

OPERATOR'S MANUAL

MODEL 2366

UNIVERSAL PROGRAMMABLE
LOGIC MODULE

Preliminary
March, 1994

(ECO 1000)

The CAMAC 2366 is a general purpose programmable logic module, using state of the art field programmable gate array technology. This CAMAC module can be used as a programmable LeCroy Ecline trigger processor module, among other uses. In addition to a full 24 bit CAMAC interface, there are 59 front panel differential ECL i/o signals, which (with some restrictions) can be independently selected to be either inputs or outputs. This module is also useful as a general purpose controller, as part of a test system or data acquisition system.

The desired logical operations are programmed in a Xilinx 4005 gate array chip. Any logic that can be implemented as a synchronous (clocked) state machine may be programmed, subject only to the limitations of the Xilinx 4005 gate array chip (approximately 5000 equivalent gates). There are 3 clocks available on the board, 40, 20, and 10 MHz, or any of 3 special front panel inputs may be used as a clock. Input and output signals use standard 10124 and 10125 TTL-ECL level translators. Input signals as short as 5 nanoseconds can be latched and synchronized with the internal state machine logic.

The gate array must be programmed after power up, and can be reprogrammed at any time. An onboard EPROM socket may contain a program which is loaded on power up, or reloaded on CAMAC command. The Xilinx chip may also be programmed directly from the CAMAC dataway (the program information is stored in RAM in the Xilinx chip, so there is no limit to the number of times that it can be reprogrammed).

A few basic CAMAC Function codes are implemented in hardware, and are available on power up. These are only used to program the Xilinx chip, and all but one of these functions dissappear after a CAMAC clear operation (F9, C, or Z).

F30	A0-A15	enter programming mode with eprom selected, enable all other hardware function codes
F28	A0-A15	select CAMAC programming mode
F25	A0-A15	program Xilinx chip (program pulse lasts until the next S1)
F16	A0-A15	write 8 bits to the Xilinx
F12	A0-A15	test Xilinx READY line (not usually required)
F13	A0-A15	test Xilinx program DONE
F14	A0-A15	test Xilinx INIT line
F9	A0-A15	disable function codes except for F30 (CAMAC C, Z have the same effect)

The Xilinx chip automatically loads itself if an EPROM is installed on the board. An F30 command followed by an F25 will cause the Xilinx chip to be reset and reloaded from the EPROM.

To load a program from CAMAC, the F30 command is followed by an F28 and F25. After the Xilinx INIT line is true (test with F14), the data is written 8 bits at a time using F16. F12 tests the Xilinx ready line before each write operation. This is not required unless the CAMAC host is capable of CAMAC operations at a rate greater than 500 kHz. This continues until all data is written and the Xilinx DONE line is true (test with F13). The Xilinx XACT software or any of several third party gate array software packages can be used to prepare the program file. An example basic program is supplied which reads the Xilinx .BIT file and uses standard ESONE CAMAC functions to load the program file into the 2366 module.

The initial program in the EPROM is T2366E (the schematic is included in the manual) This implements a simple divide chain to flash the leds with the 3 internal clocks (40mhz, 20mhz and 10 mhz), a 24 bit read write register and a register which latches the CAMAC F,A,,Z,I,N on every S2. This last register is read by F0, A1. This emulates most of the test portion of the LeCroy 2050 CAMAC dataway display and test module.

A simple basic program which exercises the 2366 is EP2366.bas. It starts by forcing the reload of the Xilinx chip from the eprom (with T2366E), and implements a simple CAMAC system test.

The example program LOAD2366.bas will load an arbitrary xxxx.bit file into the Xilinx chip.

SPECIFICATIONS: 2366 Universal Programmable Logic Module
single width CAMAC module

1 Led indicates N line activity

2 programmable Leds

59 i/o signals on front panel, all are differential ECL

all front panel i/o is to and from the Xilinx chip

input or output is selectable in groups of 4

inputs are terminated with 112 ohms

outputs will drive 100 ohm lines

Programmable gate array: Xilinx 4005-5

196 configurable logic blocks and 112 i/o blocks

fast carry logic, wide decoding

approximately 5000 gates: 616 flipflops, 6272 Ram bits

The 4005-5 can be replaced with a 4005-4 or 4005-3, by the user, as they become available from Xilinx. The socket is also pin compatible with the Xilinx 4006.

