALL RAM
- NON-STANDARD INSTRUCTION SET

SYBEX
- 300 NS INSTRUCTION CYCLE
- 16-BIT MICROINSTRUCTIONS
- 13-BIT ADDRESS
- 8-BIT DATA
- 8 REGISTERS
- ON-CHIP OSCILLATOR
- BIPOLAR SCHOTTKY

SYBEX
PROGRAM STORAGE

ROM/PROM/RAM
TTL COMPATIBLE
UP TO 8K X 16 BITS
(82S115)

8X300
INTERPRETER

SYBEX
INTERNAL CONTROL SIGNALS

IV LATCHES

INTERNAL CONTROL SIGNALS

+5

GND
- Low-power Schottky

- 50 ns cycle time

- 10-bit address = 1K microinstruction addressability

- N-way branch

- 4-level stack

- Test and skip on input line

- 3-bit command code

- Auto-reset to address 0 during power-up

SYBEX
SYBEX
5. BIT-SLICE APPLICATIONS
IMP - 16
MM 300/600 (NOVA EMULATION)
INTEL DESIGN PROJECT 16 BIT CPU
INTEL DISK CONTROLLER
SPECIAL ARCHITECTURES:
  FLOATING POINT PROCESSOR
  LARGE SCALE COMPUTERS (CHIP REDUCTION/COST REDUCTION)
  HIGH LEVEL LANGUAGE INTERPRETERS
  CHARACTER STRING PROCESSORS
MICRO APPLICATIONS
  MULTI CHANNEL SERIAL I/O ADAPTOR
  SERIAL NIBBLE PROCESSOR
  LSI MICROCOMPUTER EMULATIONS

SYBEX
49 INSTRUCTIONS STANDARD CIRCUIT IN OPTIONAL CIRCUIT
MEMORY - UP TO 65K 16 BIT WORDS
PAGE ADDRESSING (256 WORDS) / INDEXING OUT OF PAGE
1.4 us MICROCYCLE
COMPREHENSIVE I/O BUSSING / CONDITIONAL JUMP ON I/O
### IMP 16P INSTRUCTION FORMATS

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Machine Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register to Register</td>
<td>OP sr dr OP NO USED OP</td>
</tr>
<tr>
<td>Register to Memory</td>
<td>OP r disp</td>
</tr>
<tr>
<td>Memory Reference (Class 1)</td>
<td>OP r xr disp</td>
</tr>
<tr>
<td>Memory Reference (Class 2)</td>
<td>OP xr disp</td>
</tr>
<tr>
<td>I/O and Miscellaneous</td>
<td>OP ctrl</td>
</tr>
<tr>
<td>Branch</td>
<td>OP cc disp</td>
</tr>
<tr>
<td>Control Flags</td>
<td>OP1 tc OP2 ctrl</td>
</tr>
<tr>
<td>Memory Reference (Double Word)</td>
<td>OP xr OP NOT USED</td>
</tr>
</tbody>
</table>

**Explanation of Symbols**

- **Op** - Instruction Mnemonic
- **OP** - Operation Code
- **sr** - Source Register Value
- **dr** - Destination Register Value
- **xr** - Index Register Value (2 or 3)
- **disp** - Displacement Value
- **cc** - Condition Code Value
- **r** - Register Value
- **ctl** - Control Bits Value
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Mnemonic</th>
<th>Execution Cycles</th>
<th>Memory Read Cycles</th>
<th>Memory Write Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Instructions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect</td>
<td>LD</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Indirect</td>
<td>ST</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Indirect</td>
<td>ADD</td>
<td>8</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Indirect</td>
<td>SUB</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Indirect</td>
<td>JMP</td>
<td>3</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Indirect</td>
<td>JSR</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect Indirect</td>
<td>JSR</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>ISZ</td>
<td>7,8 if SKIP</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>DSZ</td>
<td>8,9 if SKIP</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>SKAZ</td>
<td>6,7 if SKIP</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>SKG</td>
<td>Like Signs: 8,9 if SKIP</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Stack Register</td>
<td>PUSH</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Stack</td>
<td>PULL</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Indirect, Skip if Zero</td>
<td>AISZ</td>
<td>4,5 if SKIP</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Indirect</td>
<td>LI</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Indirect</td>
<td>CAI</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Copy</td>
<td>RCPY</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Register and Top of Stack</td>
<td>XCHRS</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Registers</td>
<td>RXCH</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td>RAND</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Exclusive OR</td>
<td>RXOR</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>RADD</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SHL</td>
<td>4 + 3K</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHR</td>
<td>4 + 3K</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROL</td>
<td>4 + 3K</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROR</td>
<td>4 + 3K</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>7</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>7</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR</td>
<td>8</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table above shows the instruction set for the IMP 10C processor, categorized by type and execution cycles. The memory read and write cycles are also specified. The instruction set includes basic arithmetic, logical, and control flow operations.
NOTE:
The symbol ▷ represents a pin or a group of pins on the IMP-18C card edge terminal.
NS MICROCYCLE
NS CYCLE MOS MEMORY

MM 600 PROCESSOR CARD

SYBEX
## MM 600: MACRO INSTRUCTION TIMING

<table>
<thead>
<tr>
<th>Instruction Mnemonic</th>
<th>Execution Time (µs)</th>
<th>Instruction Mnemonic</th>
<th>Execution Time (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISZ</td>
<td>3.0</td>
<td>JMP@</td>
<td>2.3</td>
</tr>
<tr>
<td>DSZ</td>
<td>3.0</td>
<td>NIOSCPU (INTEN)</td>
<td>3.5</td>
</tr>
<tr>
<td>JMP</td>
<td>1.0</td>
<td>NIOCCPU (INTDS)</td>
<td>3.5</td>
</tr>
<tr>
<td>JSR</td>
<td>1.9</td>
<td>DIACPU (READS)</td>
<td>3.5</td>
</tr>
<tr>
<td>LDA@</td>
<td>2.0</td>
<td>DOBCPU (MASKO)</td>
<td>3.5</td>
</tr>
<tr>
<td>STA</td>
<td>2.0</td>
<td>DIBCPU (INTA)</td>
<td>3.5</td>
</tr>
<tr>
<td>NIO</td>
<td>3.5</td>
<td>DICCUPU (IQRST)</td>
<td>3.5</td>
</tr>
<tr>
<td>SKPBN</td>
<td>3.2*</td>
<td>DOCCPU (HALT)</td>
<td>3.5</td>
</tr>
<tr>
<td>SKPBZ</td>
<td>3.2*</td>
<td>SKPBNCPU</td>
<td>3.2*</td>
</tr>
<tr>
<td>SKPDN</td>
<td>3.2*</td>
<td>DKBZCPU</td>
<td>3.2*</td>
</tr>
<tr>
<td>SKPDZ</td>
<td>3.2*</td>
<td>SKPDNCPU</td>
<td>3.2*</td>
</tr>
<tr>
<td>DIA,B,C</td>
<td>3.5</td>
<td>SKPDZCPU</td>
<td>3.2*</td>
</tr>
<tr>
<td>OOA,B,C</td>
<td>3.5</td>
<td>DMA IN</td>
<td>2.4</td>
</tr>
<tr>
<td>COM</td>
<td>1.3</td>
<td>DMA OUT</td>
<td>2.1</td>
</tr>
<tr>
<td>NEG</td>
<td>1.3</td>
<td>DMA INC</td>
<td>3.6</td>
</tr>
<tr>
<td>MOV</td>
<td>1.3</td>
<td>DMA ADD</td>
<td>4.7</td>
</tr>
<tr>
<td>INC</td>
<td>1.3</td>
<td>MULT</td>
<td>9.2</td>
</tr>
<tr>
<td>ADC</td>
<td>1.3</td>
<td>DIV</td>
<td>10.1</td>
</tr>
<tr>
<td>SUB</td>
<td>1.3</td>
<td>LDA@</td>
<td>3.0</td>
</tr>
<tr>
<td>ADD</td>
<td>1.3</td>
<td>STA@</td>
<td>3.0</td>
</tr>
<tr>
<td>AND</td>
<td>1.3</td>
<td>JMP@@</td>
<td>3.3</td>
</tr>
</tbody>
</table>

* Add 0.3 µs if skip occurs.

**SYBEX**
MM 600 CPU IS SIMILAR

SYBEX
INTEL DESIGN PROJECT CPU USING 3000 SERIES

16 BIT PROCESSOR

PIPELINED MICRO CONTROL
LOOK AHEAD CARRY
~200 NS CYCLE
~30 CHIPS

COMPREHENSIVE MACRO INSTRUCTION SET
IMPLEMENTED IN 256 MICRO INSTRUCTIONS

SYBEX
<table>
<thead>
<tr>
<th>OHL</th>
<th>1NH</th>
<th>2NH</th>
<th>3NH</th>
<th>4NH</th>
<th>5NH</th>
<th>6NH</th>
<th>7NH</th>
<th>8NH</th>
<th>9NH</th>
<th>ANH</th>
<th>BNH</th>
<th>CNH</th>
<th>DNH</th>
<th>ENH</th>
<th>FNH</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCC</td>
<td>JFL</td>
<td>JZJ</td>
<td>JFR</td>
<td>JFL</td>
<td>JFL</td>
<td>JFR</td>
<td>JFL</td>
<td>JCC</td>
<td>JFL</td>
<td>JCC</td>
<td>JCC</td>
<td>JCC</td>
<td>JCC</td>
<td>JCC</td>
<td>JCC</td>
</tr>
<tr>
<td>000H</td>
<td>0002H</td>
<td>000FH</td>
<td>0006H</td>
<td>0002H</td>
<td>0001H</td>
<td>0006H</td>
<td>0002H</td>
<td>0003H</td>
<td>0004H</td>
<td>0005H</td>
<td>0006H</td>
<td>0007H</td>
<td>0008H</td>
<td>0009H</td>
<td></td>
</tr>
<tr>
<td>001H</td>
<td>0012H</td>
<td>0015H</td>
<td>0016H</td>
<td>0018H</td>
<td>0019H</td>
<td>001FH</td>
<td>001CH</td>
<td>001AH</td>
<td>001BH</td>
<td>001CH</td>
<td>001CH</td>
<td>001CH</td>
<td>001CH</td>
<td>001CH</td>
<td></td>
</tr>
<tr>
<td>002H</td>
<td>0024H</td>
<td>0027H</td>
<td>0028H</td>
<td>002AH</td>
<td>002BH</td>
<td>002CH</td>
<td>002FH</td>
<td>002EH</td>
<td>002FH</td>
<td>002FH</td>
<td>002FH</td>
<td>002FH</td>
<td>002FH</td>
<td>002FH</td>
<td></td>
</tr>
<tr>
<td>003H</td>
<td>0032H</td>
<td>0035H</td>
<td>0036H</td>
<td>0038H</td>
<td>0039H</td>
<td>003AH</td>
<td>003CH</td>
<td>003BH</td>
<td>003CH</td>
<td>003CH</td>
<td>003CH</td>
<td>003CH</td>
<td>003CH</td>
<td>003CH</td>
<td></td>
</tr>
</tbody>
</table>

**SYBEX**
INTEL DISK CONTROLLER USING 3000 SERIES

CONTROLS UP TO FOUR DISKS (2310/5440)

DRIVE RATE 2.5 MHz
PERFORMS ALL DATA CHANNEL FUNCTIONS
STATUS CHECKING
COMMAND EXECUTION

16 BIT CPE ARRAY
400 NS CLOCK (NO PIPELINE, NO LOOK AHEAD)
SPECIAL NIBBLE TO WORD TO NIBBLE ARCHITECTURE
230 MICROINSTRUCTION IMPLEMENTATION (32 BIT)
67 CHIP ON 8 IN. X 15 IN. CARD

SYBEX
APPLICATION OF BIT-SLICE DEVICES TO "NUMBER CRUNCHING"

ARITHMETIC OPERATIONS ON FLOATING POINT DATA

HIGH LEVEL FUNCTIONS

LOGARITHMIC VECTOR

TRIGONOMETRIC MATRIX

HIGH LEVEL LANGUAGES: PL.-1, ALGOL, APL, FORTRAN

FLOATING POINT DATA (BINARY SCIENTIFIC NOTATION)

MANTISSA x 2 EXPONENT

EXAMPLE

<table>
<thead>
<tr>
<th>MANTISSA</th>
<th>S</th>
<th>EXP.</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>36 BITS</td>
<td></td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

>10 DECIMAL DIGITS ACCURACY

RANGE 2^-2048 TO 2^+2048 APPROX 10^-600 TO 10^+600
FLOATING POINT OPERATIONS

ADDITION

REQUIRES SCALING

\[ M \times 2^E + M' \times 2^{E'} = \quad \text{NOTE: MANTISSA IS CONSIDERED FRACTIONAL} \]

\( (\text{I.E. } M \ 1) \)

IF \( E = E' \) THEN

\[ M + M' \times 2^E \]

ELSE EQUALIZE THE EXPONENTS BY RAISING THE SMALLER/SCALING ITS FRACTION

LET \( E = E' + N \)

THEN \( M' \times 2^{E'} = (M' / 2^N) \times 2^E \)

\[ \text{NOTE: } M' / 2^N \text{ IS } M' \text{ SHIFTED DOWN } N \text{ BITS} \]

IN GENERAL, FLOATING POINT ADDITION:

LET \( E \ E' \) SUCH THAT \( E = E' + N \)

\[ M \times 2^E + M' \times 2^{E'} = M + (M' / 2^N) \times 2^E \]

THE SCALING OPERATION IS THE CRUX OF FLOATING ADD AND SUBTRACT

SYBEX
SUBTRACTION (AS IN ADDITION)

MULTIPLICATION

DOES NOT REQUIRE SCALING
ONLY HIGH ORDER PRODUCT IS USED

\[(M \times 2^E) \times (M' \times 2^{E'}) = (M \times M')_{\text{H.O.}} \times 2^{E+E'}\]

DIVISION

\[(M \times 2^E) / (M' \times 2^{E'}) = (M / M') \times 2^{E-E'}\]
FLOATING POINT NORMALIZATION

All floating point operations use normalized data and return normalized data.

