Warp™

VHDL Development System

Warp System Library Manual

Cypress Semiconductor
3901 North First Street
San Jose, CA 95134
(408)943-2600
JANUARY 1995

Cypress Software License Agreement

1. LICENSE. Cypress Semiconductor Corporation ("Cypress") hereby grants you, as a Customer and Licensee, a single-user, non-exclusive license to use the enclosed Cypress software program ("Program") on a single CPU at any given point in time. Cypress authorizes you to make archival copies of the software for the sole purpose of backing up your software and protecting your investment from loss.

2. TERM AND TERMINATION. This agreement is effective from the date the diskettes are received until this agreement is terminated. The unauthorized reproduction or use of the Program and/or documentation will immediately terminate this Agreement without notice. Upon termination you are to destroy both the Program and the documentation.

3. COPYRIGHT AND PROPRIETARY RIGHTS. The Program and documentation are protected by both United States Copyright Law and International Treaty provisions. This means that you must treat the documentation and Program just like a book, with the exception of making archival copies for the sole purpose of protecting your investment from loss. The Program may be used by any number of people, and may be moved from one computer to another, so long as there is **No Possibility** of its being used by two people at the same time.

4. DISCLAIMER. THIS PROGRAM AND DOCUMENTATION ARE LICENSED "AS-IS," WITHOUT WARRANTY AS TO PERFORMANCE. CYPRESS EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTY OF MERCHANTABILITY OR

FITNESS OF THIS PROGRAM FOR A PARTICULAR PURPOSE.

5. LIMITED WARRANTY. The diskette on which this Program is recorded is guaranteed for 90 days from date of purchase. If a defect occurs within 90 days, contact the representative at the place of purchase to arrange for a replacement.

6. LIMITATION OF REMEDIES AND LIABILITY. IN NO EVENT SHALL CYPRESS BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM PROGRAM USE, EVEN IF CYPRESS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. CYPRESS'S EXCLUSIVE LIABILITY AND YOUR EXCLUSIVE REMEDY WILL BE IN THE REPLACEMENT OF ANY DEFECTIVE DISKETTE AS PROVIDED ABOVE. IN NO EVENT SHALL CYPRESS'S LIABILITY HEREUNDER EXCEED THE PURCHASE PRICE OF THE SOFTWARE.

7. ENTIRE AGREEMENT. This agreement constitutes the sole and complete Agreement between Cypress and the Customer for use of the Program and documentation. Changes to this Agreement may be made only by written mutual consent.

8. GOVERNING LAW. This Agreement shall be governed by the laws of the State of California. Should you have any question concerning this agreement, please contact:

   Cypress Semiconductor Corporation
   Attn: Legal Counsel
   3901 N. First Street
   San Jose, CA 95134-1599

   408-943-2600

Table of Contents

Chapter 5 - Math

Chapter 6 - Macrocells

Chapter 7 - Memory

Chapter 8 - Multiplexers

Chapter 9 - Registers

Chapter 10 - Shifters

Chapter 11 - TTL Parts

Chapter 1 Introduction

About This Document

This manual provides the information you are likely to need to know about each component in the *Warp*[1] system libraries.

For each component, the following information is given:

- a diagram of the component's symbol, as instantiated on a schematic;
- a listing of the VHDL entity declaration for the component. This information is useful for determining the order, direction, and type of each port when instantiating the component in a VHDL file;
- a description (usually tabular) of the functionality of the component.

The manual is made up of the following sections:

- Section 2 - Counters
- Section 3 - Gates
- Section 4 - I/O

1. Warp is a trademark of Cypress Semiconductor Corporation.

- Section 5 - Math (adders, subtractors, comparators, multipliers)
- Section 6 - MCparts (macrocells, etc.)
- Section 7 - Memory
- Section 8 - Multiplexers (and de-multiplexers)
- Section 9 - Registers
- Section 10 - Shifters
- Section 11 - TTL

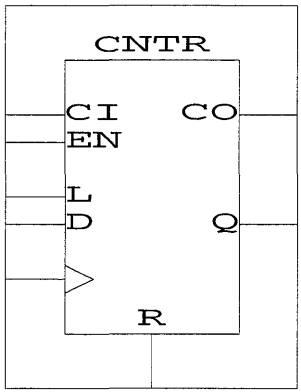
Chapter 2 Counters

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediately above the entity and architecture declarations

```
use work.counterpkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

2.1. CNTR

1-Bit Cascadable Up Counter.



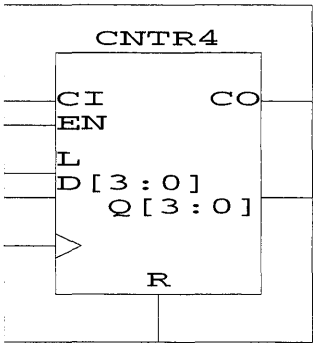

```
ENTITY cntr IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q: OUT BIT );
END cntr ;
```

DESCRIPTION

INPUTS						OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**CO**	**Q**
X	X	X	X	X	H	L	L
L	X	L	Λ	X	L	L	Q_0
H	L	L	Λ	X	L	Q_0	Q_0
H	H	L	Λ	X	L	$\overline{Q}_0$	$\overline{Q}_0$
X	X	H	Λ	L	L	L	L
X	X	H	Λ	H	L	H	H

2.2. CNTR4

4-Bit Cascadable Up Counter.



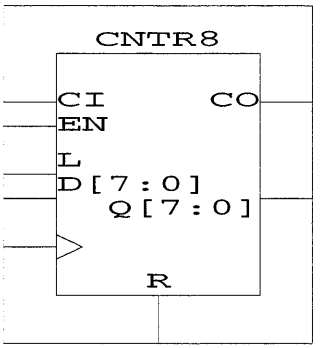

```
ENTITY cntr4 IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q3, q2, q1, q0: OUT BIT );
END cntr4 ;
```

DESCRIPTION

INPUTS						OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**CO**	**Q**
X	X	X	X	X	H	L	L
L	X	X	Λ	X	L	L	Q_0
H	L	L	Λ	X	L	H if Q=F, else L	Q_0
H	H	L	Λ	X	L	H if Q=F, else L	Q_0+1
X	X	H	Λ	L	L	L	L
X	X	H	Λ	H	L	H	H

2.3. CNTR8

8-Bit Cascadable Up Counter.



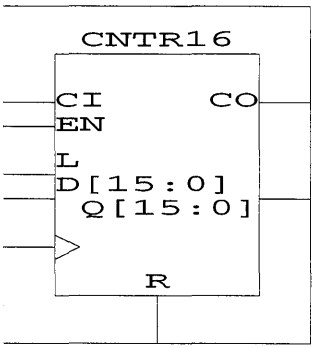

```
ENTITY cntr8 IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d7, d6, d5, d4, d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q7, q6, q5, q4, q3, q2, q1, q0: OUT
BIT );
END cntr8 ;
```

DESCRIPTION

INPUTS						OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**CO**	**Q**
X	X	X	X	X	H	L	L
L	X	X	Λ	X	L	L	Q_0
H	L	L	Λ	X	L	H if Q=FF, else L	Q_0
H	H	L	Λ	X	L	H if Q=FF, else L	Q_0+1
X	X	H	Λ	L	L	L	L
X	X	H	Λ	H	L	H	H

2.4. CNTR16

16-Bit Cascadable Up Counter.



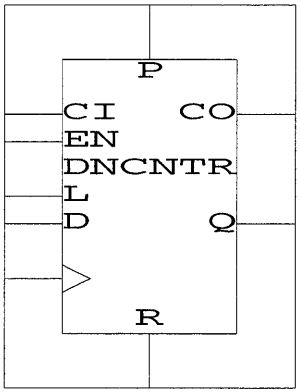

```
ENTITY cntr16 IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d15, d14, d13, d12, d11, d10, d9, d8,
         d7, d6, d5, d4, d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q15, q14, q13, q12, q11, q10, q9, q8,
         q7, q6, q5, q4, q3, q2, q1, q0: OUT
         BIT );
END cntr16 ;
```

DESCRIPTION

INPUTS						OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**CO**	**Q**
X	X	X	X	X	H	L	L
L	X	X	Λ	X	L	L	Q_0
H	L	L	Λ	X	L	H if Q=FFFF, else L	Q_0
H	H	L	Λ	X	L	H if Q=FFFF, else L	Q_0+1
X	X	H	Λ	L	L	L	L
X	X	H	Λ	H	L	H	H

2.5. DNCNTR

1-Bit Cascadable Down Counter.



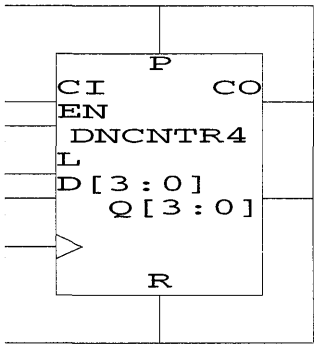

```
ENTITY dncntr IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         p: IN BIT ;
         q: OUT BIT );
END dncntr ;
```

DESCRIPTION

INPUTS							OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**P**	**CO**	**Q**
X	X	X	X	X	H	X	L	L
X	X	X	X	X	L	H	L	H
L	X	L	Λ	X	L	L	L	Q_0
H	L	L	Λ	X	L	L	$\overline{Q}_0$	Q_0
H	H	L	Λ	X	L	L	Q_0	$\overline{Q}_0$
X	X	H	Λ	L	L	L	H	L
X	X	H	Λ	H	L	L	L	H

2.6. DNCNTR4

4-Bit Cascadable Down Counter.



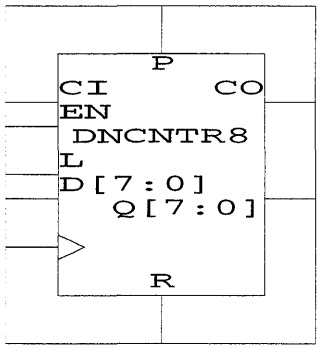

```
ENTITY dncntr4 IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         p: IN BIT ;
         q3, q2, q1, q0: OUT BIT );
END dncntr4 ;
```

DESCRIPTION

INPUTS							OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**P**	**CO**	**Q**
X	X	X	X	X	H	X	L	L
X	X	X	X	X	L	H	L	H
L	X	X	Λ	X	L	L	L	Q_0
H	L	L	Λ	X	L	L	H if Q=0, else L	Q_0
H	H	L	Λ	X	L	L	H if Q=0, else L	Q_0-1
X	X	H	Λ	L	L	L	H	L
X	X	H	Λ	H	L	L	L	H

2.7. DNCNTR8

8-Bit Cascadable Down Counter.



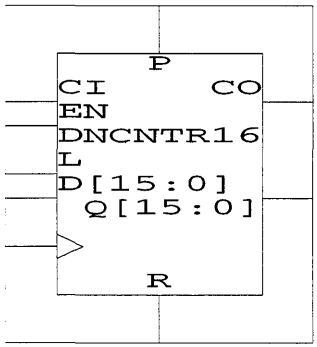

```
ENTITY dncntr8 IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d7, d6, d5, d4, d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         p: IN BIT ;
         q7, q6, q5, q4, q3, q2, q1, q0: OUT
BIT );
END dncntr8 ;
```

DESCRIPTION

INPUTS							OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**P**	**CO**	**Q**
X	X	X	X	X	H	X	L	L
X	X	X	X	X	L	H	L	H
L	X	X	Λ	X	L	L	L	Q_0
H	L	L	Λ	X	L	L	H if Q=00, else L	Q_0
H	H	L	Λ	X	L	L	H if Q=00, else L	Q_0-1
X	X	H	Λ	L	L	L	H	L
X	X	H	Λ	H	L	L	L	H

2.8. DNCNTR16

16-Bit Cascadable Down Counter.



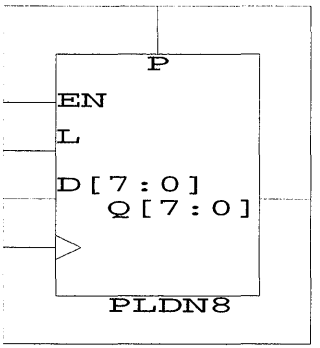

```
ENTITY dncntr16 IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d15, d14, d13, d12, d11, d10, d9, d8,
         d7, d6, d5, d4, d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         p: IN BIT ;
         q15, q14, q13, q12, q11, q10, q9, q8,
         q7, q6, q5, q4, q3, q2, q1, q0: OUT
         BIT );
END dncntr16 ;
```

DESCRIPTION

INPUTS							OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**P**	**CO**	**Q**
X	X	X	X	X	H	X	L	L
X	X	X	X	X	L	H	L	H
L	X	X	L	X	L	L	L	Q_0
H	L	L	L	X	L	L	H if Q=0000, else L	Q_0
H	H	L	L	X	L	L	H if Q=0000, else L	Q_0-1
X	X	H	L	L	L	L	H	L
X	X	H	L	H	L	L	L	H

2.9. PLDN8

8-Bit Pipelined Down Counter.



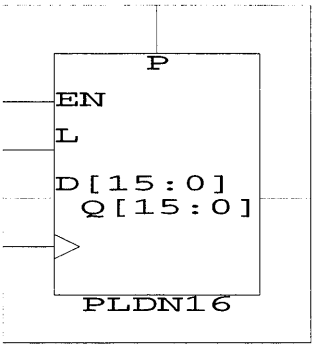

```
ENTITY pldn8 is
    PORT(EN: IN BIT;
         L: IN BIT;
         D7, D6, D5, D4, D3, D2, D1, D0: IN
BIT;
         CLK: IN BIT;
         P: IN BIT;
         Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0:
INOUT BIT);
END pldn8;
```

DESCRIPTION

INPUTS						OUTPUTS
EN	**L**	**CLK**	**D**	**R**	**P**	**Q**
X	X	X	X	H	X	L
X	X	X	X	L	H	H
L	L	Λ	X	L	L	Q_0
H	L	Λ	X	L	L	Q_0-1
X	H	Λ	L	L	L	L
X	H	Λ	H	L	L	H

2.10. PLDN16

16-Bit Pipelined Down Counter.



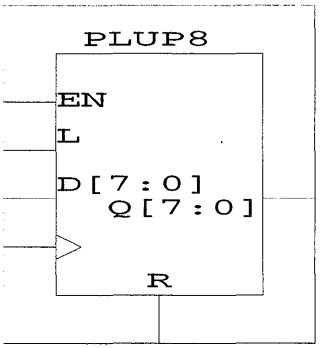

```
ENTITY pldn16 is
    PORT(EN: IN BIT;
         L: IN BIT;
         D15, D14, D13, D12, D11, D10, D9,
D8, D7, D6, D5, D4, D3, D2, D1, D0: IN BIT;
         CLK: IN BIT;
         P: IN BIT;
         Q15, Q14, Q13, Q12, Q11, Q10, Q9,
Q8, Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0: INOUT
BIT);
END pldn16;
```

DESCRIPTION

INPUTS						OUTPUTS
EN	**L**	**CLK**	**D**	**R**	**P**	**Q**
X	X	X	X	H	X	L
X	X	X	X	L	H	H
L	L	Λ	X	L	L	Q_0
H	L	Λ	X	L	L	Q_0-1
X	H	Λ	L	L	L	L
X	H	Λ	H	L	L	H

2.11. PLUP8

8-Bit Pipelined Up Counter.



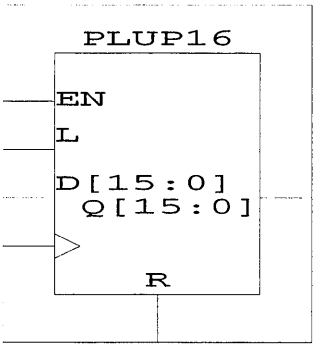

```
ENTITY plup8 is
    PORT(EN: IN BIT;
         L: IN BIT;
         D7, D6, D5, D4, D3, D2, D1, D0: IN
BIT;
         CLK: IN BIT;
         R: IN BIT;
         Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0:
INOUT BIT);
END plup8;
```

DESCRIPTION

INPUTS					OUTPUTS
EN	**L**	**CLK**	**D**	**R**	**Q**
X	X	X	X	H	L
L	L	Λ	X	L	Q_0
H	L	Λ	X	L	Q_0+1
X	H	Λ	L	L	L
X	H	Λ	H	L	H

2.12. PLUP16

16-Bit Pipelined Up Counter.



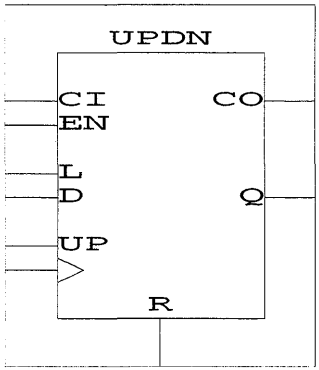