40 MHz, 20 MHz, and 10 MHz crystal clocks on the board

programmable by optional on board (socketed) Eprom on power up, or by CAMAC command to reload from Eprom

programmable via CAMAC, at any time, independent of Eprom.

Xilinx XACT software system is required for programming

Uses Xilinx .BIT file for programming information

Basic CAMAC programming software for IBM compatible included

11875 CAMAC F16 write operations are required to program

all logic must be clocked (synchronous logic)

Camac interface:

programming, 8 bit write only interface

Test for successful programming

after programming, all camac control and data lines (N, F, A, 24R/W, C, Z, S1, S2, Q, X, L) are available to Xilinx chip

Only 1 function code is reserved for reprogramming, all others are available for the user

Possible Applications:

trigger logic, digital 48 input majority logic

trigger or readout controller, pipelined sequential logic

pulse sequence generator, any arbitrary state machine logic

fast memory, FIFO or LIFO, 16 bit digital adder

FRONT PANEL

programmable LEDS, top = D14 (red), bottom = C16 (yellow)

Front Panel Input-Output pins:

PAIR NUMBER	A (8)	B (17)	C (17)	D (17)	connector (pairs)
1	F15	R6	B2	J15	
2	E16	T4	C9	J16	
3	F16	R4	B9	K14	
4	G14	R3	A9	K15	
5	P9	N2	C10	K16	
6	T6	M3	B10	L15	
7	R7	L2	A10	L16	
8	T5	L1	B11	M14	
9		K1	A11	M16	
10		J1	C12	N14	
11		G2	B12	N15	
12		G3	B13	P12	
13		E1	A13	P15	
14		E2	A14	P16	
15		C2	C15	R13	
16		D3	D15	T13	
17		R16	B14	T2	

The input (and output) polarity is such that a positive ECL edge (the odd numbered pin on the input connector is a positive going edge) produces a negative TTL edge at the Xilinx input. This gives the best possible performance for capturing short input signals (as short as 5 nsec). The input signal should be inverted inside the Xilinx chip and used as the clock to a D Flip Flop (with a logic 1 as the D input). This does mean however that a normal ECL logic 1 (odd pin +) becomes a logic 0 inside the Xilinx chip. Input and output are consistent, both are inverted. We suggest as a general rule to invert the data inside the Xilinx chip as it enters and leaves. This allows normal positive logic to be used inside the Xilinx chip. These extra inverters are not really extra, they are simply absorbed inside the logic and do not usually cause any extra propagation delay.

All front panel i/o is selectable as input or output in groups of 4, as indicated below (the 3 clocks inputs are separately selectable). This is accomplished by installing the socketed ECL-TTL or TTL-ECL level converters and the appropriate termination resistor networks. All inputs and outputs are differential ECL. The corresponding Xilinx pins must be programmed as either input or output.

For input only the 10125 ECL to TTL converters and the 56 ohm termination resistor SIPs are installed. The inputs are properly terminated for twisted pair cable.

For output, only the 10124 TTL to ECL converters and the 390 ohm pull down resistor SIP are installed.

Note that the module will work properly with short cables (but with reduced noise immunity) even if both the input termination SIPs and the output pulldown SIPs are installed.

The logical polarity of any signal is programmable in the Xilinx chip of course. There are a total of 59 i/o signals.

1 16 pin (8 pairs) header:

i/o A 1-4
i/o A 5-8

3 34 pin (17 pairs) headers:

i/o B 1-4
i/o B 5-8
i/o B 9-12
i/o B 13-16
i/o B 17 (can be connected to SGCK1)

i/o C 1-4
i/o C 5-8
i/o C 9-12
i/o C 13-16
i/o C 17 (can be connected to SGCK2)

i/o D 1-4
i/o D 5-8
i/o D 9-12
i/o D 13-16
i/o D 17 (can be connected to SGCK3)

SGCK1, 2 and 3 are Xilinx internal clock distribution networks

It is possible to convert the front panel i/o to bidirectional TTL. This user modification cannot be wholeheartedly recommended, however, since little protection is provided for the Xilinx chip. This method will only work at low speeds (less than 1 MHz) and for very short cable lengths. Deglitching will be needed at all receivers. The Xilinx pins cannot drive 100 ohm terminated lines. For best results we recommend using the terminated ECL for the cable runs, and converting to TTL at the destination circuit board. This is necessary if high speed performance is required.