Normalized = high order justified

I.e. the bit to the left of the sign ≠ sign

The last step of any floating point operation is normalization

I.e. shifting mantissa up until MSB ≠ sign

Adjusting exponent down to compensate

Multiply and divide require at most 1 shift to normalize

Add and subtract may require up to 34 shifts

(and must detect zero sum or difference)

SYBEX
FOR A 36 BIT MANTISSA

MULTIPLY AND DIVIDE TAKE 36 INTERATIONS:
ADD AND SUBTRACT MAY REQUIRE UP TO 34 BIT SHIFTS

EXAMPLE MULTIPLY

USING OPTIMAL 2901 CONFIGURATION FOR 36 BITS (W/O DBL LENGTH)

uCYLE = 220 NS (175 NS)
36 ITERATIONS FOR MULTIPLY = 7.92 uSEC (6.30 uSEC)
(MANTISSA ADDITION CONCURRENT)
PLUS SET UP AND RESTORE 5 CYCLES = 1.10 uSEC (.88 uSEC)

FLOATING MUL: 9.02 uSEC  ( OR 7.18 FOR FLOATING ONLY CPU )

SYBEX
REDUCTION FOR FLOATING MULTIPLY

FOUR STAGE MULTIPLICATION OF MANTISSAS

\[ M = M_H \times 2^{iH} + M_L \]
\[ M' = M'_H \times 2^{iH} + M'_L \]
\[ M \times M' = M_H \cdot M'_H \cdot 2^{iH} + (M_H \cdot M'_L + M'_H \cdot M_L) \cdot 2^{iH} + M_L \cdot M'_L \]

(Note: For close approximation one can ignore lower 18 bits of 2\(^{iH}\) product all 36 bits of 2\(^{iH}\) product)

TOTAL TIME REQUIRED

4 (or 3) Parallel 18 bit multiplies = 18 \times 220 \text{ NS} = 3.96 \text{ uSEC}
(18 \times 175 \text{ NS} = 3.15 \text{ uSEC})

Plus three 36 bit additions = 660 \text{ NS} (525 \text{ NS})

Four stage floating mul: 4.62 \text{ uSEC} (3.68 \text{ uSEC})
TRAIGHT FLOATING MULTIPLY

* AS IN CHAPTER 2

OUR STAGE FLOATING MULTIPLY

TWICE THE COST FOR TWICE THE SPEED

NOTE: 23 x 2901

SYBEX
COST OF SCALING IN FLOATING ADD

ADDITION

THE TIME OF ADDITION IN FLOATING ADD IS NEGLIGIBLE
1 CYCLE - 220 NS

WITHOUT SPECIAL DATA PATHS SCALING MAY TAKE
34 x 220 NS = 7.48 uSEC

EXponential shifts can reduce to
6 MAXIMUM x 220 = 1.32 uSEC (31 POSITIONS RIGHT, 5 POSITIONS LEFT)
SINGLE CYCLE SHIFT RIGHT 1, 4, 8, 16
SINGLE CYCLE SHIFT LEFT BY RotATING RIGHT AND DISABLING MUX ABOVE SIGNIFICANT NIBBLE
PLA DECODES EXPONENT DIFFERENCES FOR SCALING SEQUENCE
PLA DECODES LEADING ZERO NIBBLES FOR NORMALIZE SEQUENCE

HARDWARE REQUIRED

18 DUAL 4 LINE TO 1 LINE MUX (74S253)
1 QUAD 2 LINE TO 1 LINE MUX (74S257)
1 PLA

PLA INPUTS DRIVEN FROM EXPONENT ELEMENTS

TOTAL FLOATING ADD <2 USEC
FLOATING POINT SUMMARY — BIT SLICE APPLICATION

STRAIGHT FORWARD BIT SLICE F.P. IMPLEMENTATION FOR 36/12 DATA

MUL/DIV 7.2 TO 9 USEC
ADD/SUB 6.4 TO 8 USEC MAX

SPECIAL IMPLEMENTATION WITH BIT SLICES

MUL 3.7 TO 4.6 USEC
ADD/SUB 2 USEC MAX

THESE FIGURES AVAILABLE WITH CHIP COUNT UNDER 50 FOR DATA PATH ELEMENTS!

CONCLUSION: BIT SLICE DEVICES WILL CERTAINLY FIND THEIR WAY INTO LARGE SCALE COMPUTERS
CHIP REDUCTION

A 48 BIT FLOATING POINT CPU
WITH FAST FUNCTION ARCHITECTURES
WITH MAXIMUM SPEED, MAXIMUM PARALLEL CONTROLLER
(AS IN SECTION 2) WITH 1024 MICROINSTRUCTION ROM
WITH 48 BIT MEMORY/IO BUS

APPROXIMATELY 120 CHIPS
(WITHOUT BIT SLICE LSI: APPROXIMATELY 720 CHIPS)

SPEED

INHERENT CHIP SPEED
ARCHITECTURE ENHANCEMENTS
DISSIPATION REDUCTION

COST

NUMBER CRUNCHERS REDUCED TO MINI COMPUTER PRICES
PRESENTLY THE POPULATION OF COMPUTERS WITH FLOATING POINTS
100,000 WORLD WIDE

IS THERE ANY USE FOR A MINI PRICED F.P. MACHINE?

NAVIGATION REDUCTION FOR SHIPS AND SMALL PLANES
TRAINING SIMULATORS
COMPLEX FUNCTION MONITORS - INDUSTRIAL CONTROL, AIR CONTROL
PATTERN RECOGNITION APPLICATIONS
VOICE RECOGNITION
SCENE ANALYSIS
DISPLAY SYSTEMS INVOLVING
ROTATION
PERSPECTIVE
DISAPPEARANCE
MILITARY
BIT SLICE MACHINE CHARACTERISTICS

- FAST CYCLE
- POWERFUL MICROINSTRUCTION
- LARGE REGISTER COMPLEMENT

MAKE POSSIBLE MICRO LEVEL INTERPRETERS FOR
- APL
- ALGOL
- PL-1
- FORTRAN

EXAMPLE: 2000 CONTROL WORD VERSION OF CHAPTER 3 MACHINE COULD IMPLEMENT COMPLETE APL - SPEAKING MACHINE.

"POWER TO BURN" IMPLIES REDUCTION OF SOFTWARE DEVELOPMENT TIME

COST CRUCIAL IN LARGE SYSTEMS WHERE SOFTWARE/HARDWARE COST IS VERY HIGH
CHARACTER STRING PROCESSING

NARROW DATA: 8 BITS

SIMPLE FUNCTIONS
STRING SEARCH
STRING DELETE
STRING MODIFY
STRING INSERT

SPECIAL FUNCTIONS
CYCLIC REDUNDANCY CHECK
PARITY

OPTIMAL HARDWARE ALLOCATION
CHEAP PROCESSOR AND EXPENSIVE MEMORY
DESIRABLE CHARACTERISTICS:

ASYNCHRONOUS MEMORY ACCESS
MANY REGISTERS, ITERATIVE ACCESS
AUTOMATED LOOPS - LOW OVERHEAD
MASK FUNCTION

MULTI-BYTE PARALLEL MEMORY ACCESS
SUCH THAT (BYTES/FETCH) x (FETCH/SECOND) = \( \mu \) CYCLE/SECOND

MINIMUM \( \mu \) CYCLE (EFFECTIVE)

ALSO
POSSIBILITY OF MULTIPROCESSORS
CHARACTER STRING PROCESSING ARCHITECTURE

PROCESSOR INCLUDING BUFFERS APPROX 26 CHIPS
A - ADDRESS FUNCTIONS

0: A FIELD DIRECT TO $A_{\phi-3}$
1: LOAD COUNTER WITH A - FIELD
2: COUNTER TO 2901 $A_{\phi-3}$
3: COUNTER TO $A_{\phi-3}$ & INCR. CNTR.

SYBEX
PROCESSOR CYCLE = 150 NS
(NO END AROUND SHIFTS):

ITERATIVE ONE INSTRUCTION LOOPS UP TO 16 AT 150 NS/INSTR.
E.G. COMPARE CONTENTS OF 15 REGS TO ONE BYTE
SET LOOP COUNTER = 15
LOAD A COUNTER = 0

LOOP: CTR A(+1) | ADR B = 15 | A XOR B | BRANCH IF F=0

MAX 2.25 uSEC + .3 uSEC = 2.55 uSEC

SEARCH 8 CHARACTER SEGMENT FOR 1 OF 8 CHARACTERS
.3 + 8(.3 + 8(150)) = 12.3 uSEC

SYBEX
MULTIPLE SERIAL I/O ADAPTOR (UP TO 16 CHANNELS)

ONE REGISTER PER CHANNEL

120 ns CYCLE (NO ARITHMETIC)

ASSUME: 1 SERIAL IN PHASE
1 PARALLEL OUT PHASE
1 PARALLEL IN PHASE
1 SERIAL OUT PHASE

PER OPERATION CYCLE = 480 NS

RATE = 2 MHz/NO. OF CHANNELS
SERIAL PROCESSOR 1 CHIP

EXAMPLE: 16 REGISTER NIBBLE (4 BIT) MACHINE

APPROX 13 CHIPS
200 NS CYCLE
BIT SLICE EMULATIONS OF POPULAR MOS MICROPROCESSORS

BIPOLAR SPEED
PROGRAM COMPATIBILITY
I/O COMPATIBILITY

EXAMPLE
SIGNETICS 80E EMULATION OF 8080A

USES SIGNETICS (INTEL) 3000 SERIES
TWO TO TWELVE TIMES FASTER DEPENDING ON APPLICATION
6. DEVELOPMENT AIDS

SYBEX
1. ROM SIMULATOR

VARIABLE CONFIGURATION
PLUG IN SIMULATION OF ROMS
100 NS PERFORMANCE MAY REQUIRE SLOW CLOCK
PAPER TAPE LOADING -- TTY DUMPING
FRONT PANEL INSTRUCTION MODIFICATION AND VERIFICATION

In the system shown, the ROM Simulator simultaneously replaces the ROMS in both the microprogram and macroprogram store allowing the machine language code to be changed to find or patch a problem.
2. PROMS, ERoms, FPLAs

3. PORTABLE PROM AND PLA PROGRAMMERS

4. USER DEFINED ASSEMBLERS (E.G. RAPID)

SPEEDS DEFINITION OF MICROPROGRAM

CONVERTS SYMBOLIC PROGRAM TO ROM BIT PATTERNS

ACCEPTS USER DEFINITION OF

WORD LENGTH

FIELDS IN WORD, MULTIPLE FORMATS IF NECESSARY

VALID SYMBOLS

BINARY EQUIVALENT OF SYMBOLS

GENERATES ROM (TAPE OR CARD) IMAGE

RUNS ON IBM OR OTHER HOST SYSTEMS

VERY USEFUL IN CONJUNCTION WITH ROM SIMULATOR

SYBEX
5. DEVICE SIMULATORS (E.G. ICE - 30)

FOR DEBUGGING LOGICAL DESIGN OF SYSTEM
EMULATES DEVICE (E.G. 3001 INTEL) IN CIRCUIT
ALLOWS CONTROL AND MONITORING OF SIGNALS

SYBEX
- FROM DATA I/O

- PROGRAMS INTERSIL AND SIGNETICS

- INPUT:
  - KEYBOARD
  - MARK SENSE CARDS
  - ASCII TAPE
DATA I/O MODEL X ALLOWS UP TO 63 PRODUCT TERMS
7. CONCLUSION
BIT SLICES ARE A POWERFUL BUT FLEXIBLE UNIT OF INTEGRATION.

- WIDE RANGE OF APPLICATION COMPLEXITY

- WIDE RANGE OF PERFORMANCE

- SIGNIFICANT REDUCTIONS IN COST, CHIP COUNT AND DISSIPATION MADE POSSIBLE FOR GIVEN ARCHITECTURE

- SIGNIFICANT ENHANCEMENT IN COMPUTATIONAL POWER, SPEED, RELIABILITY FOR GIVEN COST / SPACE / COOLING

SYBEX
BIT - SLICE DEVICES WILL BECOME A STANDARD DESIGN ELEMENT.

1. ARCHITECTURES OF DEVICES ARE CONVERGING

2. SECOND SOURCES ARE PROLIFERATING

3. USERS ARE PROLIFERATING

4. CUSTOMER DEMANDS FOR MICROCOMPUTER HARDWARE ARE PROLIFERATING

5. BIT - SLICE DEVICES MERGE VERY WELL WITH OTHER LSI (PLA'S, ROM'S, RAM'S, I/O DRIVERS, INTERRUPT HANDLERS, ETC.)

SYBEX
SUMMARY

THIS COURSE HAS EXAMINED THE BIT-SLICE TECHNOLOGY IN RELATION TO:

THE EVOLUTION OF CPU DESIGN.