```
ENTITY plup16 is
    PORT(EN: IN BIT;
         L: IN BIT;
         D15, D14, D13, D12, D11, D10, D9,
D8, D7, D6, D5, D4, D3, D2, D1, D0: IN BIT;
         CLK: IN BIT;
         R: IN BIT;
         Q15, Q14, Q13, Q12, Q11, Q10, Q9,
Q8, Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0: INOUT
BIT);
END plup16;
```

DESCRIPTION

INPUTS					OUTPUTS
EN	**L**	**CLK**	**D**	**R**	**Q**
X	X	X	X	H	L
L	L	Λ	X	L	Q_0
H	L	Λ	X	L	Q_0+1
X	H	Λ	L	L	L
X	H	Λ	H	L	H

2.13. UPDN

1-Bit Cascadable Up-Down Counter.



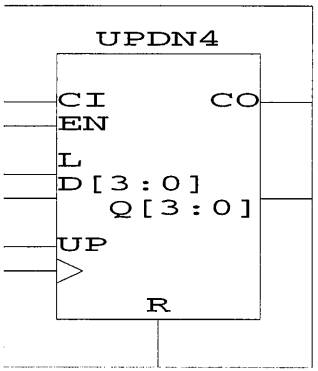

```
ENTITY updn IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         up: IN BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q: OUT BIT );
END updn ;
```

DESCRIPTION

INPUTS							OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**UP**	**CO**	**Q**
X	X	X	X	X	H	X	L	L
L	X	L	L	X	L	H	L	Q_0
H	L	L	L	X	L	H	Q_0	Q_0
H	H	L	L	X	L	H	$\overline{Q}_0$	$\overline{Q}_0$
X	X	H	L	L	L	H	L	L
X	X	H	L	H	L	H	H	H
L	X	L	L	X	L	L	L	Q_0
H	L	L	L	X	L	L	$\overline{Q}_0$	Q_0
H	H	L	L	X	L	L	Q_0	$\overline{Q}_0$
X	X	H	L	L	L	L	H	L
X	X	H	L	H	L	L	L	H

2.14. UPDN4

4-Bit Cascadable Up/Down Counter.



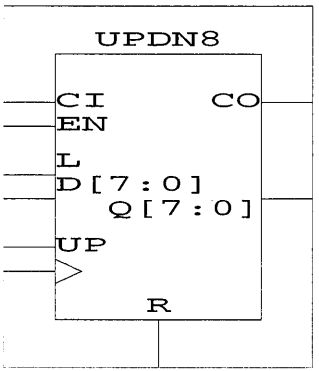

```
ENTITY updn4 IS
    PORT(ci: IN BIT ;
         co: INOUT BIT ;
         up: IN BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q3, q2, q1, q0: INOUT BIT );
END updn4 ;
```

DESCRIPTION

INPUTS							OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**UP**	**CO**	**Q**
X	X	X	X	X	H	X	L	L
L	X	L	Λ	X	L	H	L	Q_0
H	L	L	Λ	X	L	H	H if Q=F, else L	Q_0
H	H	L	Λ	X	L	H	H if Q=F, else L	Q_0+1
X	X	H	Λ	L	L	H	L	L
X	X	H	Λ	H	L	H	H	H
L	X	L	Λ	X	L	L	L	Q_0
H	L	L	Λ	X	L	L	H if Q=0, else L	Q_0
H	H	L	Λ	X	L	L	H if Q=0, else L	Q_0-1
X	X	H	Λ	L	L	L	H	L
X	X	H	Λ	H	L	L	L	H

2.15. UPDN8

8-Bit Cascadable Up/Down Counter.



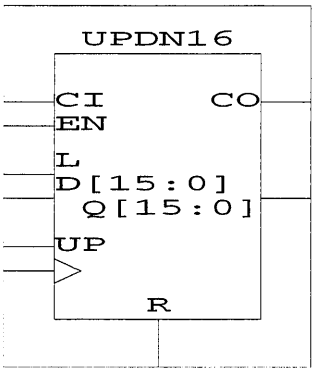

```
ENTITY updn8 IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         up: IN BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d7, d6, d5, d4, d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q7, q6, q5, q4, q3, q2, q1, q0: OUT
BIT );
END updn8 ;
```

DESCRIPTION

INPUTS							OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**UP**	**CO**	**Q**
X	X	X	X	X	H	X	L	L
L	X	L	Λ	X	L	H	L	Q_0
H	L	L	Λ	X	L	H	H if Q=FF, else L	Q_0
H	H	L	Λ	X	L	H	H if Q=FF, else L	Q_0+1
X	X	H	Λ	L	L	H	L	L
X	X	H	Λ	H	L	H	H	H
L	X	L	Λ	X	L	L	L	Q_0
H	L	L	Λ	X	L	L	H if Q=00, else L	Q_0
H	H	L	Λ	X	L	L	H if Q=00, else L	Q_0-1
X	X	H	Λ	L	L	L	H	L
X	X	H	Λ	H	L	L	L	H

2.16. UPDN16

16-Bit Cascadable Up/Down Counter.




```
ENTITY updn16 IS
    PORT(ci: IN BIT ;
         co: OUT BIT ;
         up: IN BIT ;
         en: IN BIT ;
         l: IN BIT ;
         d15, d14, d13, d12, d11, d10, d9, d8,
         d7, d6, d5, d4, d3, d2, d1, d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q15, q14, q13, q12, q11, q10, q9, q8,
         q7, q6, q5, q4, q3, q2, q1, q0: OUT
BIT );
END updn16 ;
```

DESCRIPTION

INPUTS							OUTPUTS	
CI	**EN**	**L**	**CLK**	**D**	**R**	**UP**	**CO**	**Q**
X	X	X	X	X	H	X	L	L
L	X	L	Λ	X	L	H	L	Q_0
H	L	L	Λ	X	L	H	H if Q=FFFF, else L	Q_0
H	H	L	Λ	X	L	H	H if Q=FFFF, else L	Q_0+1
X	X	H	Λ	L	L	H	L	L
X	X	H	Λ	H	L	H	H	H
L	X	L	Λ	X	L	L	L	Q_0
H	L	L	Λ	X	L	L	H if Q=0000, else L	Q_0
H	H	L	Λ	X	L	L	H if Q=0000, else L	Q_0-1
X	X	H	Λ	L	L	L	H	L
X	X	H	Λ	H	L	L	L	H

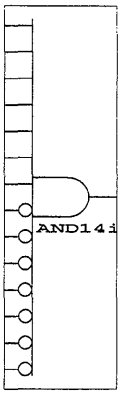
Chapter 3 Gates

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file; outside the entity and architecture declarations:

```
use work.gatespkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

3.1. AND14I7

14-input AND gate, 7 inputs inverted.



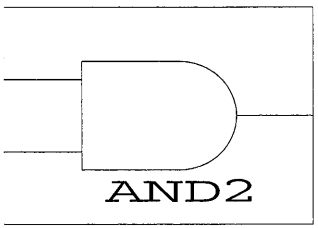

```
ENTITY and14i7 IS
    PORT(a,b,c,d,e,f,g,
         h,i,j,k,l,m,n: IN BIT;
         q: OUT BIT);
END and14i7 ;
```

DESCRIPTION

Q = A AND B AND C AND D AND E AND F AND G AND NOT H AND NOT I AND NOT J AND NOT K AND NOT L AND NOT M AND NOT N

3.2. AND2

2-Input Positive AND Gate.



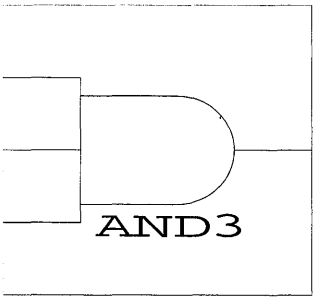

```
ENTITY and2 IS
    PORT(a, b: IN BIT ;
         q: OUT BIT );
END and2 ;
```

DESCRIPTION

Q = (A AND B).

3.3. AND3

3-Input Positive AND Gate.



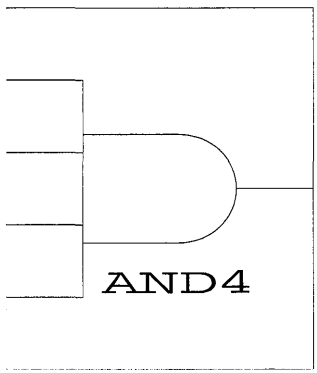

```
ENTITY and3 IS
    PORT(a, b, c: IN BIT ;
         q: OUT BIT );
END and3 ;
```

DESCRIPTION

Q = (A AND B AND C).

3.4. AND4

4-Input Positive AND Gate.



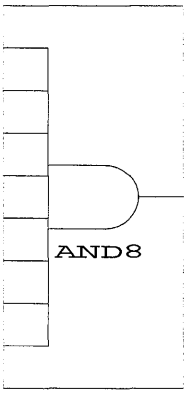

```
ENTITY and4 IS
    PORT(a, b, c, d: IN BIT ;
         q: OUT BIT );
END and4 ;
```

DESCRIPTION

Q = (A AND B AND C AND D).

3.5. AND8

8-Input Positive AND Gate.




```
ENTITY and8 IS
    PORT(a, b, c, d, e, f, g, h: IN BIT ;
         q: OUT BIT );
END and8 ;
```

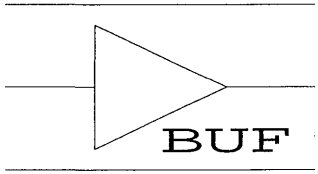
DESCRIPTION

Q = (A AND B AND C AND D AND E AND F AND G AND H).

3.6. BUF

Buffer.

```
ENTITY buf IS
    PORT(x: IN BIT ;
         y: OUT BIT );
END buf ;
```

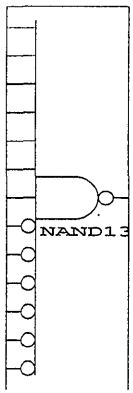


DESCRIPTION

INPUT	OUTPUT
X	Y
L	L
H	H

3.7. NAND13I6

13-input NAND gate, 6 inputs inverted.



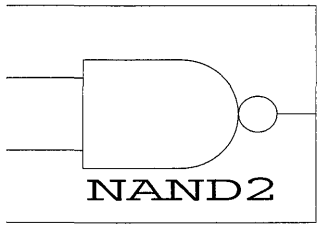

```
ENTITY nand13i6 IS
    PORT(a,b,c,d,e,f,g,
         h,i,j,k,l,m: IN BIT;
         qn: OUT BIT);
END nand13i6;
```

DESCRIPTION

QN = NOT (A AND B AND C AND D AND E AND F AND G AND NOT H AND NOT I AND NOT J AND NOT K AND NOT L AND NOT M)

3.8. NAND2

2-Input Positive NAND Gate.



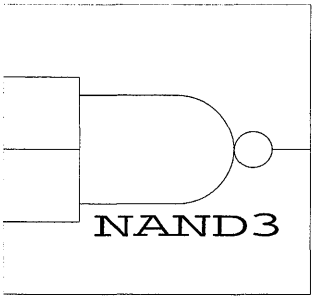

```
ENTITY nand2 IS
    PORT(a, b: IN BIT ;
         qn: OUT BIT );
END nand2 ;
```

DESCRIPTION

QN = NOT(A AND B).

3.9. NAND3

3-Input Positive NAND Gate.



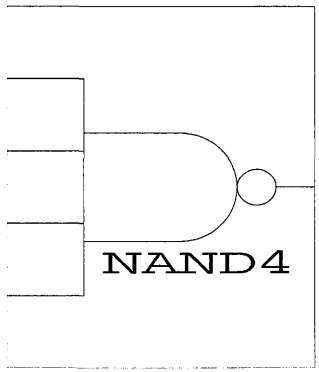

```
ENTITY nand3 IS
    PORT(a, b, c: IN BIT ;
         qn: OUT BIT );
END nand3 ;
```

DESCRIPTION

QN = NOT(A AND B AND C).

3.10. NAND4

4-Input Positive NAND Gate.



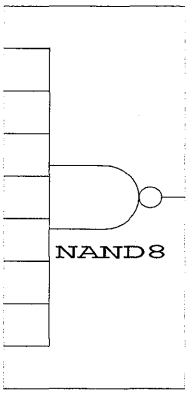

```
ENTITY nand4 IS
    PORT(a, b, c, d: IN BIT ;
         qn: OUT BIT );
END nand4 ;
```

DESCRIPTION

QN = NOT(A AND B AND C AND D).

3.11. NAND8

8-Input Positive NAND Gate.



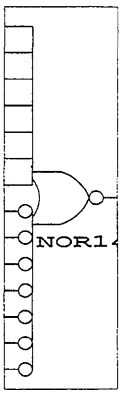

```
ENTITY nand8 IS
    PORT(a, b, c, d, e, f, g, h: IN BIT ;
         qn: OUT BIT );
END nand8 ;
```

DESCRIPTION

QN = NOT(A AND B AND C AND D AND E AND F AND G AND H).

3.12. NOR14I7

14-input NOR gate, 7 inputs inverted.



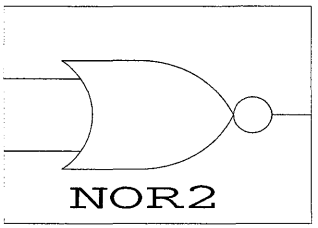

```
ENTITY nor14i7 IS
    PORT(a,b,c,d,e,f,g,
         h,i,j,k,l,m,n: IN BIT;
         qn: OUT BIT);
END nor14i7;
```

DESCRIPTION

QN = NOT(A OR B OR C OR D OR E OR F OR G OR NOT H OR NOT I OR NOT J OR NOT K OR NOT L OR NOT M OR NOT N)

3.13. NOR2

2-Input Positive NOR Gate.



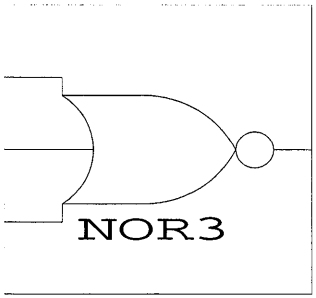

```
ENTITY nor2 IS
    PORT(a, b: IN BIT ;
         qn: OUT BIT );
END nor2 ;
```

DESCRIPTION

QN = NOT(A OR B).

3.14. NOR3

3-Input Positive NOR Gate.



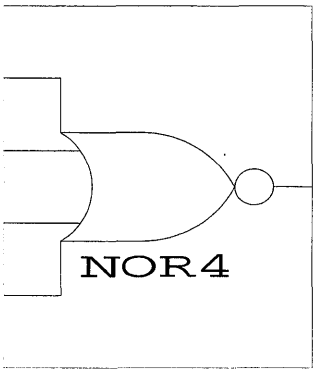

```
ENTITY nor3 IS
    PORT(a, b, c: IN BIT ;
         qn: OUT BIT );
END nor3 ;
```

DESCRIPTION

QN = NOT(A OR B OR C).

3.15. NOR4

4-Input Positive NOR Gate.



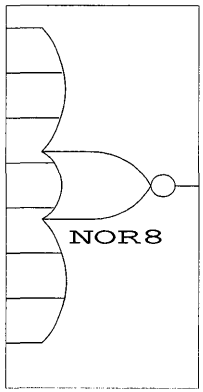

```
ENTITY nor4 IS
    PORT(a, b, c, d: IN BIT ;
         qn: OUT BIT );
END nor4 ;
```

DESCRIPTION

QN = NOT(A OR B OR C OR D).

3.16. NOR8

8-Input Positive NOR Gate.



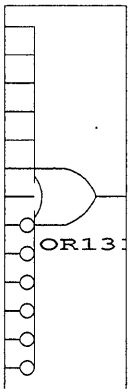

```
ENTITY nor8 IS
    PORT(a, b, c, d, e, f, g, h: IN BIT ;
         qn: OUT BIT );
END nor8 ;
```

DESCRIPTION

QN = NOT(A OR B OR C OR D OR E OR F OR G OR H).

3.17. OR13I6

13-input OR gate, 6 inputs inverted.