To provide TTL connections construct this 16 pin dip header with 4 100 ohm resistors, for each group of 4 to be used as TTL.

Install this header in the 10124 location after removing both resistor packs, and both the 10124 and 10125 chips.

This user modification connects the Xilinx pin directly to the odd numbered pin on the front panel connector. The corresponding even numbered pin is grounded. There is no buffer, ONLY a series resistor to prevent damage to the Xilinx chip.

The Xilinx chip can be programmed for either INPUT, OUTPUT or as a TRISTATE pin.

Please Use With Care!

pin

1	connect to 16
2	connect to 16
3	100 ohm resistor to 7
4	100 ohm resistor to 5
5	100 ohm resistor to 4
6	
7	100 ohm resistor to 3
8	
9	
10	100 ohm resistor to 12
11	100 ohm resistor to 13
12	100 ohm resistor to 10
13	100 ohm resistor to 11
14	connect to 16
15	connect to 16
16	connect to 1,2,14,15 (ground)

note that only 4 100 ohm resistors are required.

A QUICKBASIC program to load a Xilinx.bit file
into the 2366 ULM module, using Esone standard
CAMAC subroutines

2pages

```
DECLARE FUNCTION init% (file$, var$, value!)
```

```
PRINT
fff$ = "load2366.ini"
jj$ = "camslot"
isok% = init%(fff$, jj$, value)
IF isok% = 0 THEN
  INPUT "Where is the 2366? Camac slot #"; value
END IF
modl% = value
```

```
ON ERROR GOTO fault
IF COMMAND$ = "" THEN
INPUT " Please enter the file name"; fil$
ELSE
  fil$ = COMMAND$
END IF
```

```
start:
```

```
crate% = 1
CALLS cdreg(tmod0%, 0, crate%, modl%, 0)
CALLS cdreg(tmod1%, 0, crate%, modl%, 1)
CALLS cdreg(mtslot%, 0, crate%, 2, 0)

CALLS ccinit(0)
CALLS cssa(9, tmod0%, ddt%, qq%)
```

```
PRINT : PRINT " RELOAD VIA CAMAC"
OPEN fil$ FOR BINARY ACCESS READ AS #1
jk$ = " " '1 byte long
```

```
CALLS ccinit(0)
'enter programming mode
CALLS cssa(30, tmod0%, ddt%, qq%)
'select camac mode
CALLS cssa(28, tmod0%, dd, qq%)
'program pulse
CALLS cssa(25, tmod0%, ddt%, qq%)
'wait for init line
ttt% = 0: PRINT " INIT ";
DO
CALLS cssa(14, tmod0%, ddt%, qq%)
PRINT qq%;
ttt% = ttt% + 1
IF ttt% > 10000 THEN EXIT DO
LOOP WHILE qq% <> 1
PRINT
'STOP
'skip the first 16 bytes in the file
FOR i% = 0 TO 15
  GET #1, , jk$
  'IF ASC(jk$) = 0 THEN GOTO fault
NEXT
```

```
np% = 0
```

```
GET #1, , jk$
```

```
DO
```

```
num% = ASC(jk$)
IF num% = 255 THEN EXIT DO
IF (num% > 31) AND (num% < 123) THEN
  PRINT jk$;
  np% = 0
ELSE
  IF np% = 0 THEN PRINT " ";
  np% = 1
END IF
LOOP
GOTO startup
```

```
FOR i% = 1 TO 4
  zc = 0
  DO
    GET #1, , jk$
    zc = zc + 1
  IF zc > 1000 THEN GOTO fault
  LOOP WHILE ASC(jk$) < 32
  PRINT jk$;
  DO
    GET #1, , jk$: num% = ASC(jk$)
    IF num% > 31 THEN PRINT jk$;
    LOOP WHILE num% < 32
    PRINT " loop1 ";
  NEXT
  PRINT
  zc = 0
  DO
    GET #1, , jk$
    zc = zc + 1
  IF zc > 1000 THEN GOTO fault
  LOOP WHILE ASC(jk$) < 32
  PRINT "loop2"
```

```
GET #1, , jk$: num% = ASC(jk$)
IF num% <> 255 THEN GOTO fault
```

```
startup:
```

```
GOSUB sendit
GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
PRINT num%;
num% = num% AND 15: lengt% = num%
GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
PRINT num%;
lengt% = lengt% * 256 + num%
GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
PRINT num%;
lengt% = lengt% * 256 + num%
GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
PRINT num%;
num% = (num% AND 240) / 16
lengt% = lengt% * 16 + num%
nbytes = lengt% / 8
hide% = INT(nbytes)
PRINT : PRINT lengt%; "(bits) "; nbytes; hide%; "bytes"

FOR j% = 1 TO hide%
  GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
```

```

CALLS cssa(13, tmod0%, dd%, qq%)
IF qq% = 1 THEN PRINT "done at j%="; j%; EXIT FOR
NEXT
CLOSE 1

PRINT "finished ";
'num% = 128: OUT dp%, num%: OUT cp%, dn%: OUT cp%, up%

CALLS cssa(13, tmod0%, dd%, qq%)
IF qq% = 1 THEN
  PRINT "!!!success!!!"
ELSE
  PRINT "****FAILURE****"
END IF
PRINT

CALLS cssa(9, tmod0%, dd%, qq%)
errnum% = 0
SYSTEM

sendit:
CALLS cssa(16, tmod0%, num%, qq%)
'PRINT num%; flip; " ";
RETURN

fault:
PRINT "error"
SYSTEM
END

FUNCTION init% (file$, var$, value)
' read in an initialization file of the form:
' varname=value
' varname=value
' etc
' case is ignored
' one variable per line
' variables can be in any order
' init% returns as 0 for variables not found
'
  init% = 0
  ON LOCAL ERROR GOTO lerr
  OPEN file$ FOR INPUT AS #1
  WHILE NOT EOF(1)
    LINE INPUT #1, line$
    line$ = UCASE$(line$)
    IF (INSTR(line$, UCASE$(var$)) <> 0) AND (INSTR(line$, "=") <> 0) THEN
      jnk$ = RIGHT$(line$, LEN(line$) - INSTR(line$, "="))
      value = VAL(jnk$): init% = 1
    END IF
  WEND

lerr:
  CLOSE #1
  EXIT FUNCTION
END FUNCTION

```

A QUICKBASIC program to load a Xilinx.bit file into the 2366 ULM module, using proprietary LeCroy CAMAC subroutines and a LeCroy 8901 GPIB CAMAC crate controller, and a National Instruments PCII interface card in an IBM PC. This is also supplied as an EXE file on the disk.