LSI CRITERIA.

THE DESIGN PROCESS OF A MODERN CPU.

OTHER APPLICATIONS FROM THE MICRO LEVEL TO THE NUMBER CRUNCHER LEVEL.

ITS INTEGRATION WITH OTHER LSI DEVICES

AND IT HAS PRESENTED A SURVEY OF THE EXISTING BIT-SLICE DEVICES AND SOME OF THE PRODUCTS IN WHICH THEY ARE USED.

SYBEX
<table>
<thead>
<tr>
<th>FAMILY CHARACTERISTICS</th>
<th>SSI GATES</th>
<th>FLIP-FLOPS</th>
<th>MSI ALU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Propagation Delay</td>
<td>Toggle Rate</td>
<td>4-Bit Add Time</td>
</tr>
<tr>
<td><strong>STANDARD TTL</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54/7400 Series SSI and MSI—112 types</td>
<td>10ns at 10mW</td>
<td>25 MHz</td>
<td>28 ns</td>
</tr>
<tr>
<td>8200 Series MSI—60 types</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8T Series Interface—36 types</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard &quot;gold doped&quot; TTL is the industry's longest selling digital logic family still in high volume production. New system designs generally favor the Low Power Schottky TTL equivalent functions.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LOW POWER SCHOTTKY TTL</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54/74LS00 Series SSI and MSI—79 types</td>
<td>10ns at 2mW</td>
<td>30 MHz</td>
<td>21 ns</td>
</tr>
<tr>
<td>3000 Series Microprocessor Set</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low power Schottky provides the same speed as standard TTL at 1/5 the power. The power savings and LSI potential are encouraging the use of 74LS in most new system designs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HIGH SPEED TTL</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54/74H00 Series SSI—30 types</td>
<td>6ns at 22mW</td>
<td>45 MHz</td>
<td>no MSI</td>
</tr>
<tr>
<td>8200 Series MSI—18 types</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Higher speed versions of standard TTL SSI devices. Generally being replaced by Schottky TTL in new designs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SCHOTTKY TTL</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54/74S00 Series SSI and MSI—55 types</td>
<td>3ns at 30mW</td>
<td>90 MHz</td>
<td>11ns</td>
</tr>
<tr>
<td>Schottky TTL uses a diode clamp design to insure the highest speed possible at TTL logic levels.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ECL</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10,00 Series SSI and MSI—69 types</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECL devices use a narrow logic swing to provide the highest speed standard logic family for use in large mainframe computers and test equipment.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TTL EQUIVALENT

The 5701/6701 is similar in function to the 25 TTL MSI packages listed below. It saves 375 I/O pins, 5.6 watts and 30 square inches of board area.

### TABLE 1

<table>
<thead>
<tr>
<th>Function</th>
<th>TTL#</th>
<th>#14 Pin Pkgs.</th>
<th>#16 Pin Pkgs.</th>
<th>#24 Pin Pkgs.</th>
<th>Advertised Gate Complexity (Each Pkg.)</th>
<th>Gate Complexity Total</th>
<th>Typical Power Each (Watts)</th>
<th>Total Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 x 9 &amp; 8 x 8 ROMs</td>
<td>7488</td>
<td>3</td>
<td></td>
<td></td>
<td>70</td>
<td>210</td>
<td>.50</td>
<td>1.50</td>
</tr>
<tr>
<td>16 x 4 Multiport RAM</td>
<td>74172</td>
<td></td>
<td>4</td>
<td></td>
<td>110*</td>
<td>440</td>
<td>.56</td>
<td>2.24</td>
</tr>
<tr>
<td>Arithmetic Logic Unit</td>
<td>74181</td>
<td></td>
<td></td>
<td>1</td>
<td>75</td>
<td>75</td>
<td>.55</td>
<td>.55</td>
</tr>
<tr>
<td>Storage Latches</td>
<td>7475</td>
<td>2</td>
<td></td>
<td></td>
<td>28</td>
<td>56</td>
<td>.16</td>
<td>.32</td>
</tr>
<tr>
<td>J-K Flip Flop (Q Reg)</td>
<td>74107</td>
<td>2</td>
<td></td>
<td></td>
<td>22</td>
<td>44</td>
<td>.10</td>
<td>.20</td>
</tr>
<tr>
<td>4 to 1 MUX</td>
<td>74153</td>
<td>6</td>
<td></td>
<td></td>
<td>16</td>
<td>96</td>
<td>.20</td>
<td>.40</td>
</tr>
<tr>
<td>O/I True Complement</td>
<td>74H87</td>
<td>2</td>
<td></td>
<td></td>
<td>18</td>
<td>36</td>
<td>.27</td>
<td>.54</td>
</tr>
<tr>
<td>Dual 4 Bit Select</td>
<td>74157</td>
<td>2</td>
<td></td>
<td></td>
<td>15</td>
<td>30</td>
<td>.15</td>
<td>.30</td>
</tr>
<tr>
<td>Quad 2 to 1 MUX with 3 State Outputs</td>
<td>745257</td>
<td>2</td>
<td></td>
<td></td>
<td>15</td>
<td>30</td>
<td>.30</td>
<td>.60</td>
</tr>
<tr>
<td>3 State Buffers</td>
<td>DM8094</td>
<td>1</td>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
<td>.18</td>
<td>.18</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td></td>
<td>20</td>
<td>5</td>
<td></td>
<td>1022</td>
<td></td>
<td>6.63</td>
<td></td>
</tr>
</tbody>
</table>

*The 74172 is advertised at 201 gate complexity but we are using only 2 of the 3 address capability, hence we have counted it as 110 gates.
APPENDIX C: CARRY LOOK-AHEAD LOGIC

ALU FUNCTIONS (AM 2901)

LOGIC FUNCTIONS FOR G, P, Cn+4, AND OVR

The four signals G, P, Cn+4, and OVR are designed to indicate carry and overflow conditions when the Am2901 is in the add or subtract mode. The table below indicates the logic equations for these four signals for each of the eight ALU functions. The R and S inputs are the two inputs selected according to Figure 2.

<table>
<thead>
<tr>
<th>1643</th>
<th>Function</th>
<th>( \hat{f} )</th>
<th>( \bar{g} )</th>
<th>( C_{n+4} )</th>
<th>OVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R + S</td>
<td>( P_2P_2P_1P_0 )</td>
<td>( G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 )</td>
<td>( C_4 )</td>
<td>( C_2 \lor C_4 )</td>
</tr>
<tr>
<td>1</td>
<td>S - R</td>
<td>Same as R + S equations, but substitute ( \bar{R}_1 ) for ( R_1 ) in definitions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>R - S</td>
<td>Same as R + S equations, but substitute ( \bar{S}_1 ) for ( S_1 ) in definitions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>R VS</td>
<td>LOW</td>
<td>( P_3P_2P_1P_0 )</td>
<td>( P_3P_2P_1P_0 + C_n )</td>
<td>( P_3P_2P_1P_0 + C_n )</td>
</tr>
<tr>
<td>4</td>
<td>R &amp; S</td>
<td>LOW</td>
<td>( G_3 + G_2 + G_1 + G_0 )</td>
<td>( G_3 + G_2 + G_1 + G_0 + C_n )</td>
<td>( G_3 + G_2 + G_1 + G_0 + C_n )</td>
</tr>
<tr>
<td>5</td>
<td>R &amp; S</td>
<td>LOW</td>
<td>Same as R &amp; S equations, but substitute ( \bar{R}_1 ) for ( R_1 ) in definitions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>R &amp; S</td>
<td>Same as R &amp; S, but substitute ( \bar{R}_1 ) for ( R_1 ) in definitions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RVS</td>
<td>( G_3 + G_2 + G_1 + G_0 )</td>
<td>( P_3G_3 + P_3P_2G_2 + P_3P_2P_1G_1 + P_3P_2P_1P_0 )</td>
<td>( P_3G_3 + P_3P_2G_2 + P_3P_2P_1G_1 + P_3P_2P_1P_0 (G_0 \land C_n) )</td>
<td>Complement of ( C_{n+4} ) at left</td>
</tr>
</tbody>
</table>

\( \lor \) = OR

SYBEX
<table>
<thead>
<tr>
<th>DEVICE</th>
<th>MODEL</th>
<th>DESCRIPTION</th>
<th>SIZE (BIT)</th>
<th>INPUT</th>
<th>OUTPUT</th>
<th>TEMPERATURE RANGE</th>
<th>MARKET</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMS</td>
<td>8220</td>
<td>8-bit CAM</td>
<td>4 x 2</td>
<td>OC</td>
<td>40</td>
<td>C</td>
<td>M,C</td>
</tr>
<tr>
<td></td>
<td>10155</td>
<td>16-bit CAM</td>
<td>8 x 2</td>
<td>OE</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAMS</td>
<td>82512</td>
<td>32-bit SAM</td>
<td>8 x 4</td>
<td>OC</td>
<td>35</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825112</td>
<td>32-bit SAM</td>
<td>8 x 4</td>
<td>TS</td>
<td>35</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>RAMS</td>
<td>82521</td>
<td>64-bit RAM</td>
<td>32 x 2</td>
<td>OC</td>
<td>50</td>
<td>C</td>
<td>M,C</td>
</tr>
<tr>
<td></td>
<td>82525</td>
<td>64-bit RAM</td>
<td>16 x 4</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>54/74S89</td>
<td>64-bit RAM</td>
<td>16 x 4</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>54/74S89</td>
<td>64-bit RAM</td>
<td>16 x 4</td>
<td>TS</td>
<td>35</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3101A</td>
<td>64-bit RAM</td>
<td>16 x 4</td>
<td>OC</td>
<td>35</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>82516</td>
<td>256-bit RAM</td>
<td>256 x 1</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>82517</td>
<td>256-bit RAM</td>
<td>256 x 1</td>
<td>OC</td>
<td>50</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825116</td>
<td>256-bit RAM</td>
<td>256 x 1</td>
<td>TS</td>
<td>40</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825117</td>
<td>256-bit RAM</td>
<td>256 x 1</td>
<td>OC</td>
<td>40</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>54/74S200</td>
<td>256-bit RAM</td>
<td>256 x 1</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>54/74S201</td>
<td>256-bit RAM</td>
<td>256 x 1</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>54/74S301</td>
<td>256-bit RAM</td>
<td>256 x 1</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10144</td>
<td>256-bit RAM</td>
<td>256 x 1</td>
<td>OE</td>
<td>30</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>82510</td>
<td>1024-bit RAM</td>
<td>1024 x 1</td>
<td>OC</td>
<td>45</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>82511</td>
<td>1024-bit RAM</td>
<td>1024 x 1</td>
<td>TS</td>
<td>45</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93415A</td>
<td>1024-bit RAM</td>
<td>1024 x 1</td>
<td>OC</td>
<td>45</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93425A</td>
<td>1024-bit RAM</td>
<td>1024 x 1</td>
<td>TS</td>
<td>45</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td>ROMS</td>
<td>825226</td>
<td>1024-bit ROM</td>
<td>256 x 4</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825229</td>
<td>1024-bit ROM</td>
<td>256 x 4</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825230</td>
<td>2048-bit ROM</td>
<td>512 x 4</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825231</td>
<td>2048-bit ROM</td>
<td>512 x 4</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825214</td>
<td>2048-bit ROM</td>
<td>256 x 6</td>
<td>TS</td>
<td>60</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8228</td>
<td>4096-bit ROM</td>
<td>1024 x 4</td>
<td>TTL</td>
<td>75</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825215</td>
<td>4096-bit ROM</td>
<td>512 x 8</td>
<td>TS</td>
<td>60</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td>PROMS</td>
<td>82523</td>
<td>256-bit PROM</td>
<td>32 x 8</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825123</td>
<td>256-bit PROM</td>
<td>32 x 8</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10139</td>
<td>256-bit PROM</td>
<td>32 x 8</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>82527</td>
<td>1024-bit PROM</td>
<td>256 x 4</td>
<td>OE</td>
<td>20</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825126</td>
<td>1024-bit PROM</td>
<td>256 x 4</td>
<td>OC</td>
<td>40</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825129</td>
<td>1024-bit PROM</td>
<td>256 x 4</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10149</td>
<td>1024-bit PROM</td>
<td>256 x 4</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825114</td>
<td>2048-bit PROM</td>
<td>256 x 8</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825130</td>
<td>2048-bit PROM</td>
<td>512 x 4</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825131</td>
<td>2048-bit PROM</td>
<td>512 x 4</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825115</td>
<td>4096-bit PROM</td>
<td>512 x 8</td>
<td>TS</td>
<td>60</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825136</td>
<td>4096-bit PROM</td>
<td>1024 x 4</td>
<td>TC</td>
<td>60</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825137</td>
<td>4096-bit PROM</td>
<td>1024 x 4</td>
<td>OC</td>
<td>60</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td>FPLA</td>
<td>825100</td>
<td>FPLA</td>
<td>16 x 48 x 8</td>
<td>TS</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>825101</td>
<td>FPLA</td>
<td>16 x 48 x 8</td>
<td>OC</td>
<td>50</td>
<td>M,C</td>
<td></td>
</tr>
</tbody>
</table>

*TEMPERATURE RANGE
C = Commercial (0°C to +75°C)
M = Military (55°C to -125°C)
All ECL 10,000 | 30°C to -85°C

**OUTPUT
TS = Tri State
OC = Open Collector
OE = Open Emitter
Signetics offers a broad line of MOS memories and memory-related products with many different access modes, architectures and speeds. Sizes range from 100 bits to 8192 bits on a single chip.