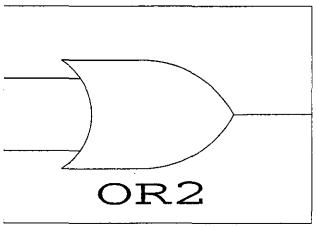

```
ENTITY or13i6 IS
    PORT(a,b,c,d,e,f,g,
         h,i,j,k,l,m: IN BIT;
         q: OUT BIT);
END or13i6;
```

DESCRIPTION

Q = A OR B OR C OR D OR E OR F OR G OR NOT H OR NOT I OR NOT J OR NOT K OR NOT L OR NOT M

3.18. OR2

2-Input Positive OR Gate.



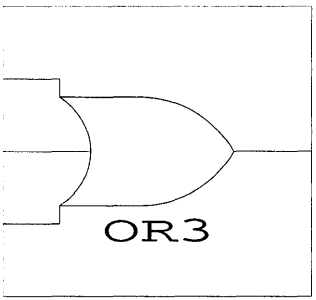

```
ENTITY or2 IS
    PORT(a, b: IN BIT ;
         q: OUT BIT );
END or2 ;
```

DESCRIPTION

Q = (A OR B).

3.19. OR3

3-Input Positive OR Gate.



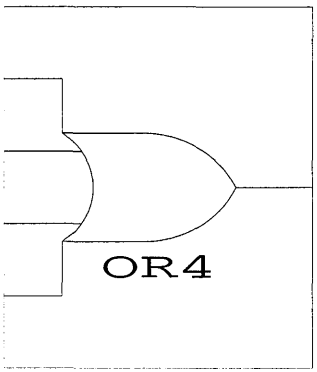

```
ENTITY or3 IS
    PORT(a, b, c: IN BIT ;
         q: OUT BIT );
END or3 ;
```

DESCRIPTION

Q = (A OR B OR C).

3.20. OR4

4-Input Positive OR Gate.



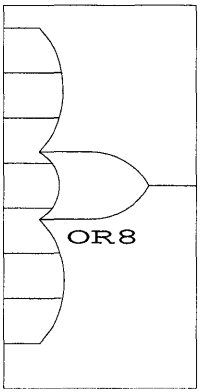

```
ENTITY or4 IS
    PORT(a, b, c, d: IN BIT ;
         q: OUT BIT );
END or4 ;
```

DESCRIPTION

Q = (A OR B OR C OR D).

3.21. OR8

8-Input Positive OR Gate.



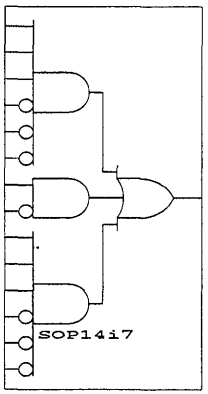

```
ENTITY or8 IS
    PORT(a, b, c, d, e, f, g, h: IN BIT ;
         q: OUT BIT );
END or8 ;
```

DESCRIPTION

Q = (A OR B OR C OR D OR E OR F OR G OR H).

3.22. SOP14I7

14-input sum-of-products component, 7 inputs inverted.




```
ENTITY sop14i7 IS
    PORT(a,b,c,d,e,f,g,
         h,i,j,k,l,m,n: IN BIT;
         q: OUT BIT);
END sop14i7;
```

DESCRIPTION

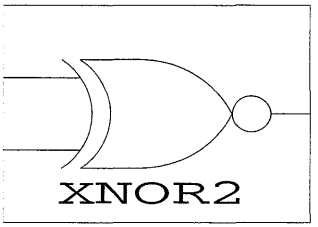
Q = (A AND B AND C AND NOT D AND NOT E AND NOT F)

OR (G AND NOT H)

OR (I AND J AND K AND NOT L AND NOT M AND NOT N)

3.23. XNOR2

2-Input Exclusive-NOR Gate.



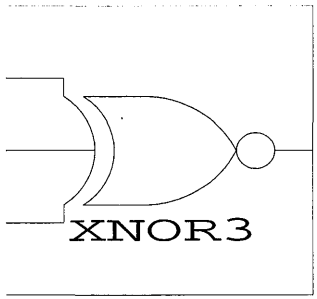

```
ENTITY xnor2 IS
    PORT(a, b: IN BIT ;
         q: OUT BIT );
END xnor2 ;
```

DESCRIPTION

Q = NOT(A XOR B).

3.24. XNOR3

3-Input Exclusive-NOR gate.



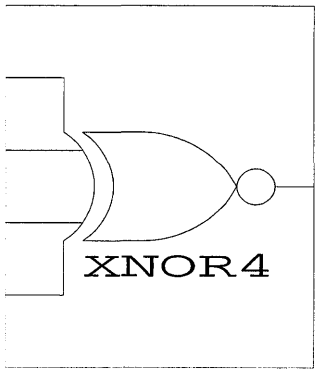

```
ENTITY xnor3 IS
    PORT(a, b, c: IN BIT ;
         q: OUT BIT );
END xnor3 ;
```

DESCRIPTION

Q = NOT(A XOR B XOR C).

3.25. XNOR4

4-Input Exclusive-NOR Gate.



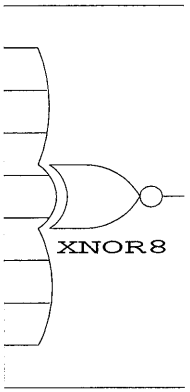

```
ENTITY xnor4 IS
     PORT(a, b, c, d: IN BIT ;
          q: OUT BIT );
END xnor4 ;
```

DESCRIPTION

Q = NOT(A XOR B XOR C XOR D).

3.26. XNOR8

8-Input Exclusive-NOR Gate.



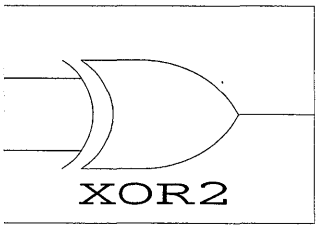

```
ENTITY xnor8 IS
    PORT(a, b, c, d, e, f, g, h: IN BIT ;
         q: OUT BIT );
END xnor8 ;
```

DESCRIPTION

Q = NOT(A XOR B XOR C XOR D XOR E XOR F XOR G XOR H).

3.27. XOR2

2-Input Exclusive-OR Gate.



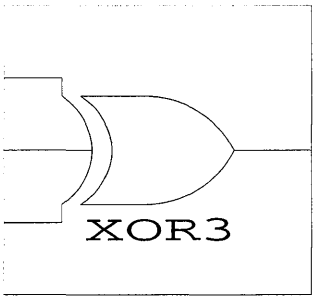

```
ENTITY xor2 IS
    PORT(a, b: IN BIT ;
         q: OUT BIT );
END xor2 ;
```

DESCRIPTION

Q = (A XOR B).

3.28. XOR3

3-Input Exclusive-OR Gate.



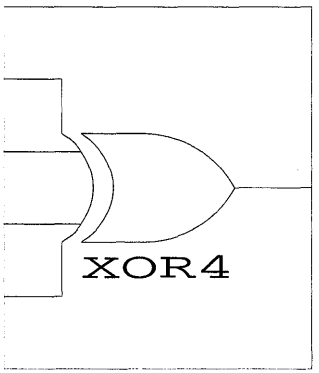

```
ENTITY xor3 IS
    PORT(a, b, c: IN BIT ;
         q: OUT BIT );
END xor3 ;
```

DESCRIPTION

Q = (A XOR B XOR C).

3.29. XOR4

4-Input Exclusive-OR Gate.



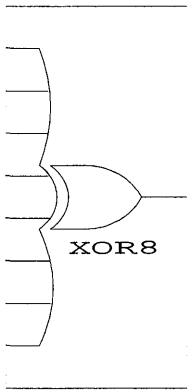

```
ENTITY xor4 IS
     PORT(a, b, c, d: IN BIT ;
          q: OUT BIT );
END xor4 ;
```

DESCRIPTION

Q = (A XOR B XOR C XOR D).

3.30. XOR8

8-Input Exclusive-OR Gate.




```
ENTITY xor8 IS
    PORT(a, b, c, d, e, f, g, h: IN BIT ;
         q: OUT BIT );
END xor8 ;
```

DESCRIPTION

Q = (A XOR B XOR C XOR D XOR E XOR F XOR G XOR H).

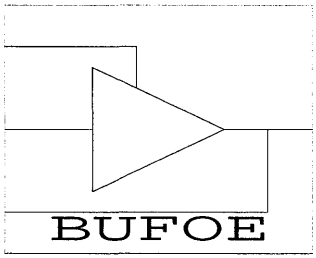
Chapter 4 I/O

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediately above the entity and architecture declarations:

```
use work.iopkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

4.31. BUFOE

3-State Buffer with Feed Back.




```
ENTITY bufoe IS
    PORT(x: IN BIT ;
         oe: IN BIT ;
         y: INOUT X01Z ;
         yfb: OUT BIT );
END bufoe ;
```

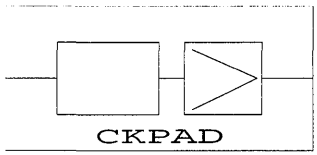
DESCRIPTION

INPUTS		OUTPUT	
X	**OE**	**Y**	**YFB**
X	L	Z	X
L	H	L	L
H	H	H	H

4.32. CKPAD

Clock Pin Input Pad (pASIC).

```
ENTITY ckpad IS
    PORT(p: IN BIT ;
         q: OUT BIT );
END ckpad ;
```



DESCRIPTION

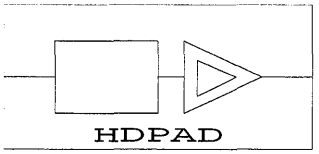
INPUT	OUTPUT
P	**Q**
L	L
H	H

The CKPAD is a special input pin for the pASIC devices. It is realized as a buffer for other devices.

4.33. HDPAD

High Drive PinInput Pad (pASIC).

```
ENTITY hdpad IS
    PORT(p: IN BIT ;
         q: OUT BIT );
END hdpad ;
```



DESCRIPTION

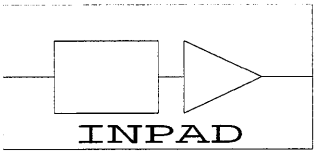
INPUT	OUTPUT
P	Q
L	L
H	H

The HDPAD is a special input pin for the pASIC devices. It is realized as a buffer for other devices.

4.34. INPAD

PinInput Pad (pASIC).

```
ENTITY inpad IS
    PORT(p: IN BIT ;
         q: OUT BIT );
END inpad ;
```



DESCRIPTION

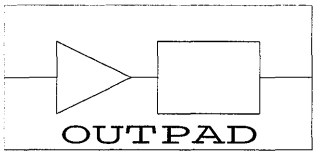
INPUT	OUTPUT
P	Q
L	L
H	H

The INPAD is an input pin for the pASIC devices. It is realized as a buffer for other devices.

4.35. OUTPAD

Pin Output Pad (pASIC).

```
ENTITY outpad IS
    PORT(a: IN BIT ;
         p: OUT BIT );
END outpad ;
```



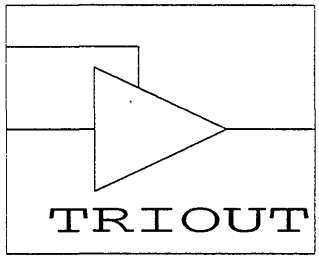
DESCRIPTION

INPUT	OUTPUT
A	P
L	L
H	H

The OUTPAD is an output pin for the pASIC devices. It is realized as a buffer for other devices.

4.36. TRIOUT

3-State Buffer.




```
ENTITY triout IS
    PORT(x: IN BIT ;
         oe: IN BIT ;
         y: OUT X01Z );
END triout ;
```

DESCRIPTION

X	**OE**	**Y**
X	L	Z
L	H	L
H	H	H

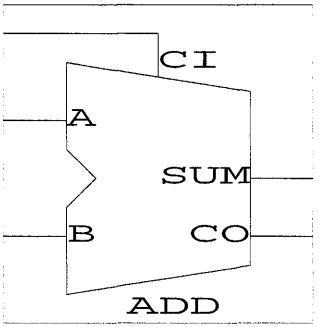
Chapter 5 Math

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediately above the entity and architecture declarations:

```
use work.mathpkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

5.1. ADD

1-Bit Full Adder.




```
ENTITY add IS
    PORT(ci: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         sum: OUT BIT ;
         co: OUT BIT );
END add ;
```

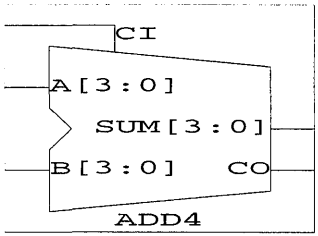
DESCRIPTION

CO = (A AND B) OR (A AND CI) OR (B AND CI)

SUM = (A XOR B) XOR C

5.2. ADD4

4-Bit Full Adder.




```
ENTITY add4 IS
    PORT(ci: IN BIT;
         a3, a2, a1, a0: IN BIT;
         b3, b2, b1, b0: IN BIT ;
         sum3, sum2, sum1, sum0: OUT BIT;
         co: OUT BIT );
END add4 ;
```

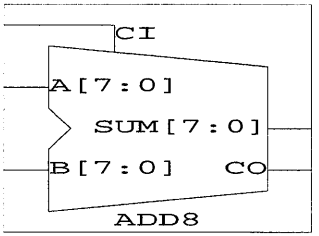
DESCRIPTION

SUM = A + B + CI

A 4-bit adder with 2 stage carry-look-ahead. It is not as fast as the FCADD4, but is more space efficient in most cases. In the case of the pASIC devices, it is identical to the FCADD4.

5.3. ADD8

8-Bit Full Adder.




```
ENTITY add8 IS
    PORT(ci: IN BIT;
         a7, a6, a5, a4, a3, a2, a1, a0: IN BIT;
         b7, b6, b5, b4, b3, b2, b1, b0: IN BIT;
         sum7, sum6, sum5, sum4, sum3,
         sum2, sum1, sum0, co: OUT BIT );
END add8 ;
```

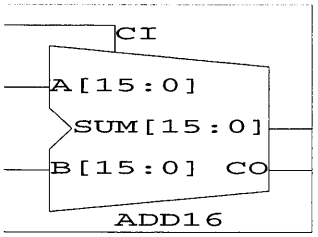
DESCRIPTION

SUM = A + B + CI

An 8-bit adder with 2 stage carry-look-ahead. It is not as fast as the FCADD8, but is more space efficient in most cases. In the case of the pASIC devices, it is identical to the FCADD8.

5.4. ADD16

16-Bit Full Adder.




```
ENTITY add16 IS
    PORT(ci: IN BIT;
        a15,a14,a13,a12,a11,a10,a9: IN BIT;
        a8,a7,a6,a5,a4,a3,a2,a1 a0: IN BIT;
        b15,b14,b13,b12,b11,b10,b9: IN BIT;
        b8,b7,b6,b5,b4,b3,b2,b1,b0: IN BIT;
        sum15,sum14,sum13,sum12: OUT BIT;
        sum11,sum10,sum9,sum8: OUT BIT;
        sum7,sum6,sum5,sum4,sum3: OUT BIT;
        sum2,sum1,sum0,co: OUT BIT );
END add16 ;
```

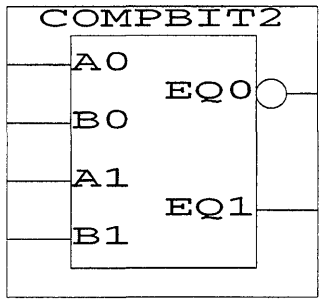
DESCRIPTION

SUM = A + B + CI

A 16-bit adder with 2 stage carry-look-ahead. It is not as fast as the FCADD16, but is more space efficient in most cases. In the case of the pASIC devices, it utilizes the FCADD16.