2 pages

```
DECLARE FUNCTION init$(file$, var$, value!)
```

```
restart:
```

```
PRINT
'fff$ = "load2366.ini"
fff$ = "j"

jj$ = "camslot%"
isok% = init$(fff$, jj$, value)
IF isok% = 0 THEN
  INPUT "Where is the 2366 to load? Camac slot #"; value
END IF
modl% = value

'ON ERROR GOTO fault
IF COMMAND$ = "" THEN
  INPUT "Please enter the XXX.BIT file name to load"; fil$
ELSE
  fil$ = COMMAND$
END IF
```

```
start:
```

```
CALLS gpinit
CALLS gpsic
CALLS gpron
crate% = 1
CALLS gpcini(crate%)
CALLS gpcccz(crate%)

' CALLS GPCFSA(crate%, f%, a%, n%, int4&, xq%)
x=1, q=2, so x&q=3
```

```
ulm% = modl%
CALLS GPCFSA(crate%, 9, 0, modl%, dd&, qq%)
GOSUB reload
CALLS GPCFSA(crate%, 9, 0, modl%, dd&, qq%)
```

```
INPUT "Load another? (Y/N)"; yn$
IF UCASE$(yn$) = "Y" THEN GOTO restart
```

```
SYSTEM
```

```
reload:
```

```
PRINT " RELOAD VIA CAMAC ";
OPEN fil$ FOR BINARY ACCESS READ AS #1
jk$ = " " '1 byte long
```

```
rel2:
```

```
'enter programming mode
'CALLS cssa(30, ulm%, dd&, qq%)
CALLS GPCFSA(crate%, 30, 0, ulm%, dd&, qq%)
  WHILE INKEY$ = "": WEND
'select camac mode
'CALLS cssa(28, ulm%, dd, qq%)
CALLS GPCFSA(crate%, 28, 0, ulm%, dd&, qq%)
'program pulse
'CALLS cssa(25, ulm%, dd&, qq%)
CALLS GPCFSA(crate%, 25, 0, ulm%, dd&, qq%)
'wait for init line
```

```
ttt% = 0: PRINT " INIT ";
CALLS GPCFSA(crate%, 14, 0, ulm%, dd&, qq%)
IF qq% = 3 THEN GOTO rel2
```

```
DO
'CALLS cssa(14, ulm%, dd&, qq%)
CALLS GPCFSA(crate%, 14, 0, ulm%, dd&, qq%)
dd% = dd&
PRINT qq%;
ttt% = ttt% + 1
IF ttt% > 100 THEN GOTO rel2
LOOP WHILE qq% <> 3
PRINT : PRINT " ";
```

```
'skip the first 16 bytes in the file
FOR i% = 0 TO 15
  GET #1, , jk$
  'IF ASC(jk$) = 0 THEN GOTO fault
NEXT
```

```
np% = 0
```

```
DO
  GET #1, , jk$
  num% = ASC(jk$)
  IF num% = 255 THEN EXIT DO
  IF (num% > 31) AND (num% < 123) THEN
    PRINT jk$;
    np% = 0
  ELSE
    IF np% = 0 THEN PRINT " ";
    np% = 1
  END IF
```

```