The memory products listed on the following pages can be categorized by both technology and product type. Basically, two technologies are used in the manufacture of Signetics MOS memory products. These are:

- P-Channel, Silicon Gate
- N-Channel, Ion-Implanted Silicon Gate

The P-Channel process is used in Signetics' 2500 series and the N-Channel is used in both the 2100 and 2600 series.

The 2500 series consists of shift registers—both static and dynamic, character generators and custom programmable ROMs, 256-bit static RAMs and 1103 type dynamic RAM.

The 2100 series is a family of 1024-bit static RAMs. This family consists of standard devices (2102), low power devices (21L02), high speed types (21F02), and military products (M2102). This series gives designer the choice of access times from 250-1.0us and power supply currents from 40mA.

The 2600 series consists of a 256 x 4 static F the 4096 x 1 dynamic RAMs, and a 1024 x 8 m programmable ROM. All N-Channel devices operate with a single 5V power supply and are compatible.

Most MOS memory products are available under Signetics Upgraded Product Reliability (SURE) Program and both products and processes are continually monitored under Signetics' SURE Reliability Program. Both programs are described elsewhere in this catalog.

A variety of packaging options are available for most MOS memory products. They are:

- A = 14-pin Silicon DIP
- B = 16-pin Silicon DIP
- D = 24-pin Silicon DIP
- F = 16-pin CERDIP
- I = Ceramic DIP
- K = 10-pin TO-100
- M = 10-pin TO-99
- N = 8-pin Silicon DIP
- V = 8-pin Silicon DIF
- X = 18-pin Silicon DI
- XA = 18-pin Silicon DI

### Memory Parts

<table>
<thead>
<tr>
<th>ORGANIZATION</th>
<th>SIGNETICS PART NUMBER</th>
<th>ACCESS TIME (ns)</th>
<th>POWER SUPPLY</th>
<th>PACKAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATIC RAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256 x 1</td>
<td>1101/2501</td>
<td>1000</td>
<td>+5, -9, -9</td>
<td>B, I</td>
</tr>
<tr>
<td>256 x 1</td>
<td>25L01</td>
<td>1000</td>
<td>+5, -12</td>
<td>B, I</td>
</tr>
<tr>
<td>256 x 4</td>
<td>2606</td>
<td>750</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>256 x 4</td>
<td>2602-1</td>
<td>500</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>2102/2602</td>
<td>1000</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>2102-1/2602-1</td>
<td>500</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>2102-2/2602-2</td>
<td>650</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>21F02</td>
<td>350</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>21F02-4</td>
<td>450</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>21L02</td>
<td>1000</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>21L02-1</td>
<td>500</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>21L02-2</td>
<td>650</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>21L02-3</td>
<td>400</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>M2102-4</td>
<td>450</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>M2102-6</td>
<td>650</td>
<td>+5, 0</td>
<td>B, F</td>
</tr>
<tr>
<td>DYNAMIC RAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024 x 1</td>
<td>1103</td>
<td>310</td>
<td>+16, 0, +19</td>
<td>XA, I</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>1103-1</td>
<td>180</td>
<td>+19, 0, +21</td>
<td>XA, I</td>
</tr>
<tr>
<td>4096 x 1</td>
<td>2680</td>
<td>200-350</td>
<td>+12, +5, 0, 5</td>
<td>I, F</td>
</tr>
<tr>
<td>4096 x 1</td>
<td>2660</td>
<td>200-350</td>
<td>+12, +5, 0, 5</td>
<td>I, F</td>
</tr>
<tr>
<td>STATIC ROM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512 x 8</td>
<td>2530</td>
<td>700</td>
<td>+5, 0</td>
<td>N, I</td>
</tr>
<tr>
<td>1024 x 8</td>
<td>2608</td>
<td>650</td>
<td>+5, 0</td>
<td>N, I</td>
</tr>
<tr>
<td>2048 x 4</td>
<td>2580</td>
<td>950</td>
<td>+5, 0</td>
<td>N, I</td>
</tr>
<tr>
<td>CHARACTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64 x 8 x 5</td>
<td>2^613</td>
<td>600</td>
<td>+5, 0, 12</td>
<td>N, I</td>
</tr>
<tr>
<td>64 x 6 x 8</td>
<td>2^516</td>
<td>600</td>
<td>+5, 0, 12</td>
<td>N, I</td>
</tr>
<tr>
<td>64 x 9 x 9</td>
<td>2^526</td>
<td>700</td>
<td>+5, 0, 12</td>
<td>N, I</td>
</tr>
</tbody>
</table>
## MOS Memories

### Static

<table>
<thead>
<tr>
<th>Organization/Register Length</th>
<th>Signetics Part Number</th>
<th>Speed (MHz)</th>
<th>Power Supply</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 x 6</td>
<td>2518</td>
<td>2.0</td>
<td>+5, -12</td>
<td>B, I</td>
</tr>
<tr>
<td>40 x 6</td>
<td>2519</td>
<td>2.0</td>
<td>+5, -12</td>
<td>B, I</td>
</tr>
<tr>
<td>50 x 2</td>
<td>2509</td>
<td>1.5</td>
<td>+5, -5, -12</td>
<td>A, K</td>
</tr>
<tr>
<td>80 x 4</td>
<td>2532</td>
<td>1.5</td>
<td>+5, 0, -12</td>
<td>A, K</td>
</tr>
<tr>
<td>100 x 2</td>
<td>2510</td>
<td>1.5</td>
<td>+5, -5, -12</td>
<td>A, K</td>
</tr>
<tr>
<td>128 x 2</td>
<td>2521</td>
<td>1.5</td>
<td>+5, -12</td>
<td>V</td>
</tr>
<tr>
<td>132 x 2</td>
<td>2522</td>
<td>1.5</td>
<td>+5, -12</td>
<td>V</td>
</tr>
<tr>
<td>200 x 2</td>
<td>2511</td>
<td>1.5</td>
<td>+5, -5, -12</td>
<td>A, K</td>
</tr>
<tr>
<td>240 x 2</td>
<td>2529</td>
<td>1.5</td>
<td>+5, -12</td>
<td>V</td>
</tr>
<tr>
<td>250 x 2</td>
<td>2528</td>
<td>1.5</td>
<td>+5, -12</td>
<td>V</td>
</tr>
<tr>
<td>256 x 2</td>
<td>2527</td>
<td>1.5</td>
<td>+5, -12</td>
<td>V</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>2533</td>
<td>1.5</td>
<td>+5, 0, -12</td>
<td>V</td>
</tr>
</tbody>
</table>

### Dynamic

<table>
<thead>
<tr>
<th>Organization/Register Length</th>
<th>Signetics Part Number</th>
<th>Speed (MHz)</th>
<th>Power Supply</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 x 2</td>
<td>2506/71/7</td>
<td>3.0</td>
<td>+5, -5</td>
<td>T, V</td>
</tr>
<tr>
<td>256 x 4</td>
<td>2502/1402</td>
<td>6.0</td>
<td>+5, -5</td>
<td>B, I</td>
</tr>
<tr>
<td>512 x 1</td>
<td>2505</td>
<td>2.5</td>
<td>+5, -5</td>
<td>K</td>
</tr>
<tr>
<td>512 x 1</td>
<td>2524</td>
<td>3.0</td>
<td>+5, -5</td>
<td>V</td>
</tr>
<tr>
<td>512 x 2</td>
<td>2503/1403</td>
<td>8.0</td>
<td>+5, -5</td>
<td>TA, V</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>2504/1404</td>
<td>8.0</td>
<td>+5, -5</td>
<td>TA, V</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>2525</td>
<td>3.0</td>
<td>+5, -5</td>
<td>V</td>
</tr>
<tr>
<td>1024 x 1</td>
<td>2512</td>
<td>2.5</td>
<td>+5, -5</td>
<td>K</td>
</tr>
</tbody>
</table>

### MOS Memories Cross Reference

<table>
<thead>
<tr>
<th>AMD</th>
<th>Signetics</th>
<th>G.I.</th>
<th>Signetics</th>
<th>M.I.I.</th>
<th>Signetics</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM1402A/2802</td>
<td>1402B</td>
<td>2509</td>
<td>2509K</td>
<td>MF1402A</td>
<td>2502B</td>
</tr>
<tr>
<td>AM1402APC</td>
<td>2502B</td>
<td>2510</td>
<td>2510A</td>
<td>MF1403A</td>
<td>2503TA</td>
</tr>
<tr>
<td>AM1403A/2803</td>
<td>1402TA</td>
<td>2511</td>
<td>2511A</td>
<td>MF1405A</td>
<td>2504TA</td>
</tr>
<tr>
<td>AM1404A</td>
<td>2503TA</td>
<td>2533</td>
<td>2533V</td>
<td>MF1406</td>
<td>2506T</td>
</tr>
<tr>
<td>AM1404A/2804</td>
<td>1404V</td>
<td></td>
<td></td>
<td>MF1407</td>
<td>2517T</td>
</tr>
<tr>
<td>AM1405A/2805</td>
<td>1405K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM1505T</td>
<td>1505V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM2505K</td>
<td>2506T/2517</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM2505K</td>
<td>2505K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM2806HC</td>
<td>2512K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM2806HC</td>
<td>2524V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM2806PC</td>
<td>2525V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM2809PC</td>
<td>2521V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM2833PC</td>
<td>2533V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InterSil</th>
<th>Signetics</th>
<th>Intersil</th>
<th>Signetics</th>
<th>M.I.I.</th>
<th>Signetics</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1402A</td>
<td>1402B</td>
<td>IM7712C</td>
<td>2512K</td>
<td>MF1402A</td>
<td>2502B</td>
</tr>
<tr>
<td>C1403A</td>
<td>1403TA/2503TA</td>
<td>IM7722C</td>
<td>2525V</td>
<td>MF1403A</td>
<td>2503TA</td>
</tr>
<tr>
<td>M1404A</td>
<td>2504TA</td>
<td>IM7780C</td>
<td>3347</td>
<td>MF1405A</td>
<td>2504TA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intel</th>
<th>Signetics</th>
<th>Intel</th>
<th>Signetics</th>
<th>M.I.I.</th>
<th>Signetics</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1402A</td>
<td>1402B/2502B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1403A</td>
<td>1403TA/2503TA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M1405A</td>
<td>1405K/2505K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fairchild</th>
<th>Signetics</th>
<th>Fairchild</th>
<th>Signetics</th>
<th>M.I.I.</th>
<th>Signetics</th>
</tr>
</thead>
<tbody>
<tr>
<td>3343</td>
<td>2521V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3344</td>
<td>2522V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3347</td>
<td>2532B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3349</td>
<td>2518B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3533</td>
<td>2533V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mostek</th>
<th>Signetics</th>
<th>Mostek</th>
<th>Signetics</th>
<th>M.I.I.</th>
<th>Signetics</th>
</tr>
</thead>
<tbody>
<tr>
<td>MK1007P</td>
<td>2532B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T.I.</th>
<th>Signetics</th>
<th>T.I.</th>
<th>Signetics</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS3112NC</td>
<td>2518B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMS3120NC</td>
<td>2532B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMS3128NC</td>
<td>2521V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMS3129NC</td>
<td>2522V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMS3133NC</td>
<td>2533V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# MOS Memories Cross Reference

## MOS N-Channel RAMs

<table>
<thead>
<tr>
<th>AMD</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2102</td>
<td>2102F</td>
</tr>
<tr>
<td>2102-1</td>
<td>2102-1F</td>
</tr>
<tr>
<td>2102-2</td>
<td>2102-2F</td>
</tr>
<tr>
<td>2102</td>
<td>2102B/2602B</td>
</tr>
<tr>
<td>2102-1</td>
<td>2102-1B/2602-1B</td>
</tr>
<tr>
<td>2102-2</td>
<td>2102-2B/2602-2B</td>
</tr>
<tr>
<td>M9102DC</td>
<td>21F02-4F</td>
</tr>
<tr>
<td>M9102PC</td>
<td>21F02-48</td>
</tr>
<tr>
<td>M91L02ADC</td>
<td>21L02-1F</td>
</tr>
<tr>
<td>M91L02ADM</td>
<td>M2102-41</td>
</tr>
<tr>
<td>M91L02APC</td>
<td>21L02-1B</td>
</tr>
<tr>
<td>M91L02BDC</td>
<td>21L02-3F</td>
</tr>
<tr>
<td>M91L02BPC</td>
<td>21L02-3B</td>
</tr>
<tr>
<td>M91L02DC</td>
<td>21L02-2F</td>
</tr>
<tr>
<td>M91L02DM</td>
<td>M2102-61</td>
</tr>
<tr>
<td>M91L02PC</td>
<td>21L02-2B</td>
</tr>
</tbody>
</table>