5.5. COMPBIT2

Two-bit comparator.




```
ENTITY compbit2 IS
    PORT(a0,b0,a1,b1: IN BIT;
         eq0,eq1: INOUT BIT);
END compbit2;
```

DESCRIPTION

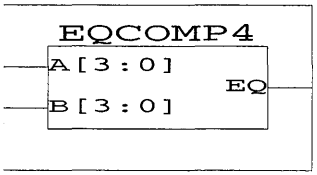
EQ0 = NOT(A0 XOR B0)

EQ1 = (A1 XOR B1)

5.6. EQCOMP4

4-Bit Equal Compare.

```
ENTITY eqcomp4 IS
    PORT(a3, a2, a1, a0, b3, b2, b1, b0: IN BIT ;
         eq: OUT BIT );
END eqcomp4 ;
```

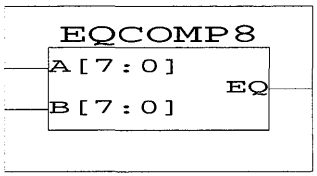


DESCRIPTION

EQ = NOT (A3 XOR B3) AND
NOT (A2 XOR B2) AND
NOT (A1 XOR B1) AND
NOT (A0 XOR B0)

5.7. EQCOMP8

8-Bit Equal Compare.




```
ENTITY eqcomp8 IS
    PORT(a7, a6, a5, a4, a3, a2, a1, a0: IN BIT ;
         b7, b6, b5, b4, b3, b2, b1, b0: IN BIT ;
         eq: OUT BIT );
END eqcomp8 ;
```

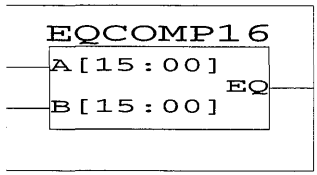
U

DESCRIPTION

EQ = NOT (A7 XOR B7) AND
NOT (A6 XOR B6) AND
NOT (A5 XOR B5) AND
NOT (A4 XOR B4) AND
NOT (A3 XOR B3) AND
NOT (A2 XOR B2) AND
NOT (A1 XOR B1) AND
NOT (A0 XOR B0)

5.8. EQCOMP16

16-Bit Equal Compare.



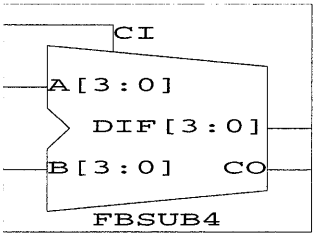

```
ENTITY eqcomp16 IS
    PORT(a15,a14,a13,a12,a11,a10,a9:IN BIT;
         a8,a7,a6,a5,a4,a3,a2,a1,a0: IN BIT ;
         b15,b14,b13,b12,b11,b10,b9: IN BIT;
         b8,b7,b6,b5,b4,b3,b2,b1,b0: IN BIT ;
         eq: OUT BIT);
END eqcomp16 ;
```

DESCRIPTION

EQ = NOT (A15 XOR B15) AND
NOT (A14 XOR B14) AND
NOT (A13 XOR B13) AND
NOT (A12 XOR B12) AND
NOT (A11XOR B11) AND
NOT (A10 XOR B10) AND
NOT (A9 XOR B9) AND
NOT (A8 XOR B8) AND
NOT (A7 XOR B7) AND
NOT (A6 XOR B6) AND
NOT (A5 XOR B5) AND
NOT (A4 XOR B4) AND
NOT (A3 XOR B3) AND
NOT (A2 XOR B2) AND
NOT (A1 XOR B1) AND
NOT (A0 XOR B0)

5.9. FBSUB4

4-Bit Fast Borrow Subtract.




```
ENTITY fbsub4 IS
    PORT(ci: IN BIT ;
         a3,a2,a1,a0: IN BIT ;
         b3,b2,b1,b0: IN BIT ;
         dif3,dif2,dif1,dif0: OUT BIT ;
         co: OUT BIT );
END fbsub4 ;
```

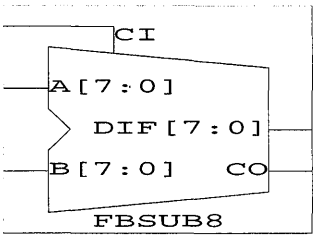
DESCRIPTION

DIF = A - B - CI

A 4-bit subtractor using a fast borrow-look-ahead architecture.

5.10. FBSUB8

8-Bit Fast Borrow Subtract.




```
ENTITY fbsub8 IS
    PORT(ci: IN BIT ;
         a7,a6,a5,a4,a3,a2,a1,a0: IN BIT ;
         b7,b6,b5,b4,b3,b2,b1,b0: IN BIT ;
         dif7.dif6,dif5,dif4: OUT BIT ;
         dif3,dif2,dif1,dif0: OUT BIT ;
         co: OUT BIT );
END fbsub8 ;
```

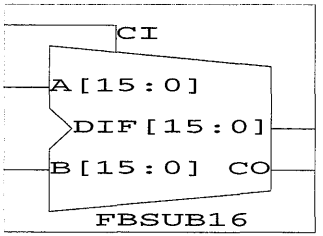
DESCRIPTION

DIF = A - B - CI

An 8-bit subtractor using a fast borrow-look-ahead architecture in the pASIC devices. For other devices, it uses 4-stage fast borrow-look-ahead.

5.11. FBSUB16

16-Bit Fast Borrow Subtract.




```
ENTITY fbsub16 IS
    PORT(ci: IN BIT ;
         a15,a14,a13,a12,a11,a10,a9: IN BIT;
         a8,a7a,6,a5,a4,a3,a2,a1,a0: IN BIT;
         b15,b14,b13,b12,b11,b10,b9: IN BIT;
         b8,b7,b6,b5,b4,b3,b2,b1,b0: IN BIT;
         dif15,dif14,dif13,dif12: OUT BIT;
         dif11,dif10,dif9,dif8: OUT BIT;
         dif7,dif6,dif5,dif4dif3: OUT BIT;
         dif2,dif1,dif0,co: OUT BIT);
END fbsub16 ;
```

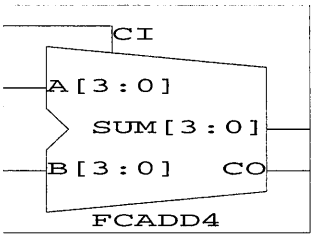
DESCRIPTION

DIF = A - B - CI

A 16-bit subtractor using a fast borrow-look-ahead architecture in the pASIC devices. For other devices, it uses 4-stage fast borrow-look-ahead.

5.12. FCADD4

4-Bit Fast Carry Add.




```
ENTITY fcadd4 IS
    PORT(ci: IN BIT;
         a3, a2, a1, a0 IN BIT;
         b3, b2, b1, b0: IN BIT;
         sum3, sum2, sum1, sum0: OUT BIT;
         co: OUT BIT );
END fcadd4 ;
```

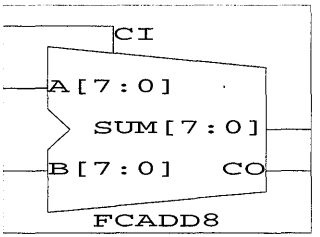
DESCRIPTION

SUM = A + B + CI

A 4-bit adder using a fast carry-look-ahead architecture.

5.13. FCADD8

8-Bit Fast Carry Add.




```
ENTITY fcadd8 IS
    PORT(ci: IN BIT;
         a7,a6,a5,a4,a3,a2,a1,a0: IN BIT;
         b7,b6,b5,b4,b3,b2,b1,b0: IN BIT ;
         sum7,sum6,sum5,sum4: OUT BIT;
         sum3,sum2,sum1,sum0: OUT BIT;
         co: OUT BIT );
END fcadd8 ;
```

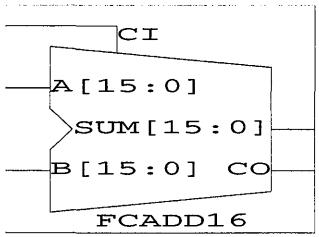
DESCRIPTION

SUM = A + B + CI

An 8-bit adder using a fast carry-look-ahead architecture in the pASIC devices. For other devices, it uses 4-stage fast carry-look-ahead.

5.14. FCADD16

16-Bit Fast Carry Add.




```
ENTITY fcadd16 IS
    PORT(ci: IN BIT;
        a15,a14,a13,a12,a11,a10,a9: IN BIT;
        a8,a7,a6,a5,a4,a3,a2,a1,a0: IN BIT;
        b15,b14,b13,b12,b11,b10,b9: IN BIT;
        b8,b7,b6,b5,b4,b3,b2,b1,b0: IN BIT;
        sum15,sum14,sum13,sum12: OUT BIT;
        sum11,sum10,sum9,sum8: OUT BIT;
        sum7,sum6,sum5,sum4,sum3: OUT BIT;
        sum2,sum1,sum0,co: OUT BIT );
END fcadd16 ;
```

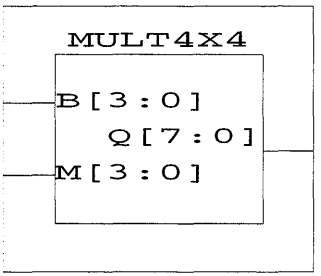
DESCRIPTION

SUM = A + B + CI

A 16-bit adder using a fast carry-look-ahead architecture in the pASIC devices. For other devices, it uses 4-stage fast carry-look-ahead.

5.15. MULT4X4

4-Bit by 4-Bit Binary Multiplier.




```
ENTITY mult4x4 IS
    PORT(b3,b2, b1,b0,m3,m2,m1,m0: IN BIT ;
         q7,q6,q5,q4,q3,q2,q1,q0: OUT BIT );
END mult4x4 ;
```

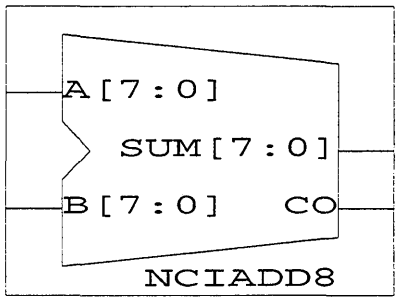
DESCRIPTION

Q = B X M

This element multiplies two 4-bit inputs and generates an 8-bit output. The range is 0 to 225 (decimal).

5.16. NCIADD8

8-bit adder, no carry input.



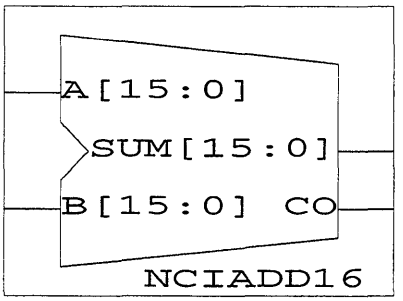

```
ENTITY nciadd8 is
    PORT(a7,a6,a5,a4,a3,a2,a1,a0,
         b7,b6,b5,b4,b3,b2,b1,b0 : IN bit;
         sum7,sum6,sum5,sum4,
         sum3,sum2,sum1,sum0,
         co: INOUT bit);
END nciadd8 ;
```

DESCRIPTION

SUM = A + B

5.17. NCIADD16

16-bit adder, no carry input.



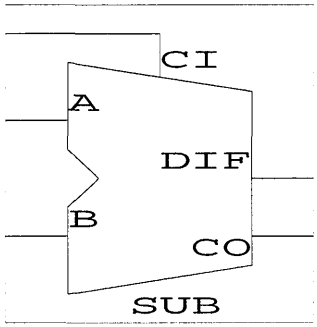

```
ENTITY nciadd16 IS
    PORT(ci,a15,a14,a13,a12,a11,a10,a9,a8,
         a7,a6,a5,a4,a3,a2,a1,a0,
         b15,b14,b13,b12,b11,b10,b9,b8,
         b7,b6,b5,b4,b3,b2,b1,b0:
         IN bit;
         sum15,sum14,sum13,sum12,
         sum11,sum10,sum9,sum8,
         sum7,sum6,sum5,sum4,sum3,
         sum2,sum1,sum0,co:
         INOUT bit);
END nciadd16 ;
```

DESCRIPTION

SUM = A + B

5.18. SUB

1-Bit Subtractor.




```
ENTITY sub IS
    PORT(ci: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         dif: OUT BIT ;
         co: OUT BIT );
END sub ;
```

DESCRIPTION

DIF = NOT(NOT(A XOR B) XOR CI)

CO = (NOT A AND B) OR ((NOT A AND CI) OR (B AND CI))

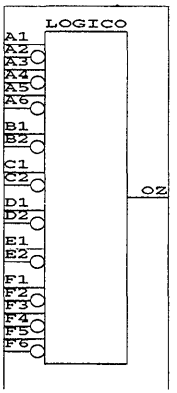
Chapter 6 Macrocells

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediately above the entity and architecture declarations:

```
use work.mcpartspkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

6.1. LOGICO

pASIC380 FPGA internal logic cell, minus register.



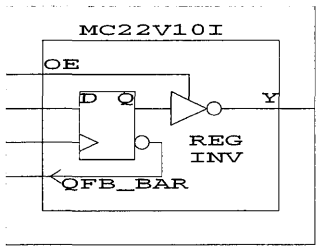

```
ENTITY logico is
    PORT(a1,a2,a3,a4,a5,a6: IN BIT;
         b1,b2: IN BIT;
         c1,c2: IN BIT;
         d1,d2: IN BIT;
         e1,e2: IN BIT;
         f1,f2,f3,f4,f5,f6: IN BIT;
         oz: INOUT BIT);
END logico;
```

DESCRIPTION

OZ <= ((EL AND NOT (E2) AND
F1 AND NOT (F2) AND F3 AND NOT (F4) AND F5 AND NOT (F6)) OR
(D1 AND NOT (D2) AND
NOT (F1 AND NOT (F2) AND F3 AND NOT (F4) AND F5 AND NOT (F6))) AND
A1 AND NOT (A2) AND A3 AND NOT (A4) AND A5 AND NOT (A6)) OR
((C1 AND NOT (C2) AND
F1 AND NOT (F2) AND F3 AND NOT (F4) AND F5 AND NOT (F6)) OR
(B1 AND NOT (B2) AND
NOT (F1 AND NOT (F2) AND F3 AND NOT (F4) AND F5 AND NOT (F6))) AND
NOT (A1 AND NOT (A2) AND A3 AD NOT (AR) AND A5 AND NOT (A6)));

6.2. MC22V10I

Inverting Output Macro-cell for C22V10.



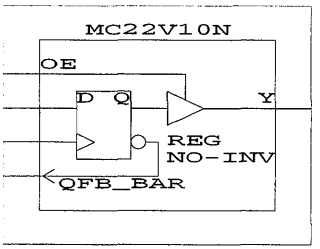

```
ENTITY mc22v10i IS
    PORT(d: IN BIT ;
         oe: IN BIT ;
         clk: IN BIT ;
         y: INOUT X01Z ;
         qfb_bar: OUT BIT );
END mc22v10i ;
```

DESCRIPTION

SELECT			IN/OUT	OUTPUT
D	**OE**	**CLK**	**Y**	**QFB_BAR**
X	L	X	Z	$\overline{Q}$
L	H	Λ	H	H
H	H	Λ	L	L

6.3. MC22V10N

Non-inverting Output Macro-cell for C22V10..



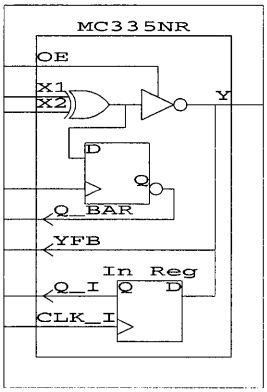

```
ENTITY mc22v10n IS
    PORT(d: IN BIT ;
         oe: IN BIT ;
         clk: IN BIT ;
         y: INOUT X01Z ;
         qfb_bar: OUT BIT );
END mc22v10n ;
```

DESCRIPTION

SELECT			IN/OUT	OUTPUT
D	**OE**	**CLK**	**Y**	**QFB_BAR**
X	L	X	Z	$\overline{Q}$
L	H	Λ	L	H
H	H	Λ	H	L

6.4. MC335NR

Non-registered Input/Output Macro-cell for C335.




```
ENTITY mc335nr IS
    PORT(x1: IN BIT ;
         x2: IN BIT ;
         clk: IN BIT ;
         clk_i: IN BIT ;
         oe: IN BIT ;
         q_bar: OUT BIT ;
         q_i: OUT BIT ;
         y: INOUT X01Z ;
         yfb: OUT BIT );
END mc335nr ;
```

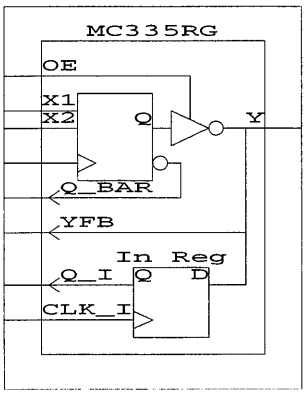
DESCRIPTION

INPUTS					OUTPUTS			IN/OUT
X1	X2	CLK	CLI_I	OE	Q_BAR	Q_I	YFB	Y
X	X	L	L	L	$\overline{Q}$	Q_I	Y	Z
L	L	L	Λ	H	$\overline{Q}$	H	H	H
L	H	L	Λ	H	$\overline{Q}$	L	L	L
H	L	L	Λ	H	$\overline{Q}$	L	L	L
H	H	L	Λ	H	$\overline{Q}$	H	H	H
L	L	Λ	Λ	H	H	H	H	H
L	H	Λ	Λ	H	L	L	L	L
H	L	Λ	Λ	H	L	L	L	L
H	H	Λ	Λ	H	H	H	H	H
X	X	L	Λ	L	$\overline{Q}$	L	L	L ψ
X	X	L	Λ	L	$\overline{Q}$	H	H	H ψ

ψ - "Y" as an input.

6.5. MC335RG

Registered Input/Output Macro-cell for C335.