LOOP
```

```
start with the FF word.....
```

```
GOSUB sendit
GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
PRINT num%;
num% = num% AND 15: lengt% = num%
GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
PRINT num%;
lengt% = lengt% * 256 + num%
GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
PRINT num%;
lengt% = lengt% * 256 + num%
GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
PRINT num%;
num% = (num% AND 240) / 16
lengt% = lengt% * 16 + num%
nbytes = lengt% / 8
hide% = INT(nbytes)
PRINT : PRINT lengt%; "(bits) "; nbytes; hide%; "bytes"

FOR j% = 1 TO hide%
  GET #1, , jk$: num% = ASC(jk$): GOSUB sendit
  'CALLS cssa(13, ulm%, dd&, qq%)
  CALLS GPCFSA(crate%, 13, 0, ulm%, dd&, qq%)
  IF qq% = 3 THEN PRINT "done at j%="; j%; " "; : EXIT FOR
NEXT
CLOSE 1
PRINT "finished ";
```

```
'CALLS cssa(13, ulm%, dd%, qq%)
CALLS GPCFSA(crate%, 13, 0, ulm%, dd%, qq%)
IF qq% = 3 THEN
  PRINT "!!!success!!!"
ELSE
  PRINT "****FAILURE****"
END IF

'CALLS cssa(9, ulm%, dd%, qq%)
CALLS GPCFSA(crate%, 9, 0, ulm%, dd%, qq%)
CALLS GPCFSA(crate%, 30, 0, ulm%, dd%, qq%)
CALLS GPCFSA(crate%, 9, 0, ulm%, dd%, qq%)

RETURN
sendit:
'CALLS cssa(16, ulm%, num%, qq%)
num& = num%
CALLS GPCFSA(crate%, 16, 0, ulm%, num&, qq%)
'PRINT num%; flip%; " ";

RETURN
```

```
END

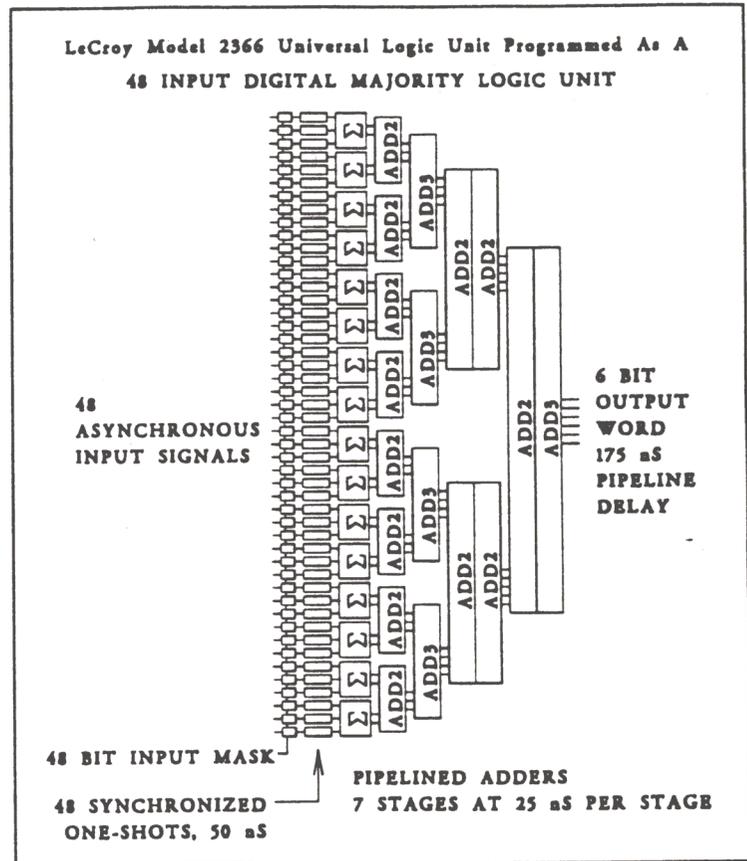
FUNCTION init% (file$, var$, value)
' read in an initialization file of the form:
' varname=value
' varname=value
' etc
' case is ignored
' one variable per line
' variables can be in any order
' init% returns as 0 for variables not found

init% = 0
ON LOCAL ERROR GOTO lerr
OPEN file$ FOR INPUT AS #1
WHILE NOT EOF(1)
  LINE INPUT #1, line$
  line$ = UCASE$(line$)
  IF (INSTR(line$, UCASE$(var$)) <> 0) AND (INSTR(line$, "=") <> 0) THEN
    jnk$ = RIGHT$(line$, LEN(line$) - INSTR(line$, "="))
    value = VAL(jnk$): init% = 1
  END IF
WEND

lerr:
CLOSE #1
EXIT FUNCTION
END FUNCTION
```