## MEMORIES

<table>
<thead>
<tr>
<th>INTEL</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2102</td>
<td>2102F</td>
</tr>
<tr>
<td>C2102-1</td>
<td>2102-1F</td>
</tr>
<tr>
<td>C2102-2</td>
<td>2102-2F</td>
</tr>
<tr>
<td>C2102A</td>
<td>21F02-2F</td>
</tr>
<tr>
<td>C2102A-2</td>
<td>21F02-2F</td>
</tr>
<tr>
<td>C2102A-4</td>
<td>21F02-4F</td>
</tr>
<tr>
<td>C2102A-4</td>
<td>21F02-4F</td>
</tr>
<tr>
<td>C2107A</td>
<td>26041</td>
</tr>
<tr>
<td>MC2102A-4</td>
<td>M2102-41</td>
</tr>
<tr>
<td>MC2102A-6</td>
<td>M2102-81</td>
</tr>
<tr>
<td>P2102</td>
<td>2102B/2602B</td>
</tr>
<tr>
<td>P2102-1</td>
<td>2102-1B/2602-1B</td>
</tr>
<tr>
<td>P2102-2</td>
<td>2102-2B/2602-2B</td>
</tr>
<tr>
<td>P2102A</td>
<td>21F02B</td>
</tr>
<tr>
<td>P2102A-2</td>
<td>21F02-2B</td>
</tr>
<tr>
<td>P2102A-4</td>
<td>21F02-4B</td>
</tr>
</tbody>
</table>

## MOSTEK

<table>
<thead>
<tr>
<th>MOSTEK</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MK4102</td>
<td>2102B/2602B</td>
</tr>
<tr>
<td>MK4102-1</td>
<td>2102-1B/2602-1B</td>
</tr>
</tbody>
</table>

## M.I.L.

<table>
<thead>
<tr>
<th>M.I.L.</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MF2102</td>
<td>2102B/2602B</td>
</tr>
<tr>
<td>MF2102-1</td>
<td>2102-1B/2602-1B</td>
</tr>
<tr>
<td>MF2102-2</td>
<td>2102-2B/2602-2B</td>
</tr>
</tbody>
</table>

## NATIONAL

<table>
<thead>
<tr>
<th>NATIONAL</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM2102D</td>
<td>2102B/2602B</td>
</tr>
<tr>
<td>MM2102-1</td>
<td>2102-1B/2602-1B</td>
</tr>
<tr>
<td>MM2102-2</td>
<td>2102-2B/2602-2B</td>
</tr>
</tbody>
</table>

## T.I.

<table>
<thead>
<tr>
<th>T.I.</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS4033</td>
<td>2602-2B</td>
</tr>
<tr>
<td>TMS4033JL</td>
<td>21F02-4F</td>
</tr>
<tr>
<td>TMS4033NL</td>
<td>21F02-48</td>
</tr>
<tr>
<td>TMS4034</td>
<td>2602-1B</td>
</tr>
<tr>
<td>TMS40334JL</td>
<td>2102-2F</td>
</tr>
<tr>
<td>TMS4034NL</td>
<td>2102-2B</td>
</tr>
<tr>
<td>TMS4035</td>
<td>2602B</td>
</tr>
<tr>
<td>TMS4035JL</td>
<td>2102F</td>
</tr>
<tr>
<td>TMS4035NL</td>
<td>2102B</td>
</tr>
</tbody>
</table>

## MOS P-Channel RAMs

<table>
<thead>
<tr>
<th>AMD</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM1101A</td>
<td>1101B</td>
</tr>
<tr>
<td>P1101A</td>
<td>2501B</td>
</tr>
<tr>
<td>P1101A</td>
<td>1101B</td>
</tr>
<tr>
<td>P1103</td>
<td>1103XA</td>
</tr>
<tr>
<td>P1101</td>
<td>2501B</td>
</tr>
</tbody>
</table>

## INTERSIL

<table>
<thead>
<tr>
<th>INTERSIL</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IM7501/11/12C</td>
<td>1101B/2501B</td>
</tr>
</tbody>
</table>

## INTEL

<table>
<thead>
<tr>
<th>INTEL</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1101A</td>
<td>2501B</td>
</tr>
<tr>
<td>P1101A1</td>
<td>1101B</td>
</tr>
<tr>
<td>P1103</td>
<td>1103XA</td>
</tr>
</tbody>
</table>

## MOSTEK

<table>
<thead>
<tr>
<th>MOSTEK</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MK4007</td>
<td>2501B/2501B</td>
</tr>
<tr>
<td>MK4007P</td>
<td>1101B</td>
</tr>
</tbody>
</table>

## M.I.L.

<table>
<thead>
<tr>
<th>M.I.L.</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MF1101A</td>
<td>1101B/2501B</td>
</tr>
<tr>
<td>MF1103</td>
<td>1103XA</td>
</tr>
</tbody>
</table>

## NATIONAL

<table>
<thead>
<tr>
<th>NATIONAL</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM1101A</td>
<td>2501B</td>
</tr>
<tr>
<td>MM1101A1</td>
<td>1101B</td>
</tr>
</tbody>
</table>

## MOS RAMS

<table>
<thead>
<tr>
<th>G.I.</th>
<th>SIGNETICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2580</td>
<td>2580N</td>
</tr>
</tbody>
</table>
TTL MEMORIES

RANDOM-ACCESS READ/WRITE MEMORIES

Bulletin No. DL S 515257 May 1975

<table>
<thead>
<tr>
<th>TYPE NUMBER (PACKAGES)</th>
<th>TYPE OF OUTPUT(S)</th>
<th>BIT SIZE (ORGANIZATIONS)</th>
<th>TYPICAL ACCESS TIMES</th>
<th>WRITE CYCLE TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>-55°C to 70°C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SN74S189, SN74S289</td>
<td>3-State</td>
<td>64 Bits</td>
<td>12 ns</td>
<td>25 ns</td>
</tr>
<tr>
<td>SN74S289 (J, D)</td>
<td>Open-Collector</td>
<td>(16 W x 4 B)</td>
<td>13 ns</td>
<td>65 ns</td>
</tr>
<tr>
<td>SN74S201 (J, D)</td>
<td>3-State</td>
<td>256 Bits</td>
<td>1024 Bits</td>
<td>20 ns</td>
</tr>
<tr>
<td>SN74S301 (J, D)</td>
<td>Open-Collector</td>
<td>(1024 W x 1 B)</td>
<td>1024 Bits</td>
<td>20 ns</td>
</tr>
</tbody>
</table>

Pin assignments for all of these memories are the same for all packages.

- Schottky-Clamped for High Performance
- Full On-Chip Decoding and Fast Chip-Enable Simplify System Decoding
- P-N-P Inputs Reduce Loading on System Buffers/Drivers
- Choice of 3-State or Open-Collector Outputs

description

These monolithic TTL memories feature Schottky clamping for high performance, a fast chip select access time to enhance decoding at the system level, and the 'S201 and 'S209 RAMs utilize inverted cell memory elements to achieve high densities. The memories feature p-n-p input transistors that reduce the low level input current requirement to a maximum of ~0.25 milliamperes, only one-eighth that of a Series 54S/74S standard load factor.

A three-state-output version and an open-collector-output version are offered for each of the three organizations. A three-state output offers the convenience of an open-collector output with the speed of a totem pole output; it can be bus-connected to other similar outputs, yet it retains the fast rise time characteristic of the TTL totem pole output. An open collector output offers the capability of direct interface with a data line having a passive pull-up.

write cycle

Information to be stored in the memory is written into the selected address (AD) location when the chip enable (CE) and the read/write (R/W) inputs are low. While the read/write input is low, the memory output(s) is(are) off (three-state = Hi-Z, open-collector = high). When a number of outputs are bus-connected, this off state neither loads nor drives the data bus; however, it permits the bus line to be driven by other active outputs or a passive pull-up.

read cycle

Information stored in the memory (see function table for input/output phase relationship) is available at the output(s) when the read/write input is high and the chip-enable input(s) is(are) low. When one (or more) chip-enable input(s) are high, the output(s) will be off.

Texas Instruments

8-10
TTL MEMORIES

TITANIUM· TUNGSTEN (Ti-W) Fuse Links for Fast, Low-Voltage, Reliable Programming

All Schottky-Clamped PROM's Offer:
- Fast Chip Select to Simplify System Decode
- Choice of Three-State or Open-Collector Outputs
- P-N-P Inputs for Reduced Loading on System Buffers/Drivers
- Full Decoding and Chip Select Simplify System Design
- Applications Include: Microprogramming/Firmware Loaders
  Code Converters/Character Generators
  Translators/Emulators
  Address Mapping/Look-Up Tables

| TYPE NUMBER (PACKAGES) | BIT SIZE (ORGANIZATION) | OUTPUT CONFIGURATION | TYPICAL ACCESS TIME (ns) FROM ADDRESS FROM CHIP SELECT |
|------------------------|-------------------------|---------------------|--------------------------------|--------------------------------|
| SN54186(J, WI) SN74186(J, N) | 512 bits (64 W x 8 B) | open-collector | 60 | 55 |
| SN54188A(J, WI) SN74188A(J, N) | 256 bits | open-collector | 30 | 34 |
| SN54188B(J, WI) SN74188B(J, N) | (32 W x 8 B) | open-collector | 25 | 12 |
| SN54328(J, WI) SN74328(J, N) | 1024 bits | three-state | 42 | 15 |
| SN54387(J, WI) SN74387(J, N) | (256 W x 4 B) | three-state | 50 | 20 |
| SN54407(J, WI) SN74407(J, N) | 2048 bits | open-collector | 50 | 20 |
| SN54472(J, WI) SN74472(J, N) | 4096 bits | three-state | 55 | 20 |
| SN54473(J, WI) SN74473(J, N) | (512 W x 8 B) | open-collector | 55 | 20 |

512 BITS (64 WORDS BY 8 BITS)

1286

256 BITS (128 WORDS BY 8 BITS)

1664A, 1568A, 5208

1024 BITS (256 WORDS BY 4 BITS)

2566, 5367

2048 BITS (256 WORDS BY 8 BITS)

5470, 5471

4096 BITS (1512 WORDS BY 8 BITS)

5472, 5473

Pin assignments for all of these memories are the same for all packages.

DESCRIPTION

These monolithic TTL programmable read-only memories (PROM's) feature titanium-tungsten (Ti-W) fuse links with each link designed to program in one millisecond or less. The Schottky-clamped versions of these PROM's offer considerable flexibility for upgrading existing designs or improving new designs as they feature full Schottky clamping for improved performance, low current MOS compatible P-N-P inputs, choice of bus driving three state or open collector outputs, and improved chip select access times.

The high complexity 2048- and 4096-bit PROM's can be used to significantly improve system density for fixed memories as all are offered in the 20 pin dual in line package having pin row spacings of 0.300 inch.
TTL MEMORIES

- Mask-Programmed Memories That Can Replace PROMs
- Full On-Chip Decoding and Fast Chip Select(s) Simplify System Decoding
- All Schottky-Clamped ROMs Offer
  - Choice of 3-State or Open-Collector Outputs
  - P-N-P Inputs for Reduced Loading on System Buffers/Drivers
- Applications Include:
  - Microprogramming Firmware/Firmware Loaders
  - Code Converters/Character Generators
  - Translators/Emulators
  - Address Mapping/Look-Up Tables

These monolithic TTL custom-programmed read-only memories (ROMs) are particularly attractive for applications requiring medium to large quantities of the same bit pattern. Plug-in replacements can be obtained for most of the popular TTL PROMs.

The high-complexity 2048-bit ROMs can be used to significantly improve system bit density for fixed memory as all are offered in compact 16- or 20-pin dual-in-line packages having pin-row spacings of 0.300-inch.

The Schottky-clamped versions offer considerable flexibility for upgrading existing designs or improving new designs as they feature improved performance; plus, they offer low-current MOS-compatible p-n-p inputs, choice of bus-driving three-state or open-collector outputs, and improved chip-select access times.

Data from a sequenced deck of data cards punched according to the specified format are permanently programmed by the factory into the monolithic structure for all bit locations. Upon receipt of the order, Texas Instruments will assign a special identifying number for each pattern programmed according to the order. The completed devices will be marked with the appropriate TI special device number. It is important that the customer specify not only the output levels desired at all bit locations, but also the other information requested under ordering instructions.

The three-state outputs offer the convenience of an open-collector output with the speed of a totem-pole output: they can be bus-connected to other similar outputs yet they retain the fast rise time characteristic of the TTL totem-pole output. The open-collector outputs offer the capability of direct interface with a data line having a passive pull-up.

Word-addressing is accomplished in straight positive-logic binary and the memory may be read when all CS inputs are low. A high at any CS input causes the outputs to be off.

Texas Instruments
Incorporated

8-12
APPENDIX E: INTEGRATED INJECTION LOGIC

I^2L BIPOLAR LOGIC

DESCRIPTION OF GATE

IMPLEMENTATION OF LOGIC

FABRICATION

INTERFACING

FROM TEXAS INSTRUMENTS
INTEGRATED INJECTION LOGIC, $I^2L$

$I^2L$ is a highly efficient new bipolar technology which reduces a basic gate function to a single current injected transistor switch. The logical simplicity of a single geometry gate, requiring no isolation, no load resistors and no ground metallization, achieves gate component densities 10 times those of conventional TTL or CMOS. $I^2L$ gates can be operated along a virtually constant speed/power product value over better than 5 magnitudes of injector current—from picoamps to microamps—at speeds ranging from hundreds of microseconds to tens of nanoseconds (Figure 21). They can be powered up for maximum speed then powered down without loss of function or data.

In addition, $I^2L$ gates are static, requiring no multiphase clocks, and are capable of stable operation in severe temperature environments.