```
ENTITY mc335rg IS
    PORT(x1: IN BIT ;
         x2: IN BIT ;
         clk: IN BIT ;
         clk_i: IN BIT ;
         oe: IN BIT ;
         q_bar: OUT BIT ;
         q_i: OUT BIT ;
         y: INOUT X01Z ;
         yfb: OUT BIT );
END mc335rg ;
```

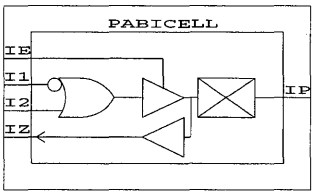
DESCRIPTION

INPUTS					OUTPUTS			IN/OUT
X1	X2	CLK	CLI_I	OE	Q_BAR	Q_I	YFB	Y
X	X	L	L	L	$\overline{Q}$	Q_I	Y	Z
L	L	Λ	Λ	H	H	Y_0	H	H
L	H	Λ	Λ	H	L	Y_0	L	L
H	L	Λ	Λ	H	L	Y_0	L	L
H	H	Λ	Λ	H	H	Y_0	H	H
X	X	L	Λ	L	$\overline{Q}$	L	L	L ψ
X	X	L	Λ	L	$\overline{Q}$	H	H	H ψ

ψ - "Y" as an input.

6.6. PABICELL

pASIC Bi-directional Master Cell.




```
ENTITY pabicell IS
    PORT(ie: IN BIT ;
         i1: IN BIT ;
         i2: IN BIT ;
         iz: OUT BIT ;
         ip: INOUT X01Z );
END pabicell ;
```

DESCRIPTION

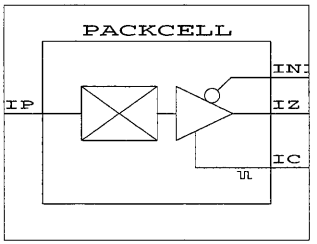
INPUTS			IN/OUT	OUTPUT
IE	**I1**	**I2**	**IZ**	**IP**
L	X	X	X	Z
H	L	L	H	H
H	H	L	L	L
H	L	H	H	H
H	H	H	H	H

The PABICELL implements the I/O pads other than the high-drive elements for pASIC devices. It is implemented as follows for other devices:

```
ory <= i2 or not i1;
U1: bufoe port map (oe => ie, x => ory, yfb => iz, y => ip);
```

6.7. PACKCELL

pASIC Clock Master Cell.




```
ENTITY packcell IS
    PORT(ip: IN BIT ;
         ini: OUT BIT ;
         iz: OUT BIT ;
         ic: OUT BIT );
END packcell ;
```

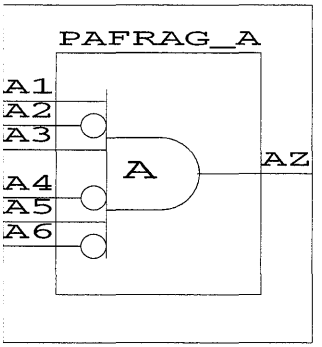
DESCRIPTION

INPUT	OUTPUTS		
IP	**INI**	**IZ**	**IC**
H	L	H	H
L	H	L	L

The PACKCELL implements the clock input pad for pASIC devices. It is implemented as two buffers and an inverter for other devices.

6.8. PAFRAG_A

pASIC Macro Cell "A" Fragment.




```
ENTITY pafrag_a IS
    PORT(a1: IN BIT ;
         a2: IN BIT ;
         a3: IN BIT ;
         a4: IN BIT ;
         a5: IN BIT ;
         a6: IN BIT ;
         az: OUT BIT );
END pafrag _a;
```

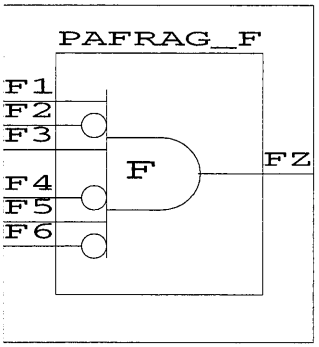
DESCRIPTION

The PAFRAG_A implements the "A" fragment of the pASIC macro cell. It is implemented with combinatorial logic in other devices. The logic equivalent is:

az <= a1 AND NOT(a2) AND a3 AND NOT(a4) AND a5 AND NOT(a6)

6.9. PAFRAG_F

pASIC Macro Cell "F" Fragment.




```
ENTITY pafrag_f IS
    PORT(f1: IN BIT ;
         f2: IN BIT ;
         f3: IN BIT ;
         f4: IN BIT ;
         f5: IN BIT ;
         f6: IN BIT ;
         fz: OUT BIT );
END pafrag _f;
```

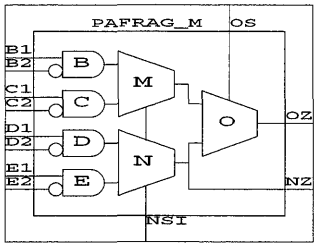
DESCRIPTION

The PAFRAG_F implements the "F" fragment of the pASIC macro cell. It is implemented with combinatorial logic in other devices. The logic equivalent is:

fz <= f1 AND NOT(f2) AND f3 AND NOT(f4) AND f5 AND NOT(f6)

6.10. PAFRAG_M

pASIC Macro Cell "M" Fragment.




```
ENTITY pafrag_m IS
    PORT(os: IN BIT ;
         nsi: IN BIT ;
         b1: IN BIT ;
         b2: IN BIT ;
         c1: IN BIT ;
         c2: IN BIT ;
         d1: IN BIT ;
         d2: IN BIT ;
         e1: IN BIT ;
         e2: IN BIT ;
         oz: OUT BIT ;
         nz: OUT BIT );
END pafrag _m;
```

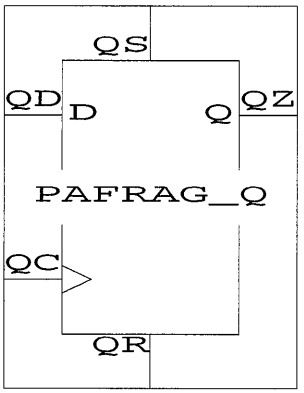
DESCRIPTION

The PAFRAG_M implements the "M" fragment of the pASIC macro cell. It is implemented with combinatorial logic in other devices. The logic equivalent is:

```
nz <= (e1 and not(e2) and nsi) or (d1 and not(d2) and not(nsi));
oz <= ((e1 and not(e2) and nsi) or
       (d1 and not(d2) and not(nsi)) and os) or
       ((c1 and not(c2) and nsi) or
       (b1 and not(b2) and not(nsi)) and not(os));
```

6.11. PAFRAG_Q

pASCI Macro Cell "Q" Fragment.




```
ENTITY pafrag_q IS
    PORT(qs: IN BIT ;
         qc: IN BIT ;
         qr: IN BIT ;
         qd: IN BIT ;
         qz: OUT BIT );
END pafrag _q;
```

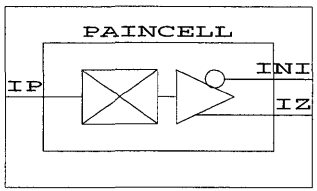
DESCRIPTION

The PAFRAG_Q implements the "Q" fragment of the pASIC macro cell. It is implemented with logic elements in other devices. The logic equivalent is:

U1: dsrff port map(s => qs, clk => qc, r => qr, d => qd, q => qz);

6.12. PAINCELL

pASIC Input Master Cell.




```
ENTITY paincell IS
    PORT(ip: IN BIT ;
         ini: OUT BIT ;
         iz: OUT BIT );
END paincell ;
```

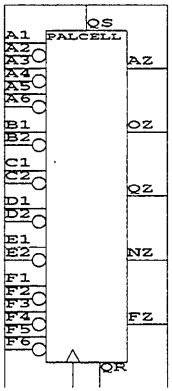
DESCRIPTION

INPUT	OUTPUTS	
IP	**INI**	**IZ**
H	L	H
L	H	L

The PAINCELL implements the input pad for pASIC devices. It is implemented as a buffer and an inverter for other devices.

6.13. PALCELL

pASIC380 FPGA internal logic cell.




```
ENTITY palcell IS
    PORT(qs: IN BIT;
         a1,a2,a3,a4,a5,a6: IN BIT;
         b1,b2: IN BIT;
         c1,c2: IN BIT;
         d1,d2: IN BIT;
         e1,e2: IN BIT;
         f1,f2,f3,f4,f5,f6: IN BIT;
         qc,qr: IN BIT;
         az,oz,qz,nz,fz: INOUT BIT);
END palcell;
```

DESCRIPTION

AZ < = A1 AND NOT (A2) AND A3 AND NOT (A4) AND A5 AND NOT (A6);

OZ <= (NZ AND AZ) OR ((C1 AND NOT (C2) AND FZ) OR (B1 AND NOT (B2) AND NOT (FZ)) AND NOT (AZ));

NZ <= (EL AND NOT (E2) AND FZ) OR (D1 AND NOT (D2) AND NOT (FZ));

FZ <= F1 AND NOT (F2) AND F3 AND NOT (F4) AND F5 AND NOT (F6);

The output 'qz' is from the internal DFF, for which 'qc' is the clock, 'oz' is the D-input, 'qs' is the present, and 'qr' is the clear.

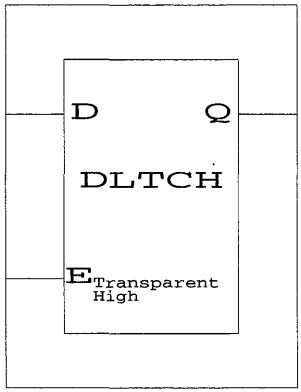
Chapter 7 Memory

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediately above the entity and architecture declarations:

```
use work.memorypkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

7.1. DFF

D-Type Edge Triggered Flip-Flop.



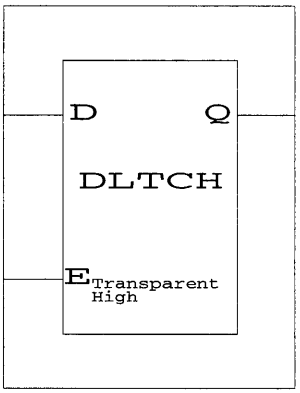

```
ENTITY dff IS
    PORT(d: IN BIT ;
         clk: IN BIT ;
         q: OUT BIT );
END dff ;
```

DESCRIPTION

INPUTS		OUTPUTS
D	**CLK**	**Q**
L	Λ	L
H	Λ	H

7.2. DLTCH

D-Type Latch.



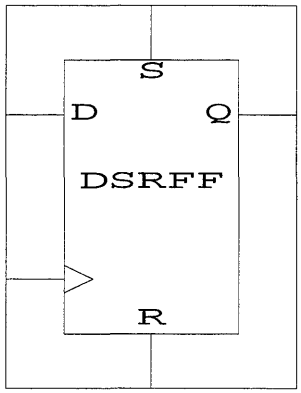

```
ENTITY dltch IS
    PORT(d: IN BIT ;
         e: IN BIT ;
         q: OUT BIT );
END dltch ;
```

DESCRIPTION

INPUTS		OUTPUT
D	**E**	**Q**
X	L	Q_0
L	H	L
H	H	H

7.3. DSRFF

D-Type Edge Triggered Flip-Flop with Set and Reset.



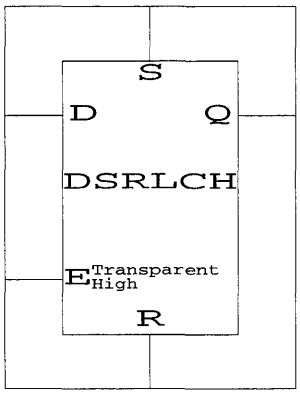

```
ENTITY dsrff IS
    PORT(d: IN BIT ;
         s: IN BIT ;
         r: IN BIT ;
         clk: IN BIT ;
         q: OUT BIT );
END dsrff ;
```

DESCRIPTION

INPUTS				OUTPUTS
D	**S**	**R**	**CLK**	**Q**
X	X	H	X	L
X	H	L	X	H
L	L	L	Λ	L
H	L	L	Λ	H

7.4. DSRLCH

D-Type Latch with Set and Reset.



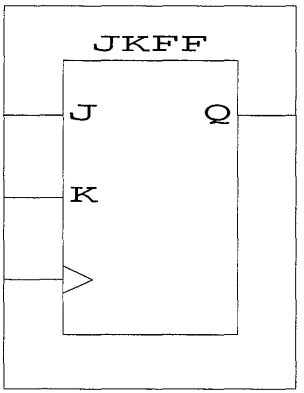

```
ENTITY dsrlch IS
    PORT(d: IN BIT ;
         s: IN BIT ;
         r: IN BIT ;
         e: IN BIT ;
         q: INOUT BIT );
END dsrlch ;
```

DESCRIPTION

INPUTS				OUTPUT
D	**S**	**R**	**E**	**Q**
X	X	H	X	L
X	H	L	X	H
X	L	L	L	Q_0
L	L	L	H	L
H	L	L	H	H

7.5. JKFF

J-K-Type Edge Triggered Flip-Flop.



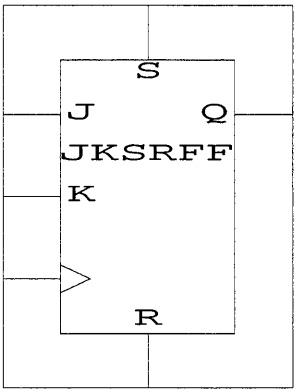

```
ENTITY jkff IS
    PORT(j: IN BIT ;
         k: IN BIT ;
         clk: IN BIT ;
         q: OUT BIT );
END jkff ;
```

DESCRIPTION

INPUTS			OUTPUTS
J	**K**	**CLK**	**Q**
L	L	Λ	Q_0
L	H	Λ	L
H	L	Λ	H
H	H	Λ	$\overline{Q}_0$

7.6. JKSRFF

J-K-Type Edge Triggered Flip-Flop with Set and Reset.



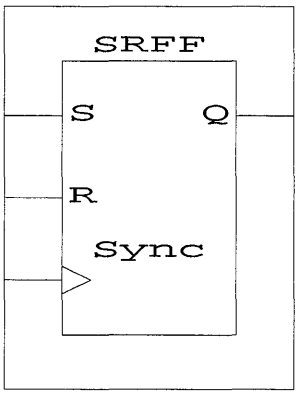

```
ENTITY jksrff IS
    PORT(j: IN BIT ;
         k: IN BIT ;
         s: IN BIT ;
         r: IN BIT ;
         clk: IN BIT ;
         q: INOUT BIT );
         attribute dont_touch of jksrff:
ENTITY IS TRUE ;
END jksrff ;
```

DESCRIPTION

INPUTS					OUTPUTS
J	**K**	**S**	**R**	**CLK**	**Q**
X	X	X	H	X	L
X	X	H	L	X	H
L	L	L	L	Λ	Q_0
L	H	L	L	Λ	L
H	L	L	L	Λ	H
H	H	L	L	Λ	$\overline{Q}_0$

7.7. SRFF

S-R-Type Edge-Triggered Flip-Flop.



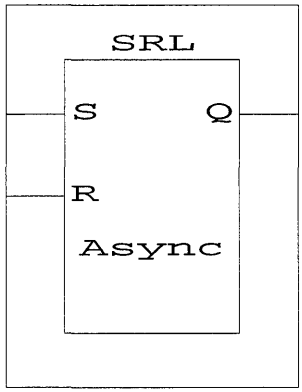

```
ENTITY srff IS
    PORT(s: IN BIT ;
         r: IN BIT ;
         clk: IN BIT ;
         q: OUT BIT );
END srff ;
```

DESCRIPTION

INPUTS			OUTPUTS
S	**R**	**CLK**	**Q**
L	L	Λ	Q_0
L	H	Λ	L
H	L	Λ	H
H	H	Λ	L

7.8. SRL

Set-Reset Latch.