AN-52

USING THE MODEL 2366 UNIVERSAL LOGIC MODULE AS A 48-INPUT DIGITAL MAJORITY LOGIC UNIT



The LeCroy CAMAC Model 2366 Universal Logic Module is a general purpose logic module using a Xilinx 4005 field programmable gate array. The module has 59 front panel I/O signals, which can be configured as either input or output in 7 groups of 4 channels, and 3 individual channels. The Xilinx gate array can be programmed with an on-board EPROM, or from CAMAC.

For use as the 48-input majority logic module, the 2366 is configured as 48 inputs and 8 outputs. Three of the 59 possible I/O are not used unless needed for the clock (as discussed below). The 48 inputs are synchronized with the internal 40 MHz clock and trigger a 50 nsec one-shot.

The inputs are edge sensitive, triggering on the leading edge of pulses as short as 5 nsec. Seven pipelined adder stages sum the 48 input signals to produce the binary output word. The output word is updated at the 40 MHz clock rate, with a 175 nsec pipeline delay. A 48-bit mask register (CAMAC F16, A0 and A1, read with F0) is used to enable (=1) or disable (=0) the individual inputs.

Since the inputs have been synchronized with the clock, the coincidence resolution is quantized. The coincidence resolution for determining the accidental triggers is 37.5 nsec. Real events must have all inputs within a 25 nsec span to trigger with 100% efficiency.