![Figure 21. $I^2L$ Gate Performance Range](image-url)
The basic I^2L gate is a NPN grounded-emitter transistor switch as shown in Figure 22. Positive NAND logic is implemented, and logical isolation is accomplished by the use of multiple collector outputs (C1 and C2). When the base (B) is open or high and an injector (Ep) current source (I) is applied, the I^2L NPN transistor is normally used "on" (low-level output) by the current (I) supplied by a PNP current-injector transistor. Switching action is accomplished by the steering of this injector current. As shown on the inverter/buffer schematic, low input voltage to the base of less than one VBE (750mV) starts pulling injector current out of the input through the "on" (low) output of the driving gate. When the driving gate reaches a VCEsat, the input is robbed of its base drive, and the driven I^2L transistor/gate turn off causing its open-collector output to assume a high logic level. As with any open-collector logic, this output voltage level is determined by the load circuit or "pull-up" utilized. Internally, for a typical I^2L circuit design, this simply the clamp level at the input of the next stage, one VBE (750mV above ground). A high input logic level is achieved by default whenever a low-impedance path of less than one VBE potential is absent from the input. Deprived of a ground path of less than one VBE potential, the injector current will forward bias the I^2L transistor/gate "on" and produce an output low logic level one VCEsat above ground, typically 50mV. A typical I^2L internal logic swing of 700mV, from a VCEsat of 0mV to a VBE of +750mV, is thereby achieved by current steering of a NPN switch.

**FIGURE 22. BASIC I^2L GATE**

That the base region of the NPN transistor serves as an N-wide input, the number of steering inputs can be connected to the base. As all of these inputs are common to the base, and each is driven by a separate output source, logical ambiguity of the drivers can be avoided only if each driver exists as an individual source. Hence, the mechanism used is individual collectors for each load. Of the two collectors illustrated, C1 may be connected as the base of another gate. Simultaneously, this same base may be driven by the collector of another NPN gate.
IMPLEMENTING $I^2L$ LOGIC

Figure 23 displays the manner in which $I^2L$ transistors/gates are interconnected to perform logic. The NAND gate logic diagrammed is that of a common D-type flip flop. The schematic directly below it is the same D-type flip flop in $I^2L$ logic at a component count of one transistor/gate.

![Diagram of $I^2L$ D-type flip flop logic]

The logic diagram indicates that each gate input is a discrete entity; whereas, the $I^2L$ schematic shows that each base input has more than one driving source. The requirement for multiple isolated $I^2L$ collectors becomes quite evident as they identify the active and inactive driving sources; or, in other words, they isolate each discrete logical decision.

The common clock input to the two logic gates, drives two isolated inputs. Separate clock inputs, shown in the $I^2L$ schematic, would be driven from isolated (individual) collectors of the same npn transistor.
A section of two $I^2L$ gates is shown in Figure 24. The N+ circuit sub-
erves as both the mechanical base for fabrication and a common ground
or interconnection of all the grounded-emitter transistor/gates in a
thic $I^2L$ structure. One reason for $I^2L$ high densities is apparent here;
face metallization is required for ground interconnections as the entire
substrate serves this purpose. N epitaxial, grown on top of
substrate, provides both the grounded-emitter region of the vertical
etch and the grounded-base region of the lateral PNP injector. The
of two diffusions serves as both the P base region of the vertical NPN
collector of the lateral PNP injector. The second diffusion then com-
the $I^2L$ component geometries by providing the multiple-collector N+
ns of the vertical NPN. Metallization is then deposited and etched to
interconnection between $I^2L$ transistors/gates. Note that the lateral
integrated into the vertical NPN and therefore does not exist as a
de component. A symmetrical lateral PNP transistor can be utilized
current injector for multiple NPN/gates. Non-isolated $I^2L$ density is
enhanced by the fact that the single transistor gate requires no
ent isolation.

![Diagram of $I^2L$ structure]

**FIGURE 24. MONOLITHIC $I^2L$ STRUCTURE**

**SBP0400 ELECTRICAL CHARACTERISTICS**

00 input/output characteristics were selected with one objective in mind - full TTL
ility. The schematics and characteristics of the SBP0400 are shown in Figure 25.
circuit chosen is actually an RTL configuration modified for TTL compatibility.
threshold of nominally +1.5 volts is achieved by use of two 10K ohm resistors
ng as a voltage divider to boost the one $V_{BE}$ threshold of the input transistor to
input electrical characteristics are plotted as input current versus input vol-
e both the 10K and 20K ohm load lines and the threshold knee at +1.5 volts. The
dance, high-threshold characteristics were chosen to reduce input loading and
Increase the input noise margin over a standard TTL input yet retain full capability with virtually all 5 volt logic families. The I2L input also utilizes an input-clamping diode to limit negative excursions, "ringing", on the receiving end of a transmission line.

![Schematics of Equivalent Inputs, Outputs, Inputs/Outputs](image)

**Figure 25. Schematics of Equivalent Inputs, Outputs, Inputs/Outputs**

The output schematic is identical with that of an open-collector TTL circuit. The output characteristics are similar to that of the TTL output, but the I2L output demonstrates a considerably improved low-level output voltage, typically 0.06v, at rated load currents. Typical VOL versus IOL is shown in Figure 26. The output high logic level, as well as output rise times, and next stage input noise immunity are a function of the load circuit used. The load can be:

a. The input of the next stage if no source current is required.

b. A discrete pull-up resistor for greater noise immunity and improved rise times.

Common input/output configurations are also utilized for improved functional performance and increased packing densities. The schematic is recognizable as a "joining" of the separate input/output schematics and electrical characteristics already described.

![Input Current vs Input Voltage](image)

![Output Voltage vs Output Current](image)
APPENDIX F: FIELD PROGRAMMABLE LOGIC ARRAYS

SIGNETICS 82S100/82S101 DEVICE DESCRIPTION

FPLA MANUAL FUSER CIRCUIT
**DESCRIPTION**

82S100 (Tri-State Outputs) and the 82S101 (Open Collector Outputs) are Bipolar Programmable Logic Arrays, containing 48 Product terms (AND terms), and 8 Sum terms (OR terms). Each OR term controls an output function which can be programmed either true active-High (Fp), or true active-Low (Fo). The true state of each output function is activated by any logical combination of 16 input variables, or their complements, up to 48 terms. Both devices are field-programmable, which means that fuse patterns are immediately available by following the fusing procedure outlined in this data sheet.

82S100 and 82S101 are fully TTL compatible, and include chip-enable control for expansion of input variables, and output inhibit. They feature either Open Collector or Tri-State outputs for ease of expansion of product terms and application in is-organized systems.

**FEATURES**

- FIELD PROGRAMMABLE (Ni-Cr LINK)
- INPUT VARIABLES—16
- OUTPUT FUNCTIONS—8
- PRODUCT TERMS—48
- ADDRESS ACCESS TIME—50 ns, MAXIMUM
- POWER DISSIPATION—600mW, TYPICAL
- POWER DISSIPATION—600mW, TYPICAL
- POWER DISSIPATION—600mW, TYPICAL
- POWER DISSIPATION—600mW, TYPICAL
- INPUT LOADING—(−100µA), MAXIMUM
- OUTPUT OPTION:
  - TRI-STATE OUTPUTS—82S100
  - OPEN COLLECTOR OUTPUTS—82S101
- OUTPUT DISABLE FUNCTION:
  - TRI-STATE—Hi-Z
  - OPEN COLLECTOR—Hi

**APPLICATIONS**

- LARGE READ ONLY MEMORY
- RANDOM LOGIC
- CODE CONVERSION
- PERIPHERAL CONTROLLERS
- LOOK-UP AND DECISION TABLES
- MICROPROGRAMMING
- ADDRESS MAPPING
- CHARACTER GENERATORS
- SEQUENTIAL CONTROLLERS

**PIN CONFIGURATION**

![PIN CONFIGURATION Diagram](image)

**FPLA EQUIVALENT LOGIC PATH**

![FPLA EQUIVALENT LOGIC PATH Diagram](image)

**NOTE**

For each of the 8 outputs, either the function Fp (active high) or Fo (active low) is available, but not both. The required function availability is user programmable via fuse 1B.
TABLE

\[ \sum (k_{i1}k_{i2}k_{i3})^* \quad : \quad k = 0, 1, X \text{ (Don't Care)} \]
\[ n = 0, 1, 2 \]

<table>
<thead>
<tr>
<th>MODE</th>
<th>( P_n )</th>
<th>( \bar{C}E )</th>
<th>( S_{7}^{2} f(P_n) )</th>
<th>( F_p )</th>
<th>( \bar{F}_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### ABSOLUTE MAXIMUM RATINGS

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>RATING</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vcc</td>
<td>+7</td>
<td>Vdc</td>
</tr>
<tr>
<td>V_in</td>
<td>+5.5</td>
<td>Vdc</td>
</tr>
<tr>
<td>V_out</td>
<td>+5.5</td>
<td>Vdc</td>
</tr>
<tr>
<td>T_a</td>
<td>0° to +75°</td>
<td>°C</td>
</tr>
<tr>
<td>T_mf</td>
<td>-65° to +150°</td>
<td>°C</td>
</tr>
</tbody>
</table>

### ELECTRICAL CHARACTERISTICS 0°C ≤ T_a ≤ 75°C; 4.75V ≤ Vcc ≤ 5.25V

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>LIMITS</th>
<th>UNIT</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vih</td>
<td>Vcc = 5.25V</td>
<td>MIN</td>
<td>2</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYP*1</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>V_il</td>
<td>Vcc = 4.75V</td>
<td>MAX</td>
<td>0.2</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_cl</td>
<td>Vcc = 4.75V, I_cm = -18mA</td>
<td>MIN</td>
<td>1.2</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYP*1</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>V_HiL</td>
<td>Vcc = 4.75V, I_hi = -2mA</td>
<td>MAX</td>
<td>2.4</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_OL</td>
<td>Vcc = 4.75V, I Olson = 9.6mA</td>
<td>MIN</td>
<td>0.35</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYP*1</td>
<td>0.45</td>
<td>V</td>
</tr>
<tr>
<td>Ios</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ioc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Icc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C_in</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C_o</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SWITCHING CHARACTERISTICS 0°C ≤ T_a ≤ 75°C, 4.75V ≤ Vcc ≤ 5.25V

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>LIMITS</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagation Delay</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_oca</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_ce</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. All voltage values are with respect to network ground terminal.
2. All typical values are at Vcc = 5V, T_a = 25°C.
3. Duration of short circuit should not exceed one second.
4. Icc is measured with the chip enable input grounded, all other inputs at 4.5V and the outputs open.
5. Measured with Vcc applied to CE and a logic "1" stored.
6. Measured with Vin applied to CE.
7. Test each output one at a time.
8. Measured with a programmed logic condition for which the output is at a "0" logic level. Output sink current is supplied through a resistor to Vcc.
AMMING SPECIFICATIONS (Testing of these limits may cause programming of device.) $T_a = +25^\circ C$

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>LIMITS</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CC}$ Supply (Program &quot;OR&quot;)</td>
<td>$I_{CCS} = 550 \text{ mA, min.}$ (Transient or steady state)</td>
<td>8.5</td>
<td>8.75</td>
</tr>
<tr>
<td>$V_{CC}$ Supply (Program Output Polarity)</td>
<td></td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{CC}$ Limit (Program &quot;OR&quot;)</td>
<td>$V_{CCS} = +8.75 \pm 0.25V$</td>
<td>550</td>
<td>1,000</td>
</tr>
<tr>
<td>Output Voltage (Program Output Polarity)</td>
<td>$I_{OPH} = 300 \pm 25\text{mA}$</td>
<td>16.0</td>
<td>17.0</td>
</tr>
<tr>
<td>Output Voltage (Idle)</td>
<td></td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>Output Current Limit (Program Output Polarity)</td>
<td>$V_{OPH} = +17 \pm 1V$</td>
<td>275</td>
<td>300</td>
</tr>
<tr>
<td>Input Voltage (Logic &quot;1&quot;)</td>
<td>$V_{IH} = +5.5V$</td>
<td>50</td>
<td>50 µA</td>
</tr>
<tr>
<td>Input Voltage (Logic &quot;0&quot;)</td>
<td>$V_{IL} = 0V$</td>
<td>-500</td>
<td>500 µA</td>
</tr>
<tr>
<td>Forced Output (Logic &quot;1&quot;)</td>
<td></td>
<td>2.4</td>
<td>5.5</td>
</tr>
<tr>
<td>Forced Output (Logic &quot;0&quot;)</td>
<td></td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>Output Current (Logic &quot;1&quot;)</td>
<td>$V_{OH} = +5.5V$</td>
<td>100</td>
<td>100 µA</td>
</tr>
<tr>
<td>Output Current (Logic &quot;0&quot;)</td>
<td>$V_{OL} = 0V$</td>
<td>-1</td>
<td>1 mA</td>
</tr>
<tr>
<td>CE Program Enable Level</td>
<td>$V_{CE} = +10V$</td>
<td>2.5</td>
<td>2.5 mA</td>
</tr>
<tr>
<td>Input Variables Current</td>
<td>$V_{IX} = +10V$</td>
<td>5.0</td>
<td>5.0 mA</td>
</tr>
<tr>
<td>CE Input Current</td>
<td>$V_{IX} = +10V$</td>
<td>5.0</td>
<td>5.0 mA</td>
</tr>
<tr>
<td>FE Supply (Program)</td>
<td>$I_{FEH} = 300 \pm 25\text{mA}$ (Transient or steady state)</td>
<td>16.0</td>
<td>17.0</td>
</tr>
<tr>
<td>FE Supply (Idle)</td>
<td></td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>FE Supply Current Limit</td>
<td>$V_{FEH} = +17 \pm 1V$</td>
<td>275</td>
<td>300</td>
</tr>
<tr>
<td>$V_{CC}$ Supply (Program &quot;AND&quot;)</td>
<td>$I_{CCP} = 550 \text{ mA, min.}$ (Transient or steady state)</td>
<td>4.75</td>
<td>5.0</td>
</tr>
<tr>
<td>$I_{CC}$ Limit (Program &quot;AND&quot;)</td>
<td>$V_{CCP} = +5.0 \pm 0.25V$</td>
<td>550</td>
<td>1,000</td>
</tr>
<tr>
<td>Forced Output (Program)</td>
<td></td>
<td>9.5</td>
<td>10</td>
</tr>
<tr>
<td>Output Current (Program)</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Output Pulse Rise Time</td>
<td></td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>CE Programming Pulse Width</td>
<td></td>
<td>1</td>
<td>1.5</td>
</tr>
<tr>
<td>Pulse Sequence Delay</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Programming Time</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Programming Duty Cycle</td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Fusing Attempts per Link</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Verify Threshold</td>
<td></td>
<td>0.9</td>
<td>1.0</td>
</tr>
</tbody>
</table>
GNETICS BIPOLAR FIELD-PROGRAMMABLE LOGIC ARRAY • 82S100, 82S101