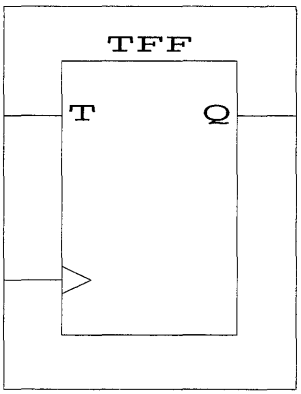

```
ENTITY srl IS
    PORT(s: IN BIT ;
         r: IN BIT ;
         q: OUT BIT );
END srl ;
```

DESCRIPTION

INPUTS		OUTPUT
S	**R**	**Q**
H	L	H
L	L	Q_0
X	H	L

7.9. TFF

T-Type Edge Triggered Flip-Flop.



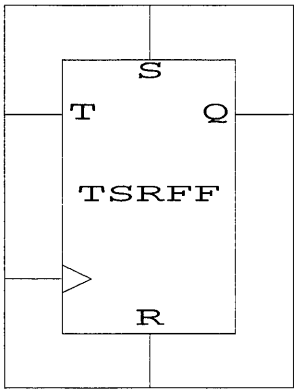

```
ENTITY tff IS
    PORT(t: IN BIT ;
         clk: IN BIT ;
         q: OUT BIT );
END tff ;
```

DESCRIPTION

INPUTS		OUTPUT
T	**CLK**	**Q**
L	Λ	Q_0
H	Λ	$\overline{Q}_0$

7.10. TSRFF

T-Type Edge Triggered Flip-Flop with Set and Reset.



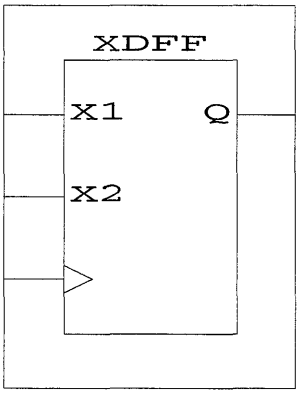

```
ENTITY tsrff IS
    PORT(t: IN BIT ;
         s: IN BIT ;
         r: IN BIT ;
         clk: IN BIT ;
         q: INOUT BIT );
END tsrff ;
```

DESCRIPTION

INPUTS				OUTPUTS
T	**S**	**R**	**CLK**	**Q**
X	X	H	X	L
X	H	L	X	H
L	L	L	Λ	Q_0
H	L	L	Λ	$\overline{Q}_0$

7.11. XDFF

D-Type Edge Triggered Flip-Flop with XOR Inputs.



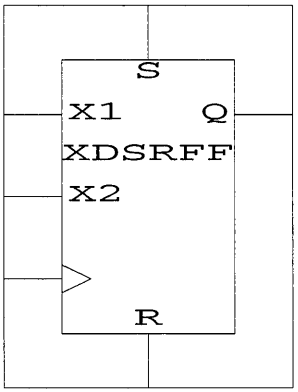

```
ENTITY xdff IS
    PORT(x1: IN BIT ;
         x2: IN BIT ;
         clk: IN BIT ;
         q: OUT BIT );
END xdff ;
```

DESCRIPTION

INPUTS			OUTPUT
X1	X2	CLK	Q
L	L	Λ	L
L	H	Λ	H
H	L	Λ	H
H	H	Λ	L

7.12. XDSRFF

D-Type Edge Triggered Flip-Flop with XOR Inputs and Set and Reset.




```
ENTITY xdsrff IS
    PORT(x1: IN BIT ;
         x2: IN BIT ;
         s: IN BIT ;
         r: IN BIT ;
         clk: IN BIT ;
         q: OUT BIT );
END xdsrff ;
```

DESCRIPTION

INPUTS					OUTPUT
S	**R**	**X1**	**X2**	**CLK**	**Q**
X	H	X	X	X	L
H	L	X	X	X	H
L	L	L	L	Λ	L
L	L	L	H	Λ	H
L	L	H	L	Λ	H
L	L	H	H	Λ	L

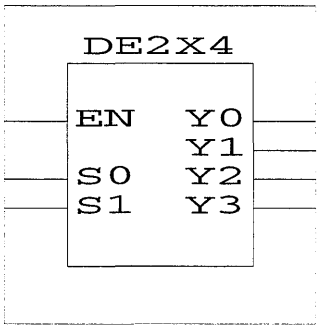
Chapter 8 Multiplexers

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediately above the entity and architecture declarations:

```
use work.muxpkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

8.1. DE2X4

2-Line to 4-Line Decoder/Demultiplexer.



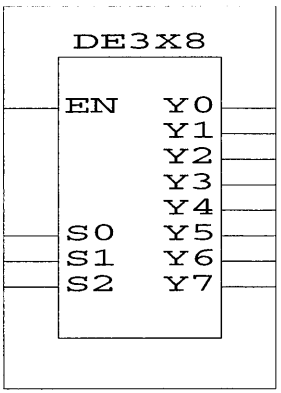

```
ENTITY de2x4 IS
    PORT(en: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         y0: OUT BIT ;
         y1: OUT BIT ;
         y2: OUT BIT ;
         y3: OUT BIT );
END de2x4 ;
```

DESCRIPTION

INPUTS			OUTPUTS			
EN	**S1**	**S0**	**Y3**	**Y2**	**Y1**	**Y0**
L	X	X	L	L	L	L
H	L	L	L	L	L	H
H	L	H	L	L	H	L
H	H	L	L	H	L	L
H	H	H	H	L	L	L

8.2. DE3X8

3-Line to 8-Line Decoder/Demultiplexer.



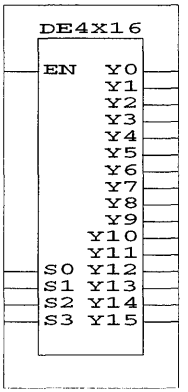

```
ENTITY de3x8 IS
    PORT(en: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         s2: IN BIT ;
         y0: OUT BIT ;
         y1: OUT BIT ;
         y2: OUT BIT ;
         y3: OUT BIT ;
         y4: OUT BIT ;
         y5: OUT BIT ;
         y6: OUT BIT ;
         y7: OUT BIT );
END de3x8 ;
```

DESCRIPTION

INPUTS				OUTPUTS							
EN	**S2**	**S1**	**S0**	**Y7**	**Y6**	**Y5**	**Y4**	**Y3**	**Y2**	**Y1**	**Y0**
L	X	X	X	L	L	L	L	L	L	L	L
H	L	L	L	L	L	L	L	L	L	L	H
H	L	L	H	L	L	L	L	L	L	H	L
H	L	H	L	L	L	L	L	L	H	L	L
H	L	H	H	L	L	L	L	H	L	L	L
H	H	L	L	L	L	L	H	L	L	L	L
H	H	L	H	L	L	H	L	L	L	L	L
H	H	H	L	L	H	L	L	L	L	L	L
H	H	H	H	H	L	L	L	L	L	L	L

8.3. DE4X16

4-Line to 16-Line Decoder/Demultiplexer.




```
ENTITY de4x16 IS
    PORT(en: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         s2: IN BIT ;
         s3: IN BIT ;
         y0,y1,y2,y3,y4,y5,y6,y7,y8,y9,
         y10,y11,y12,y13,y14,y15: OUT BIT);
END de4x16 ;
```

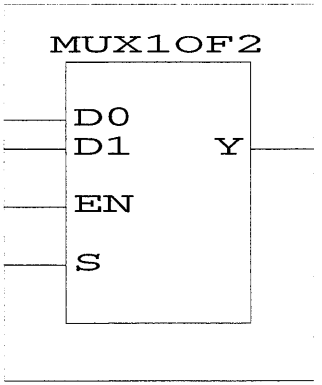
DESCRIPTION

INPUTS					OUTPUTS														
EN	**S3**	**S2**	**S1**	**S0**	**Y15**	**Y14**	**Y13**	**Y12**	**Y11**	**Y10**	**Y9**	**Y8**	**Y7**	**Y6**	**Y5**	**Y4**	**Y3**	**Y2**	**Y1**	**Y0**
L	X	X	X	X	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H
H	L	L	L	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L
H	L	L	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L	L
H	L	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	H	L	L	L
H	L	H	L	L	L	L	L	L	L	L	L	L	L	L	L	H	L	L	L	L
H	L	H	L	H	L	L	L	L	L	L	L	L	L	L	H	L	L	L	L	L
H	L	H	H	L	L	L	L	L	L	L	L	L	L	H	L	L	L	L	L	L

INPUTS					OUTPUTS															
EN	S3	S2	S1	S0	Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
H	L	H	H	H	L	L	L	L	L	L	L	L	H	L	L	L	L	L	L	L
H	H	L	L	L	L	L	L	L	L	L	L	H	L	L	L	L	L	L	L	L
H	H	L	L	H	L	L	L	L	L	L	H	L	L	L	L	L	L	L	L	L
H	H	L	H	L	L	L	L	L	L	H	L	L	L	L	L	L	L	L	L	L
H	H	L	H	H	L	L	L	L	H	L	L	L	L	L	L	L	L	L	L	L
H	H	H	L	L	L	L	L	H	L	L	L	L	L	L	L	L	L	L	L	L
H	H	H	L	H	L	L	H	L	L	L	L	L	L	L	L	L	L	L	L	L
H	H	H	H	L	L	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L
H	H	H	H	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L

8.4. MUX1OF2

2-Line to 1-Line Encoder/Multiplexer.



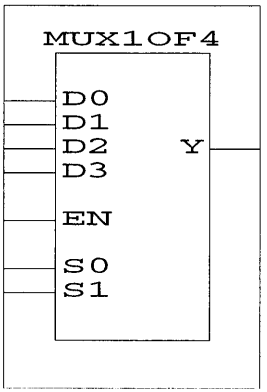

```
ENTITY mux1of2 IS
    PORT(en: IN BIT ;
         s: IN BIT ;
         d0: IN BIT ;
         d1: IN BIT ;
         y: OUT BIT );
END mux1of2 ;
```

DESCRIPTION

INPUTS		OUTPUT
EN	S	Y
L	X	L
H	L	D0
H	H	D1

8.5. MUX1OF4

4-Line to 1-Line Encoder/Multiplexer.



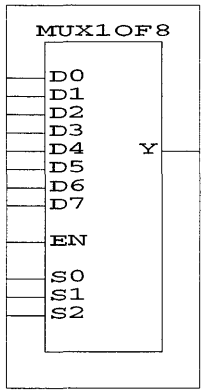

```
ENTITY mux1of4 IS
    PORT(en: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         d0: IN BIT ;
         d1: IN BIT ;
         d2: IN BIT ;
         d3: IN BIT ;
         y: OUT BIT );
END mux1of4 ;
```

DESCRIPTION

INPUTS			OUTPUT
EN	**S1**	**S0**	**Y**
L	X	X	L
H	L	L	D0
H	L	H	D1
H	H	L	D2
H	H	H	D3

8.6. MUX1OF8

8-Line to 1-Line Encoder/Multiplexer.



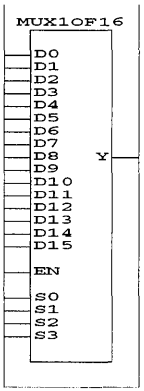

```
ENTITY mux1of8 IS
    PORT(en: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         s2: IN BIT ;
         d0,d1,d2,d3,d4,d5,d6,d7: IN BIT ;
         y: OUT BIT );
END mux1of8 ;
```

DESCRIPTION

INPUTS				OUTPUT
EN	**S2**	**S1**	**S0**	**Y**
L	X	X	X	L
H	L	L	L	D0
H	L	L	H	D1
H	L	H	L	D2
H	L	H	H	D3
H	H	L	L	D4
H	H	L	H	D5
H	H	H	L	D6
H	H	H	H	D7

8.7. MUX1OF16

16-Line to 1-Line Encoder/Multiplexer.




```
ENTITY mux1of16 IS
    PORT(en: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         s2: IN BIT ;
         s3: IN BIT ;
         d0,d1,d2,d3,d4,d5,d6,d7,d8,d9
         d10,d11,d12,d13,d14,d15: IN BIT ;
         y: OUT BIT );
END mux1of16 ;
```

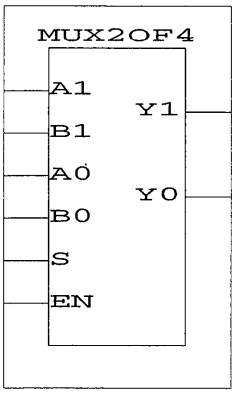
DESCRIPTION

INPUTS					OUTPUT
EN	**S3**	**S2**	**S1**	**S0**	**Y**
L	X	X	X	X	L
H	L	L	L	L	D0
H	L	L	L	H	D1
H	L	L	H	L	D2
H	L	L	H	H	D3
H	L	H	L	L	D4
H	L	H	L	H	D5
H	L	H	H	L	D6
H	L	H	H	H	D7

INPUTS					OUTPUT
EN	**S3**	**S2**	**S1**	**S0**	**Y**
H	H	L	L	L	D8
H	H	L	L	H	D9
H	H	L	H	L	D10
H	H	L	H	H	D11
H	H	H	L	L	D12
H	H	H	L	H	D13
H	H	H	H	L	D14
H	H	H	H	H	D15

8.8. MUX2OF4

Dual 2-Line to 1-Line Encoder/Multiplexer.




```
ENTITY mux2of4 IS
    PORT(en: IN BIT ;
         s: IN BIT ;
         a1: IN BIT ;
         a0: IN BIT ;
         b1: IN BIT ;
         b0: IN BIT ;
         y1: OUT BIT ;
         y0: OUT BIT );
END mux2of4 ;
```

DESCRIPTION

INPUTS		OUTPUT	
EN	**S**	**Y1**	**Y0**
L	X	L	L
H	L	A1	A0
H	H	B1	B0

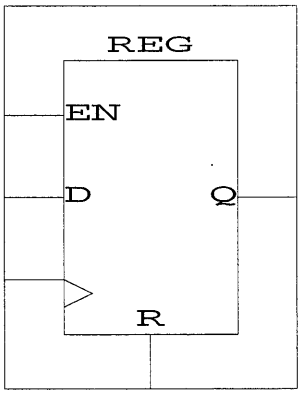
Chapter 9 Registers

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediatley above the entity and architecture declarations:

```
use work.registerpkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

9.1. REG

1-Bit D-Type Register with Clear.



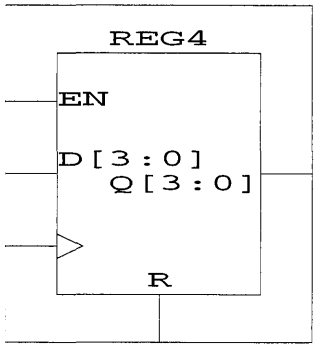

```
ENTITY reg IS
    PORT(d: IN BIT ;
         en: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q: INOUT BIT );
END reg ;
```

DESCRIPTION

INPUTS			OUTPUT
EN	**CLK**	**R**	**Q**
X	X	H	L
L	Λ	L	Q_0
H	Λ	L	D

9.2. REG4

4-Bit D-Type Register with Clear.



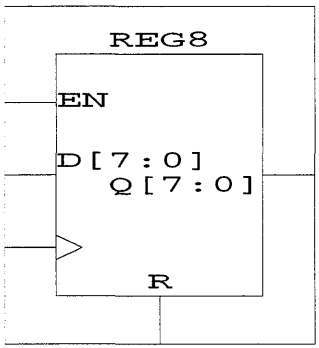

```
ENTITY reg4 IS
    PORT(d3: IN BIT ;
         d2: IN BIT ;
         d1: IN BIT ;
         d0: IN BIT ;
         en: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q3: INOUT BIT ;
         q2: INOUT BIT ;
         q1: INOUT BIT ;
         q0: INOUT BIT );
END reg4 ;
```

DESCRIPTION

INPUTS			OUTPUT
EN	**CLK**	**R**	**Q(i)**
X	X	H	L
L	Λ	L	$Q(i)_0$
H	Λ	L	D(i)

9.3. REG8

8-Bit D-type Register with Clear.




```
ENTITY reg8 IS
    PORT(d7,d6,d5,d4,d3,d2,d1,d0: IN BIT;
         en: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q7,q6,q5,q4,q3,q2,q1,q0: OUT BIT) ;
END reg8 ;
```

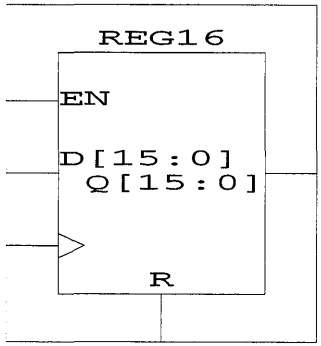
DESCRIPTION

INPUTS			OUTPUT
EN	**CLK**	**R**	**Q(i)**
X	X	H	L
L	Λ	L	$Q(i)_0$
H	Λ	L	D(i)

NOTE: This library element is only available through schematic entry.