This program uses most of the resources of the Xilinx 4005 in the 2366 module, 243 function generators (62%) and 352 CLB flip-flops (90%). In order to operate at 40 MHz, adders with more than 3 input bits are split into 2 pipeline stages. Several variations of this circuit are possible. By slowing the clock rate to 20 MHz, the circuit can be simplified to four pipeline stages. The coincidence resolution would be twice that above, 75 nsec for accidentals, and 50 nsec for real events. The pipeline delay would be 200 nsec, with a 20 MHz output word rate.

If the input signals are wide pulses, a different synchronizing circuit is possible. The input can simply be clocked with the 40 MHz clock using the I/O flip-flop, then differentiated to produce a short pulse. This method frees up two extra flip-flops per input, which can be used to lengthen the input pulse from 50 to 75 nsec. This changes the coincidence resolution to 62.5 nsec for accidentals and 50 nsec for real events.

The output word is a 6-bit binary number which is the number of inputs that were true during a 25 nsec period 175 nsec earlier. If 48 inputs is sufficient, the 6-bit output word of the majority logic can be digitally compared with a CAMAC loaded register and produce a trigger signal directly.

For systems which require a larger majority logic unit, multiple 48-input majority logic units can be combined. The 6-bit output word can be time aligned (in 25 nsec steps) and added to the outputs of other majority logic modules, using another 2366 module. A single 2366 module configured as an adder tree can add eight 6-bit input words in 7 pipeline stages for a pipeline delay of 175 nsec. The output word is limited to 8 bits in this example, so the sum will limit at 255 (there can be as many as 384 inputs). If counting to beyond 255 is required, the 2366 can be configured to have fewer inputs and a wider output word (in units of 4 bits).

Extending this to one more level (adding another 2366 to sum the outputs of the adders) allows six 8-bit inputs to be counted in 9 pipeline stages (again limiting at 255). This produces a 2304 input majority logic in a total of 25 pipeline stages (allowing one stage for going from module to module), or 625 nsec. When using multiple 2366 modules, a common clock should be used to synchronize the modules. Each 2366 has 3 inputs which can be used as the clock source.

This majority logic program consumes most of the resources of the Xilinx chip. There is very little left with which to implement test features. A trigger system for a large high energy physics experiment

is usually very complex, with many modules and very, very many cables and connectors which combine the logic modules to produce the desired trigger algorithm. Testing a system like this usually requires looking at signals with oscilloscopes and logic analyzers. It also usually requires disconnecting cables. It is usually not possible to test the system completely in situ. This allows many possibilities for errors, and for undetected (until long after the run) problems such as intermittent or missing connections.

The 2366 universal logic module solves this problem by allowing COMPLETE reprogrammability of the trigger logic. This is dynamic reprogramming, on the fly, without moving a single cable or jumper anywhere in the system. Test logic can be downloaded over CAMAC in a second or two per module, completely replacing the trigger logic. This test logic can be as simple as read write registers which drive the cable connections (testing both the integrity of the connections AND verifying the topology) to complex pattern generators which test the trigger algorithms of other modules. When testing is finished (which could require several different downloads of test logic) the trigger logic is simply reprogrammed with the trigger algorithms and the experiment is ready to run, with a much higher confidence level than normally found in trigger systems.

This reprogrammability also allows the trigger to be easily and quickly changed for new requirements. If the topology is the same, or a superset of the previous topology, then switching back and forth between the old and new triggers can be done quickly AND safely, without touching a cable.

SPECIFICATIONS

- 48 Inputs, Differential ECL, Pulse Width Greater Than 5 nsec
- Synchronized Internally With the 40 MHz Clock
- 24-Bit CAMAC Interface
- 48-Bit Input Mask Register to Enable/Disable Individual Inputs
- Output: 6-Bit Digital Word Equal to the Number of Inputs True Within the Coincidence Resolving Time
- 40 MHz Output Rate
- Output Pipeline Delay: 175 nsec
- Coincidence Resolution: Minimum 25 nsec, Maximum 50 nsec
- Complete Schematic and Xilinx Programming Files are Available on Request