TEST FIGURE AND WAVEFORMS

INM =-----------------I

CE

18V

VCC

INPUT

15V

15V

15V

15V

VOL

VDD

OUT

VCC

INCLDES SCOPE AND JIG CAPACITANCE

ALL INPUTS $t_i = t_f = 500$ (10% TO 90%)

PICAL FUSING PATHS

OR Matrix

AND Matrix

OUTPUT POLARITY.
Definitions

The RAPID assembly language consists of six kinds of statements. In this section we will begin the definition of these statements by introducing the character set out of which the statements are formed, the notation in which we will define elements of statements, and the language primitives common to all the statements.

The character set RAPID recognizes may be viewed as having three parts; alphabetic, numeric, and special symbols. The alphabetic part contains the twenty-six letters, A, B, C, . . . , Z. The ten digits 0, 1, 2, . . . , 9 comprise the numeric part. The special symbols are eleven characters used for punctuation and operator names; ampersand (&), apostrophe ('), asterisk (*), comma (,), dollar sign ($), equals (=), minus (-), parentheses ( () ), plus (+), and slash (/). The uses of the characters are fully described in the definitions.

Statement definitions will be given in the form of syntax equations. That is, statements will be divided (parsed, really) into syntactic elements and each of the elements defined. These elements will often be further divided into more primitive syntactic elements and those elements defined. This process is repeated until the definitions finally resolve to the character set described above.

The definitions accomplish three things:

1. They name the syntactic elements of each statement.
2. They specify the order in which those elements occur in the statements.
3. They indicate what characters are used to form each element.

The syntax equations possess their own notation. These symbols are not part of the character set used to express statements, but merely give form to the definitions. This notation is as follows:

<> The left and right acute brackets enclose the names of syntactic elements.

:= This symbol separates the name of an element from its definition; it means "is defined as".

| The vertical bar indicates an alternative definition; that is, the elements on either side of the bar are equally valid.
The syntactic elements enclosed in braces may occur at this place in the statement zero or more times, up to \( n \) times maximum; if no subscript is given, either there is no limit to the repetitions, or it depends on other variables.

The syntactic element enclosed in brackets begins in card column \( n \).

In example, we can define the character set for statements:

\[
\begin{align*}
alphabetic & : = \{A, B, C, \ldots, X, Y, Z\} \\
numeric & : = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
special \text{ symbol} & : = \{\&'\ast, $=\,(\leq\geq\leq\ast\leq\ast/\}
\end{align*}
\]

These equations give a name to each of the 47 characters. Since we are interested only in what set of characters applies to each name, the elements given as alternatives; 2 is as valid a numeric as 9 or $ is as valid a special symbol as *. Whenever a character or sequence of characters occurs side the acute brackets, it is a literal (it represents itself) and should written in a statement exactly as shown.

name of a syntactic element may be used in a definition as well as literals. s is in fact how statement definitions are built from more primitive definitions. For instance:

\[
alphanumeric : = \langle \text{alphabetic} \rangle \mid \langle \text{numeric} \rangle
\]

ines an alphanumeric character as any one of thirty-six alphabetics and numerics.

many definitions, we will be interested in the order of elements. To specify articular order, we merely write one element after the other without any ervening vertical bar. Thus:

\[
symbol : = \langle \text{alphabetic} \rangle \left\{ \langle \text{alphanumeric} \rangle \right\}_4
\]

ifies that the element known as a symbol always begins with an alphabetic character and may be followed by as many as four alphanumerics. Examples of symbols are A, PQ, B101, and ZOZ1. 5E and ABCDEF would not be valid. RAPID will detect any violation of these syntax definitions and signal an error on e assembly listing.
The rest of the primitives deal with the definition of constants:

\[
\text{〈constant〉} : = \text{〈integer〉} | \text{ˈA} \{\text{〈alphabetical〉} \} 5 | \text{ˈB} \{\text{〈binary〉} \} 128 | \text{ˈH} \{\text{〈hexadecimal〉} \} 32 | \text{ˈO} \{\text{〈octal〉} \} 43
\]

\[
\text{〈integer〉} : = \text{〈numeric〉} \{\text{〈numeric〉} \} 8
\]

\[
\text{〈binary〉} : = 0 | 1
\]

\[
\text{〈hexadecimal〉} : = \text{〈numeric〉} | A | B | C | D | E | F
\]

\[
\text{〈octal〉} : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
\]

Constant, as will be seen later, has several very specific uses. Quite ten a constant is a decimal number, the integer defined above, of no more an nine digits. Alphabetic, binary, hexadecimal, and octal constants are ecified by prefixing an apostrophe and A, B, H, or 0 to the constant. Con­ants must fit in whatever field they are designed for; the maximum lengths ecified here will rarely, if ever, be required. Examples of constants are: 177, 'HFF, 899, 'B101, and 'AXY. Note that 'B101 is the three-bit constant ual to 5, and B101 without the apostrophe is a symbol. The primitives of e RAPID assembly language, then, are symbols and constants.

This discussion of statements proceeds in the following chapters, eachoup of syntax equations is followed by an explanation and several examples clarify the definitions. This manner of defining statements, a modifica­on of the Backus Normal Form, is quite precise. Once a reader becomes quainted with it, he should find it easy and unambiguous to use as a re­ence.

\section*{Library of Statements}

This section briefly introduces the six statement types. The definitions for h type will be given completely at the beginning of their respective pters.
\[
\begin{align*}
\langle \text{RAPID input} \rangle & : = \langle \text{option statement} \rangle \langle \text{parity statement} \rangle \\
& \quad \langle \text{format statements} \rangle \langle \text{opcodes statement} \rangle \\
& \quad \langle \text{program section} \rangle \langle \text{end statement} \rangle \\
\end{align*}
\]
\]

The statements must appear in the specified order.

\[
\langle \text{option statement} \rangle : = \langle [\$ROM] \rangle \langle \text{option specifications} \rangle
\]

The option specifications describe the bit storage configurations of ROM, its mapping into words, and call for various assembly outputs.

\[
\langle \text{parity statement} \rangle : = \langle [\$PARITY] \rangle \langle \text{parity specifications} \rangle
\]

Parity specifications describe how parity bits, if any are desired, will be generated in each output word. For example, this permits the designer to indicate he wants bit 5 in every word to reflect odd parity.

\[
\langle \text{format statements} \rangle : = \langle [\$FORMATS] \rangle \langle \text{format specifications} \rangle
\]

These statements name and describe the various formats that output words can take and set up the correspondence between input fields and output fields. For example, one statement might specify that the symbol in the third input field (in a program statement naming a given format) be translated into a code which will appear in bits 12 through 15 in the output word.

\[
\langle \text{opcodes statement} \rangle : = \langle [\$OPCODES] \rangle \langle \text{opcode specifications} \rangle
\]

The opcodes specifications list the operation codes in symbolic form and the machine language representations into which they should be translated. For example, it could instruct RAPID to translate the opcode ADD into the binary code 1010.

\[
\langle \text{program section} \rangle : = \langle [\$PROGRAM] \rangle \langle \text{program statements} \rangle
\]

The program section contains the symbolic program for ROM as a series of statements. Using the previous four kinds of statements, RAPID interprets program statements and translates them to machine language.

\[
\langle \text{end statement} \rangle : = \langle [\$END] \rangle
\]

The end card terminates the input to RAPID. It contains no other information.

Each of these six statements begins with $ in column one. The option, parity, and end statements must each be given on one card. The others may use as many cards as desired. At least one blank must follow the words ROM, PARITY, FORMATS, and OPCODES, and the names of formats in format and program statements.
OPTION STATEMENT

The option statement provides RAPID with information about the configuration of ROM modules, the width of the program word, and the types of output generated from the assembler.

\[
\text{option specifications} : = \{ \text{option} \} \}
\]

\[
\text{option} : = \text{CHIP} = \{ \text{depth} \} \cdot \{ \text{width} \} \}
\]

- \text{WORD} = \{ \text{word size} \}
- \text{LIST}
- \text{MASK}
- \text{MEMORY} = \{ \text{table type} \}
- \text{NULL} = \{ \text{binary} \}

\[
\text{depth} : = \{ \text{integer} \}
\]

\[
\text{width} : = \{ \text{integer} \}
\]

\[
\text{word size} : = \{ \text{integer} \}
\]

\[
\text{table type} : = \text{COMPACT} \}
- \text{EXTENDED}
\]

The 8 shows a number of legitimate option statements. This statement may have continuation cards. One option is separated from the next by a \}; between the word ROM and the first option is at least one blank.

The CHIP option describes to RAPID the configuration of the memory element to implement the control store. For example, if a 256-bit semiconductor (Random Access Memory) is used and it is organized 128 by 2 bits, the configuration would be CHIP = 128*2. In the case of a 4096 bit ROM (Read Memory) it might be CHIP = 4096*1, CHIP = 1024*4, CHIP = 512*8, etc.

The width may be no greater than 64 bits.

The word option specifies the number of bits of a word in ROM. The maximum size is 128 bits. RAPID will assume as many chips side-by-side as necessary to provide the specified word size.

The output options, LIST and MASK control listing and punched card output, respectively. LIST causes all of the input to be listed. Any cards in which RAPID detects errors will be listed with a diagnostic message, whether selected or not.
PARITY STATEMENT

Output words from a control store may contain one or more parity bits to validate the contents of the words. The parity statement describes to RAPID what parity bits, if any, to generate for each word.

\[
\text{(parity specifications)} : = \text{(parity declaration)} \{, \text{(parity declaration)}\}^4 | \text{NONE}
\]

\[
\text{(parity declaration)} : = \text{(parity type)} (\text{(bit position)}) = \text{(field)}\{+ \text{(field)}\}
\]

\[
\text{(parity type)} : = \text{ODD} | \text{EVEN}
\]

\[
\text{(bit position)} : = \text{<integer>}
\]

\[
\text{(field)} : = (\text{(left bit)}/\text{(length)})
\]

\[
\text{(left bit)} : = \text{<integer>}
\]

\[
\text{(length)} : = \text{<integer>}
\]

The parity specifications must occur on one card; hence, no more than five may be given. Each declaration is separated from the next by a comma, and the first from the word PARITY by at least one blank. If no parity bits are to be generated, the word NONE is written.

For each specified parity bit, its type, odd or even, is indicated as well as the fields over which parity is computed. A field is defined by giving the number of the left-most bit position of the field and the total number of bits in the field. For instance, the three fields shown in the word in Figure 10 would be specified as (6/6), (13/11), and (12/1). Fields may overlap one another, but not the parity bit positions themselves.