9.4. REG16

16-Bit D-Type Register with Clear.




```
ENTITY reg16 IS
    PORT(d15,d14,d13,d12,d11,d10,d9,d8
         d7,d6,d5,d4,d3,d2,d1: IN BIT;
         en: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q15,q14,q13,q12,q11,q10,q9,q8
         q7,q6,q5,q4,q3,q2,q1,q0: OUT BIT) ;
END reg16 ;
```

DESCRIPTION

INPUTS			OUTPUT
EN	**CLK**	**R**	**Q(i)**
X	X	H	L
L	Λ	L	$Q(i)_0$
H	Λ	L	D(i)

NOTE: This library element is only available through schematic entry.

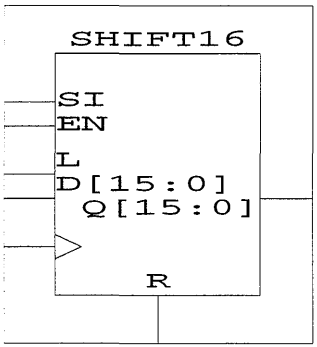
Chapter 10 Shifters

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediately above the entity and architecture declarations:

```
use work.shifterpkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

10.1. SHIFT

1-Bit Cascadable Shift Register.



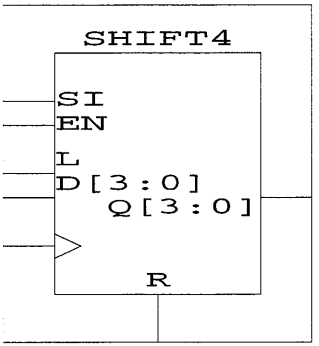

```
ENTITY shift IS
    PORT(si: IN BIT ;
         l: IN BIT ;
         d: IN BIT ;
         en: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q: OUT BIT );
END shift ;
```

DESCRIPTION

INPUTS				OUTPUT
L	**EN**	**CLK**	**R**	**Q**
X	X	X	H	L
H	X	Λ	L	D
L	L	Λ	L	Q_0
L	H	Λ	L	SI

10.2. SHIFT4

4-Bit Cascadable Shift Register.



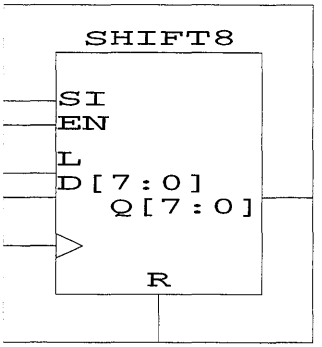

```
ENTITY shift4 IS
    PORT(si: IN BIT ;
         l: IN BIT ;
         d3,d2,d1,d0: IN BIT ;
         en: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q3,q2,q1,q0: OUT BIT);
END shift4 ;
```

DESCRIPTION

INPUTS				OUTPUT
L	**EN**	**CLK**	**R**	**Q**
X	X	X	H	0
H	X	Λ	L	D
L	L	Λ	L	Q_0
L	H	Λ	L	Q0=SI Q1=$Q0_0$ Q2=$Q1_0$ Q3=$Q2_0$

10.3. SHIFT8

8-Bit Cascadable Shift Register.



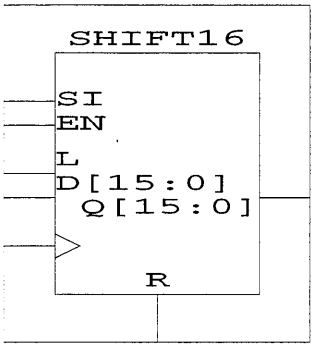

```
ENTITY shift8 IS
    PORT(si: IN BIT ;
         l: IN BIT ;
         d7.d6,d5,d4,d3,d2,d1,d0: IN BIT;
         en: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q7,q6,q5,q4,q3,q2,q1,q0: OUT BIT) ;
END shift8 ;
```

DESCRIPTION

INPUTS				OUTPUT
L	**EN**	**CLK**	**R**	**Q**
X	X	X	H	00
H	X	Λ	L	D
L	L	Λ	L	Q_0
L	H	Λ	L	Q0=SI Q4=$Q3_0$ Q1=$Q0_0$ Q5=$Q4_0$ Q2=$Q1_0$ Q6=$Q5_0$ Q3=$Q2_0$ Q7=$Q6_0$

10.4. SHIFT16

16-Bit Cascadable Shift Register.



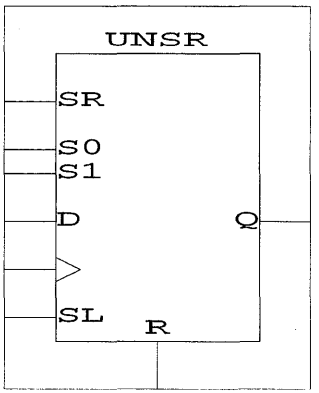

```
ENTITY shift16 IS
    PORT(si: IN BIT ;
         l: IN BIT ;
         d15,d14,d13,d12,d11,d10,d9,d8,
         d7,d6,d5,d4,d3,d2,d1,d0: IN BIT;
         en: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q15,q14,q13,q12,q11,q10,q9,q8,
         q7,q6,q5,q4,q3,q2,q1,q0: OUT BIT) ;
END shift16 ;
```

DESCRIPTION

INPUTS				OUTPUT
L	**EN**	**CLK**	**R**	**Q**
X	X	X	H	0000
H	X	Λ	L	D
L	L	Λ	L	Q_0
L	H	Λ	L	Q0=SI Q1=$Q0_0$ Q2=$Q1_0$ . . Q1=$Q0_0$ Q14=$Q13_0$

10.5. UNSR

1-Bit Cascadable Universal Shift Register.



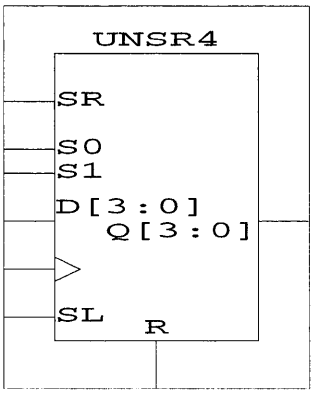

```
ENTITY unsr IS
    PORT(sl: IN BIT ;
         sr: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q: INOUT BIT );
END unsr ;
```

DESCRIPTION

INPUTS				OUTPUT
S1	**S0**	**CLK**	**R**	**Q**
X	X	X	H	0000
H	H	Λ	L	D
L	L	Λ	L	Q_0
L	H	Λ	L	Q=SR
H	L	Λ	L	Q=SL

10.6. UNSR4

4-Bit Cascadable Universal Shift Register.



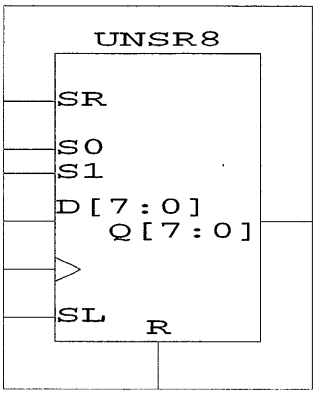

```
ENTITY unsr4 IS
    PORT(sl: IN BIT ;
         sr: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         d3,d2,d1,d0: IN BIT ;
         clk: IN BIT ;
         r: IN BIT ;
         q3,q2,q1,q0: INOUT BIT );
END unsr4 ;
```

DESCRIPTION

INPUTS				OUTPUT
S1	**S0**	**CLK**	**R**	**Q**
X	X	X	H	0
H	H	Λ	L	D
L	L	Λ	L	Q_0
L	H	Λ	L	Q3=SR Q2=$Q3_0$ Q1=$Q2_0$ Q0=$Q1_0$
H	L	Λ	L	Q0=SL Q1=$Q0_0$ Q2=$Q1_0$ Q3=$Q2_0$

10.7. UNSR8

8-Bit Cascadable Universal Shift Register.



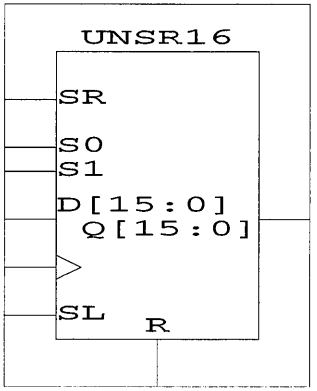

```
ENTITY unsr8 IS
    PORT (sl: IN BIT ;
          sr: IN BIT ;
          s0: IN BIT ;
          s1: IN BIT ;
          d7,d6,d5,d4,d3,d2,d1,d0: IN BIT;
          clk: IN BIT ;
          r: IN BIT ;
          q7,q6,q5,q4,q3,q2,q1,q0: OUT BIT) ;
END unsr8 ;
```

DESCRIPTION

INPUTS				OUTPUT
S1	**S0**	**CLK**	**R**	**Q**
X	X	X	H	0000
H	H	Λ	L	D
L	L	Λ	L	Q_0
L	H	Λ	L	Q7=SR Q3=$Q2_0$ Q6=$Q5_0$ Q2=$Q1_0$ Q5=$Q4_0$ Q1=$Q2_0$ Q4=$Q3_0$ Q0=$Q1_0$
H	L	Λ	L	Q0=SL Q4=$Q3_0$ Q1=$Q0_0$ Q5=$Q4_0$ Q2=$Q1_0$ Q6=$Q5_0$ Q3=$Q2_0$ Q7=$Q6_0$

10.8. UNSR16

16-Bit Cascadable Universal Shift Register.




```
ENTITY unsr16 IS
    PORT (sl: IN BIT ;
          sr: IN BIT ;
          s0: IN BIT ;
          s1: IN BIT ;
          d15,d14,d13,d12,d11,d10,d9,d8
          d7,d6,d5,d4,d3,d2,d1,d0: IN BIT;
          clk: IN BIT ;
          r: IN BIT ;
          q15,q14,q13,q12,q11,q10,q9,q8
          q7,q6,q5,q4,q3,q2,q1,q0: OUT BIT) ;
END unsr16 ;
```

DESCRIPTION

INPUTS				OUTPUT
S1	**S0**	**CLK**	**R**	**Q**
X	X	X	H	0000
H	H	Λ	L	D
L	L	Λ	L	Q_0
L	H	Λ	L	Q15=SR Q14=$Q15_0$ Q13=$Q14_0$ . . . Q1=$Q2_0$ Q0=$Q1_0$
H	L	Λ	L	Q0=SL Q1=$Q0_0$ Q2=$Q1_0$ . . . Q14=$Q13_0$ Q15=$Q14_0$

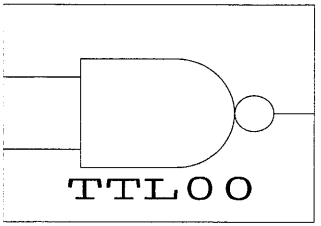
Chapter 11 TTL Parts

To use any of the components described in this chapter within a VHDL description, include the following lines in your VHDL file, immediately above the entity and architecture declarations:

```
use work.ttlpkg.all;
use work.rtlpkg.all;
use work.cypress.all;
```

11.1. TTL00

2-Input Positive-NAND Gate.




```
ENTITY ttl00 IS
    PORT(qn: OUT BIT ;
         a: IN BIT ;
         b: IN BIT );
END ttl00 ;
```

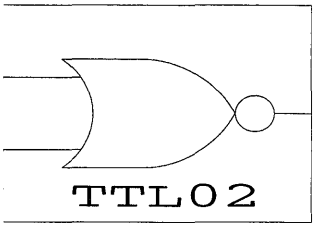
DESCRIPTION

INPUTS		OUTPUT
A	B	QN
H	H	L
L	X	H
X	L	H

NOTE: This library element is only available through schematic entry.

11.2. TTL02

2-Input Positive-NOR Gate.




```
ENTITY ttl02 IS
    PORT(qn: OUT BIT ;
         a: IN BIT ;
         b: IN BIT );
END ttl02 ;
```

DESCRIPTION

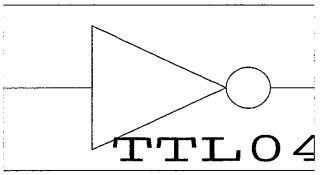
INPUTS		OUTPUT
A	**B**	**QN**
H	X	L
X	H	L
L	L	H

NOTE: This library element is only available through schematic entry.

11.3. TTL04

Inverter.

```
ENTITY ttl04 IS
    PORT(qn: OUT BIT ;
         a: IN BIT );
END ttl04 ;
```



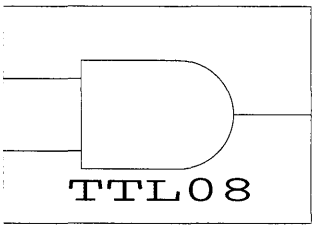
DESCRIPTION

INPUT	OUTPUT
A	QN
H	L
L	H

NOTE: This library element is only available through schematic entry.

11.4. TTL08

2-Input Positive-AND gate.




```
ENTITY ttl08 IS
    PORT(q: OUT BIT ;
         a: IN BIT ;
         b: IN BIT );
END ttl08 ;
```

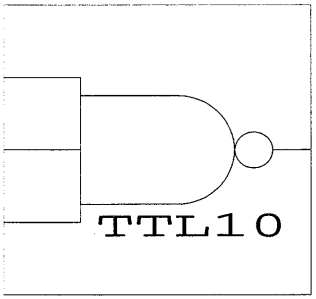
DESCRIPTION

INPUTS		OUTPUT
A	B	Q
H	H	H
L	X	L
X	L	L

NOTE: This library element is only available through schematic entry.

11.5. TTL10

3-Input Positive-NAND Gate.




```
ENTITY ttl10 IS
    PORT(qn: OUT BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT );
END ttl10 ;
```

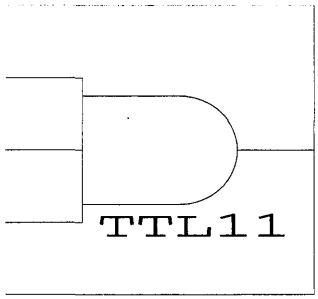
DESCRIPTION

INPUTS			OUTPUT
A	B	C	QN
H	H	H	L
L	X	X	H
X	L	X	H
X	X	L	H

NOTE: This library element is only available through schematic entry.

11.6. TTL11

3-Input Positive-AND Gate.




```
ENTITY ttl11 IS
    PORT(q: OUT BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT );
END ttl11 ;
```

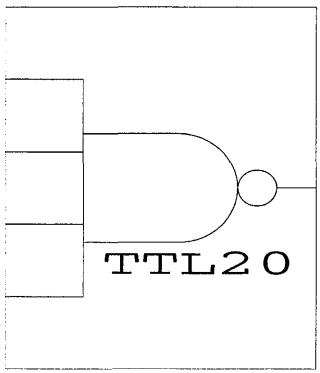
DESCRIPTION

INPUTS			OUTPUT
A	B	C	Q
H	H	H	H
L	X	X	L
X	L	X	L
X	X	L	L

NOTE: This library element is only available through schematic entry.

11.7. TTL20

4-Input Positive-NAND Gate.




```
ENTITY ttl20 IS
    PORT(qn: OUT BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT );
END ttl20 ;
```

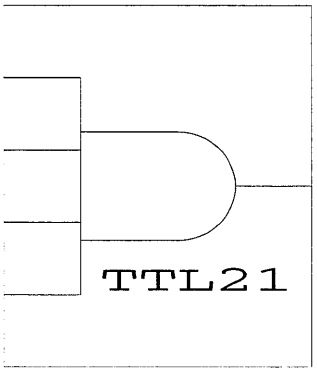
DESCRIPTION

INPUTS				OUTPUT
A	**B**	**C**	**D**	**QN**
H	H	H	H	L
L	X	X	X	H
X	L	X	X	H
X	X	L	X	H
X	X	X	L	H

NOTE: This library element is only available through schematic entry.

11.8. TTL21

4-Input Positive-AND Gate.