Figure 11 shows several examples of parity statements for a 64-bit word. The last example defines six fields. Bits 1-16 contribute to even parity in bit 0; bits 18-21, 26-35 and 55-63 contribute to odd parity in bit 54; and bits 22-25 and 36-53 contribute to odd parity in bit 17. Notice that parity declarations may be given in any order in the statement.
FORMAT STATEMENTS

Format statements establish the correspondence between input fields and fields. By careful use of these statements and the opcodes statement (see Opcodes Statements), the designer can define a convenient for expressing his control program.

mat specifications
: = [ format id ] , format definition
    
    , format definition ]

mat id
: = alphabetic , alphabetic , alphabetic

mat definition
: = field ( field type )

field type
: = A | O | T

er of formats may be specified. Each format is a separate statement 
s in column one with the name of the format, up to five characters. 
t may have continuation cards, though a format definition may not 
 across cards. That is, a right parenthesis and comma must complete 
definition before continuing the statement on the next card. The 
ction of format statements is headed by the $FORMATS statement card. 
 is an example of a format section.

programmed systems often use several control word or microinstruction 
Each format is independent of the others, though, of course, they 
y to words of the same length as defined in the option statement. 
e 12, the three formats named CMD, TEST, and JUMP clearly identify 
action type.

definition describes one field. A field is specified as shown in 
ious section; field type must be Address, Opcode, or Truncated address. 
statement may contain any number of definitions in any order. How-
elds must not overlap one another or the parity bits, and they must 
in the ROM words.

point, it is important to understand that each program statement 
er to a format. It is that named format which will guide the trans-
 of the program statement. Information in the program statement must 
ed with the fields of its format. This matching is accomplished in 
. An opcode is matched against a list in the opcodes statement (see 
on Opcodes Statement); that list points to one of the fields of the 
and the value associated with that opcode is placed in that field. 
ively, an address expression in the program statement itself points 
 of the fields of the format; the result of the expression fills that 
The Program Statements Section describes fully this matching process 
kind of information in program statements.
OPCODES STATEMENT

OPCODES are the mnemonic symbols the programmer may use in program statements to direct that specific fields in the output word be given certain values. The opcodes statement lists all the valid symbols for each opcode field of each format and their associated values.

\[
\text{<opcode specifications>} : = \text{<opcode field>} \{ \text{<opcode field>} \}
\]

\[
\text{<opcode field>} : = \text{<format id>} \text{<field number>} = ( \text{<opcode>} \{, \text{<opcode>} \})
\]

\[
\text{<field number>} : = \text{<integer>}
\]

\[
\text{<opcode>} : = \text{<symbol>} / \text{<constant>}
\]

The opcodes specification is one statement. As many continuation cards as desired may be used. Cards may break the statement at three places: immediately following a right parenthesis completing an opcode field specification; after a comma between opcodes for one field; or after a hyphen within a numeric constant. In the latter case, the numeric constant is hyphenated on one card and the rest of the constant continues (somewhere beyond column one) on the following card. This is a useful facility mainly for long binary constants.

The programmer should define one opcode field for each opcode type in the format statements. He may list as many opcodes for one field as he wishes. Figure 13 illustrates an opcode statement for the CMD, TEST and JUMP formats of Figure 12.

Format id and field number together point to a specific field in a format statement. Format id identifies the format statement. Fields in that statement are numbered, starting with 1, in the order they are defined. It is this order of definition that is important, not the left-to-right order in the word. Thus, field JUMP1 of Figure 12 is the twelve-bit address field beginning in bit six.

There are three kinds of errors in the opcodes statement. First, any field pointed to must be an opcode type. Second, the value associated with a symbol must fit into the designated field. And third, for one format, no opcode symbol may be repeated.

The reason for this last restriction will become clear in the next section. Briefly, for a given format, a particular opcode field is singled out in a program statement by the appearance of one of the opcode symbols defined for it. Thus, all of the opcode fields of that format must share the same set of symbols; if one symbol is repeated, it is ambiguous in identifying a field. Of course, from format to format symbols may be repeated.
PROGRAM STATEMENTS

Statements represent the encoding of the customer's control logic. All logic, formats, and mapping he has designed previously are all used to create this symbolic program to a binary representation for ROM. The logic established, is now used:

\[
\text{program statements} \rightarrow \text{custom statement} \mid \text{predefined statement}
\]

\[
\text{custom statement} \rightarrow [\text{format id}] \cdot \{\text{label symbol}\} \cdot
\]

\[
\cdot \text{separator} \cdot \langle \text{program field} \rangle \cdot \{\text{separator} \cdot \langle \text{program field} \rangle \}
\]

\[
\text{label symbol} \rightarrow \text{/symbol}
\]

\[
\text{separator} \rightarrow \text{,} \mid \text{blank}
\]

\[
\text{program field} \rightarrow \text{opcode symbol} \mid \text{address expression}
\]

\[
\text{opcode symbol} \rightarrow \text{symbol}
\]

\[
\text{address expression} \rightarrow \text{field number} \cdot \{\text{expression}\}
\]

\[
\text{expression} \rightarrow \{\text{operator}\} \cdot \langle \text{term} \rangle \cdot \{\text{operator} \cdot \langle \text{term} \rangle \}
\]

\[
\text{operator} \rightarrow + \mid -
\]

\[
\text{term} \rightarrow \text{label symbol} \mid
\]

\[
\cdot \text{constant} \mid \cdot *
\]

\[
\text{predefined statement} \rightarrow [\text{*}] \cdot \{\text{predefined type}\}
\]

\[
\text{predefined type} \rightarrow \text{align statement} \mid \text{constant statement} \mid \text{equate statement}
\]
The program section begins with the $PROGRAM statement and ends with the $END statement. Each statement between these two is a program statement. Theoretically, there is no limit to the number of statements to a program. Program statements fall into two major classes: those whose symbology the customer has defined, the predefined statements. We will consider these two classes separately.
MICROPROCESSOR BOOKS

THE FOLLOWING BOOKS MAY BE OBTAINED FROM SYBEX:

A 1 - MICROCOMPUTERS:
FROM CHIPS TO SYSTEMS

A 2 - MICROPROCESSOR PROGRAMMING

B 3 - SEVERE - ENVIRONMENT MICROPROCESSOR SYSTEMS:
MILITARY, AVIONICS, AEROSPACE, MARINE, INDUSTRIAL

B 5 - BIT - SLICE:
TECHNOLOGY AND APPLICATIONS

CONTACT US FOR SEMINARS AND SPECIAL IN - HOUSE COURSES.

USA:
SYBEX INCORPORATED
510 GRIZZLY PEAK BLVD.
BERKELEY, CALIFORNIA 94708
TEL: (415) 526 - 2748

EUROPE:
SYBEX - EUROPE
313 RUE LECOURBE
75015 - PARIS, FRANCE
TEL: (1) 828 - 2502

SYBEX
ACKNOWLEDGEMENTS

ACKNOWLEDGEMENTS ARE GRATEFULLY EXTENDED TO THE FOLLOWING COMPANIES OR ORGANIZATIONS WHO HAVE PROVIDED TECHNICAL INFORMATION, PERSONAL INTERVIEWS, OR AUTHORIZED THE REPRODUCTION OF COPYRIGHTED MATERIAL.

ACTRON
ADVANCED MICRO DEVICES
AUTONETICS
AMERICAN MICROSYSTEMS
CMC
DATA GENERAL
DCA LABS
DEC
E & L
ELECTRONIC ARRAYS
EMM
FAIRCHILD SEMICONDUCTOR
FLUKE
GENERAL AUTOMATION
GENERAL INSTRUMENTS
HARRIS SEMICONDUCTOR
HEWLETT - PACKARD

HUGHES
IBM
IMSAI
INTEL CORPORATION
INTERSIL
LOCKHEED
MARTIN RESEARCH
MICROCOMPUTER ASSOCIATES
MONOLITHIC MEMORIES
MOS TECHNOLOGY
MOSTEK CORPORATION
MOTOROLA SEMICONDUCTOR
MYCRO - TEK
NATIONAL SEMICONDUCTOR
NEC
NELC
OSBORNE AND ASSOCIATES

PHILIPS
PLESSEY
PROLOG
R 2 E
RAYTHEON
RCA SOLID STATE
ROCKWELL INTERNATIONAL
SCIENTIFIC MICROSYSTEMS
SESCOSEM
SIEMENS
SIGNETICS
TEKTRONIX
TEXAS INSTRUMENTS
TRANSITRON ELECTRONICS
TRENDAR
WESTERN DIGITAL CORPORTATION
ZILOG
MICROPROCESSOR MANUFACTURERS (I)

AMD (ADVANCED MICRODEVICES)
901 THOMPSON PLACE, SUNNYVALE, CALIFORNIA 94806
(408) 732-2400 TELEX: 346306

AMI (AMERICAN MICROSYSTEMS)
3800 HOMESTEAD ROAD, SANTA CLARA, CALIFORNIA 95051
(408) 246-0330

DATA GENERAL
SOUTHBORO, MASSACHUSETTS 01772
(617) 485-9100 TELEX: 48460

FAIRCHILD SEMICONDUCTOR
1725 TECHNOLOGY DRIVE, SAN JOSE, CALIFORNIA 95110
(408) 998-0123
MICROPROCESSOR MANUFACTURERS (II)

GI (GENERAL INSTRUMENTS)
600 WEST JOHN STREET, HICKSVILLE, NEW YORK 16002
(516) 733 - 3107  TWX : (510) 221 - 1666

INTEL
3065 BOWERS AVENUE, SANTA CLARA, CALIFORNIA 95051
(408) 246 - 7501  TELEX : 346372

INTERSIL
10090 NORTH TANTAU AVENUE, CUPERTINO, CALIFORNIA 95014
(408) 996 - 5000  TWX : (916) 338 - 0228

MMI (MONOLITHIC MEMORIES)
1165 EAST ARQUES AVENUE, SUNNYVALE, CALIFORNIA 94086
(408) 739 - 3535
MICROPROCESSOR MANUFACTURERS (III)

MOSTEK
1215 WEST CROSBY ROAD, CARROLLTON, TEXAS 75006
(214) 242 - 0444 TELEX : 30423

MOTOROLA SEMICONDUCTOR
BOX 20912, PHOENIX, ARIZONA 85036
(602) 244 - 6900 TELEX : 67325

NS (NATIONAL SEMICONDUCTOR)
2900 SEMICONDUCTOR DRIVE, SANTA CLARA, CALIFORNIA 95051
(408) 732 - 5000 TWX : (910) 339 - 9240

RCA SOLID STATE
BOX 3200, SOMERVILLE, NEW JERSEY 08876
(201) 722 - 3200 TWX : (718) 480 - 9333

ROCKWELL INTERNATIONAL
BOX 3669, ANAHEIM, CALIFORNIA 92803
(714) 632 - 3698
SIGNETICS
811 EAST ARQUES AVENUE, SUNNYVALE, CALIFORNIA 94086
(408) 739 - 7700

TI (TEXAS INSTRUMENTS) - DIGITAL SYSTEMS DIVISION
P.O. BOX 1444, HOUSTON, TEXAS 77001
(713) 494 - 5115

WESTERN DIGITAL
3128 REDHILL AVENUE, NEWPORT BEACH, CALIFORNIA 92663
(714) 557 - 3550 TWX: (910) 595 - 1139

ZILOG
170 STATE STREET, LOS ALTOS, CALIFORNIA 94022
(415) 526 - 2748 TWX: (910) 370 - 7955
<table>
<thead>
<tr>
<th>Code</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>B 1</td>
<td>MICROPROCESSORS: FROM CHIPS TO SYSTEMS</td>
</tr>
<tr>
<td>B 2</td>
<td>MICROCOMPUTER PROGRAMMING</td>
</tr>
<tr>
<td>B 3</td>
<td>MILITARY MICROPROCESSOR SYSTEMS</td>
</tr>
<tr>
<td>B 5</td>
<td>BIT SLICE</td>
</tr>
<tr>
<td>B 6</td>
<td>INDUSTRIAL MICROPROCESSOR SYSTEMS</td>
</tr>
<tr>
<td>B 7</td>
<td>MICROPROCESSOR INTERFACING TECHNIQUES</td>
</tr>
<tr>
<td>C 4</td>
<td>LES MICROPROCESSEURS (IN FRENCH)</td>
</tr>
<tr>
<td>C 10</td>
<td>INTRODUCTION AUX MICROPROCESSEURS (IN FRENCH)</td>
</tr>
<tr>
<td>E 5</td>
<td>MICROPROCESSOR ENCYCLOPEDIA: BIT SLICES</td>
</tr>
<tr>
<td>E 8</td>
<td>MICROPROCESSOR ENCYCLOPEDIA: 8-BITS</td>
</tr>
<tr>
<td>F 1</td>
<td>THE 30 MICROPROCESSOR FAMILIES - THE CHIPS</td>
</tr>
<tr>
<td>F 2</td>
<td>THE 30 MICROPROCESSOR FAMILIES - INSTRUCTION SETS</td>
</tr>
<tr>
<td>M 7</td>
<td>INTERNATIONAL MICROPROCESSOR DICTIONARY</td>
</tr>
<tr>
<td>M 11</td>
<td>AN INTRODUCTION TO MICROCOMPUTERS: BASIC CONCEPT</td>
</tr>
<tr>
<td>M 12</td>
<td>AN INTRODUCTION TO MICROCOMPUTERS: SOME REAL PROD</td>
</tr>
<tr>
<td>M 13</td>
<td>8080 PROGRAMMING FOR LOGIC DESIGN</td>
</tr>
<tr>
<td>M 14</td>
<td>6800 PROGRAMMING FOR LOGIC DESIGN</td>
</tr>
<tr>
<td>ACR</td>
<td>ACRONYMS</td>
</tr>
<tr>
<td>L 1</td>
<td>MICROPROCESSOR DESIGN LABORATORY</td>
</tr>
</tbody>
</table>

SYBEX
SYBEX seminars are presented throughout the world in major cities at regular intervals. They can also be taught at your facility for a minimum number of participants (10 to 15) at substantial savings.

Partial list of courses:

A 1 - Microprocessors
A 2 - Programming Microprocessors
B 3 - Military Systems
B 4 - Microprocessors for Microwave Engineers
B 5 - Bit-Slice
B 6 - Industrial Applications
B 7 - Interfacing
M 12 - Comparative Evaluation
L 1 - Laboratory Course
MOS 1 - MOS fabrication
MOS 2 - MOS circuit design
MOS 3 - N-channel process control
P 20 - Microprogrammed APL

Contact the nearest SYBEX office for full details.