```
ENTITY ttl21 IS
    PORT(q: OUT BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT );
END ttl21 ;
```

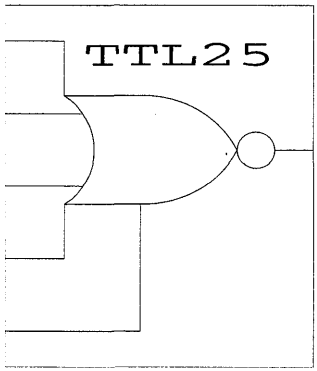
DESCRIPTION

INPUTS				OUTPUT
A	B	C	D	Q
H	H	H	H	H
L	X	X	X	L
X	L	X	X	L
X	X	L	X	L
X	X	X	L	L

NOTE: This library element is only available through schematic entry.

11.9. TTL25

4-Input Positive-NOR Gate with Strobe.




```
ENTITY ttl25 IS
    PORT(qn: OUT BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         g: IN BIT );
END ttl25 ;
```

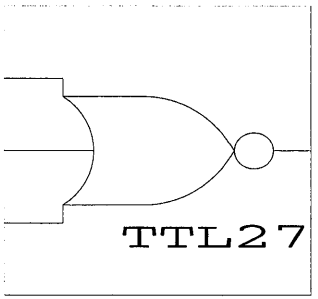
DESCRIPTION

INPUTS					OUTPUT
A	**B**	**C**	**D**	**G**	**QN**
H	X	X	X	H	L
X	H	X	X	H	L
X	X	H	X	H	L
X	X	X	H	H	L
L	L	L	L	X	H
X	X	X	X	L	H

NOTE: This library element is only available through schematic entry.

11.10. TTL27

3-Input Positive-NOR Gate.




```
ENTITY ttl27 IS
    PORT(qn: OUT BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT );
END ttl27 ;
```

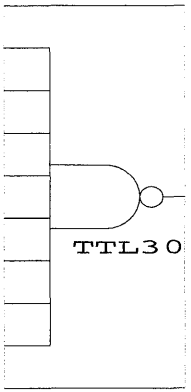
DESCRIPTION

INPUT			OUTPUT
A	**B**	**C**	**QN**
H	X	X	L
X	H	X	L
X	X	H	L
L	L	L	H

NOTE: This library element is only available through schematic entry.

11.11. TTL30

8-Input Positive-NAND Gate.




```
ENTITY ttl30 IS
    PORT (qn: OUT BIT ;
          a: IN BIT ;
          b: IN BIT ;
          c: IN BIT ;
          d: IN BIT ;
          e: IN BIT ;
          f: IN BIT ;
          g: IN BIT ;
          h: IN BIT );
END ttl30 ;
```

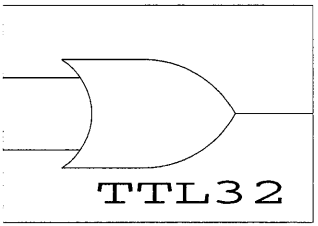
DESCRIPTION

INPUTS	OUTPUT
INPUTS A THRU H	**QN**
All inputs H	L
One or more inputs L	H

NOTE: This library element is only available through schematic entry.

11.12. TTL32

2-Input Positive-OR Gate.




```
ENTITY ttl32 IS
    PORT (q: OUT BIT ;
          a: IN BIT ;
          b: IN BIT );
END ttl32 ;
```

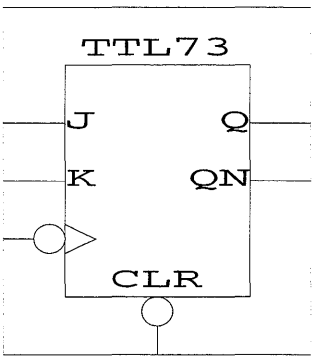
DESCRIPTION

INPUTS		OUTPUT
A	**B**	**Q**
H	X	H
X	H	H
L	L	L

NOTE: This library element is only available through schematic entry.

11.13. TTL73

J-K Flip-Flop with Clear.




```
ENTITY ttl73 IS
    PORT (j: IN BIT ;
          k: IN BIT ;
          clk: IN BIT ;
          clr: IN BIT ;
          q: OUT BIT ;
          qn: OUT BIT );
END ttl73 ;
```

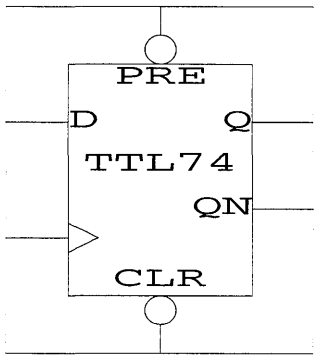
DESCRIPTION

INPUTS				OUTPUTS	
CLR	**CLK**	**J**	**K**	**Q**	**QN**
L	X	X	X	L	H
H	∀	L	L	Q_0	$\overline{Q}_0$
H	∀	H	L	H	L
H	∀	L	H	L	H
H	∀	H	H	TOGGLE	

NOTE: This library element is only available through schematic entry.

11.14. TTL74

D-Type Positive-Edge-Triggered Flip-Flop with Preset and Clear.




```
ENTITY ttl74 IS
    PORT (pre: IN BIT ;
          d: IN BIT ;
          clk: IN BIT ;
          clr: IN BIT ;
          q: OUT BIT ;
          qn: OUT BIT );
END ttl74 ;
```

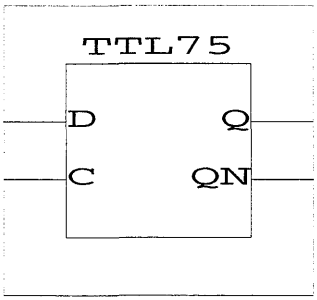
DESCRIPTION

INPUTS				OUTPUTS	
PRE	**CLR**	**CLK**	**D**	**Q**	**QN**
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H
H	H	Λ	H	H	L
H	H	Λ	L	L	H

NOTE: This library element is only available through schematic entry.

11.15. TTL75

4-Bit Bistable Latch.




```
ENTITY ttl75 IS
    PORT (d: IN BIT ;
          c: IN BIT ;
          q: OUT BIT ;
          qn: OUT BIT );
END ttl75 ;
```

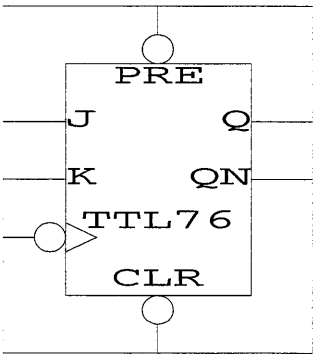
DESCRIPTION

INPUTS		OUTPUTS	
D	**C**	**Q**	**QN**
L	H	L	H
H	H	H	L
X	L	Q_0	$\overline{Q}_0$

NOTE: This library element is only available through schematic entry.

11.16. TTL76

J-K Flip-Flop with Preset and Clear.




```
ENTITY ttl76 IS
    PORT (pre: IN BIT ;
          j: IN BIT ;
          k: IN BIT ;
          clk: IN BIT ;
          clr: IN BIT ;
          q: OUT BIT ;
          qn: OUT BIT );
END ttl76 ;
```

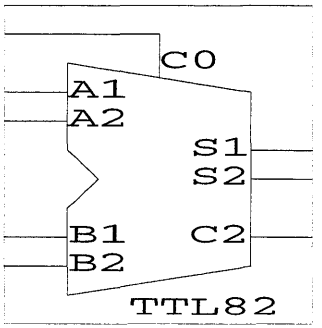
DESCRIPTION

INPUTS					OUTPUTS	
PRE	**CLR**	**CLK**	**J**	**K**	**Q**	**QN**
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H	H
H	H	∀	L	L	Q_0	$\overline{Q}_0$
H	H	∀	H	L	H	L
H	H	∀	L	H	L	H
H	H	∀	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\overline{Q}_0$

NOTE: This library element is only available through schematic entry.

11.17. TTL82

2-Bit Binary Full Adder with Fast Carry.




```
ENTITY ttl82 IS
    PORT(c0: IN BIT ;
         a1: IN BIT ;
         a2: IN BIT ;
         b1: IN BIT ;
         b2: IN BIT ;
         s1: OUT BIT ;
         s2: OUT BIT ;
         c2: OUT BIT );
END ttl82 ;
```

DESCRIPTION

S1 = A1 XOR B1 XOR C0

s2 = A2 XOR B2 XOR ((A1 AND B1) OR (A1 AND C0) OR (B1 AND C0))

C2 = (A2 AND B2) OR
(A2 AND A1 AND B1) OR (A2 AND A1 AND C0) OR (A2 AND B1 AND C0) OR
(B2 AND A1 AND B1) OR (B2 AND A1 AND C0) OR (B2 AND B1 AND C0)

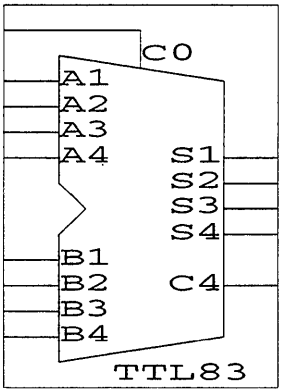
In general,

S = A + B + C0; with carry out on C2.

NOTE: This library element is only available through schematic entry.

11.18. TTL83

4-Bit Binary Full Adder with Fast Carry.



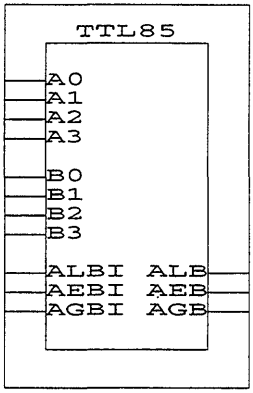

```
ENTITY ttl83 IS
    PORT(c0: IN BIT ;
         a1: IN BIT ;
         a2: IN BIT ;
         a3: IN BIT ;
         a4: IN BIT ;
         b1: IN BIT ;
         b2: IN BIT ;
         b3: IN BIT ;
         b4: IN BIT ;
         s1: OUT BIT ;
         s2: OUT BIT ;
         s3: OUT BIT ;
         s4: OUT BIT ;
         c4: OUT BIT );
END ttl83 ;
```

DESCRIPTION

S = A + B + C0; with carry out on C4.

11.19. TTL85

4-Bit Magnitude Comparator.




```
ENTITY ttl85 IS
    PORT(a0: IN BIT ;
         a1: IN BIT ;
         a2: IN BIT ;
         a3: IN BIT ;
         b0: IN BIT ;
         b1: IN BIT ;
         b2: IN BIT ;
         b3: IN BIT ;
         albi: IN BIT ;
         aebi: IN BIT ;
         agbi: IN BIT ;
         alb: OUT BIT ;
         aeb: OUT BIT ;
         agb: OUT BIT );
END ttl85 ;
```

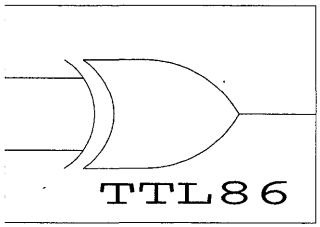
DESCRIPTION

INPUTS							OUTPUTS		
COMPARING				CASCADING					
A3,B3	A2,B2	A1,B1	A0,B0	AGBI	ALBI	AEBI	AGB	ALB	AEB
a3>b3	X	X	X	X	X	X	H	L	L
a3<b3	X	X	X	X	X	X	L	H	L
a3=b3	a2>b2	X	X	X	X	X	H	L	L
a3=b3	a2<b2	X	X	X	X	X	L	H	L
a3=b3	a2=b2	a1>b1	X	X	X	X	H	L	L

INPUTS							OUTPUTS		
COMPARING				CASCADING					
A3,B3	**A2,B2**	**A1,B1**	**A0,B0**	**AGBI**	**ALBI**	**AEBI**	**AGB**	**ALB**	**AEB**
a3=b3	a2=b2	a1<b1	X	X	X	X	L	H	L
a3=b3	a2=b2	a1=b1	a0>b0	X	X	X	H	L	L
a3=b3	a2=b2	a1=b1	a0<b0	X	X	X	L	H	L
a3=b3	a2=b2	a1=b1	a0=b0	H	L	L	H	L	L
a3=b3	a2=b2	a1=b1	a0=b0	L	H	L	L	H	L
a3=b3	a2=b2	a1=b1	a0=b0	X	X	H	L	L	H
a3=b3	a2=b2	a1=b1	a0=b0	H	H	L	L	L	L
a3=b3	a2=b2	a1=b1	a0=b0	L	L	L	H	H	L

11.20. TTL86

2-Input EXCLUSIVE-OR Gate.



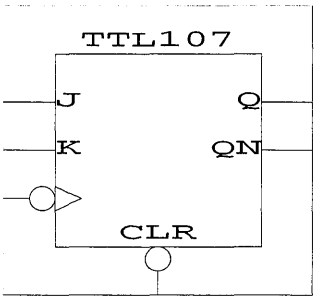

```
ENTITY ttl86 IS
    PORT (q: OUT BIT ;
          a: IN BIT ;
          b: IN BIT );
END ttl86 ;
```

DESCRIPTION

INPUTS		OUTPUT
A	**B**	**Q**
L	L	L
L	H	H
H	L	H
H	H	L

11.21. TTL107

J-K Flip-Flop with Clear.




```
ENTITY ttl107 IS
    PORT(j: IN BIT ;
         k: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         q: OUT BIT ;
         qn: OUT BIT );
END ttl107 ;
```

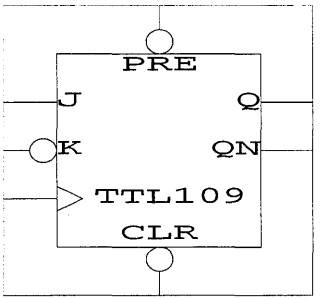
DESCRIPTION

INPUTS				OUTPUTS	
CLR	**CLK**	**J**	**K**	**Q**	**QN**
L	X	X	X	L	H
H	∀	L	L	Q_0	$\overline{Q}_0$
H	∀	H	L	H	L
H	∀	L	H	L	H
H	∀	H	H	TOGGLE	
H	H	X	X	Q_0	$\overline{Q}_0$

NOTE: This library element is only available through schematic entry.

11.22. TTL109

J-$\overline{K}$ Positive-Edge-Triggered Flip-Flop with Preset and Clear.




```
ENTITY ttl109 IS
    PORT(pre: IN BIT ;
         j: IN BIT ;
         k: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         q: OUT BIT ;
         qn: OUT BIT );
END ttl109 ;
```

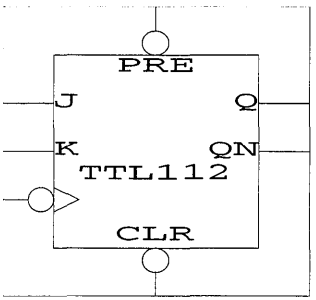
DESCRIPTION

INPUTS					OUTPUTS	
PRE	**CLR**	**CLK**	**J**	**K**	**Q**	**QN**
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H	H
H	H	Λ	L	L	L	H
H	H	Λ	H	L	TOGGLE	
H	H	Λ	L	H	Q_0	$\overline{Q}_0$
H	H	Λ	H	H	H	L
H	H	H	X	X	Q_0	$\overline{Q}_0$

NOTE: This library element is only available through schematic entry.

11.23. TTL112

J-K Negative-Edge-Triggered Flip-Flop with Preset and Clear.




```
ENTITY ttl112 IS
    PORT(pre: IN BIT ;
         j: IN BIT ;
         k: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         q: OUT BIT ;
         qn: OUT BIT );
END ttl112 ;
```

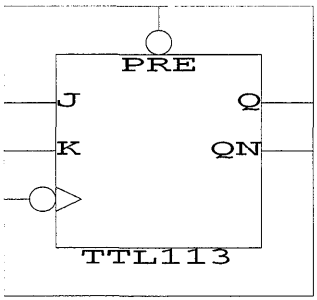
DESCRIPTION

INPUTS					OUTPUTS	
PRE	**CLR**	**CLK**	**J**	**K**	**Q**	**QN**
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H	H
H	H	∀	L	L	Q_0	$\overline{Q}_0$
H	H	∀	H	L	H	L
H	H	∀	L	H	L	H
H	H	∀	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\overline{Q}_0$

NOTE: This library element is only available through schematic entry.

11.24. TTL113

J-K Negative-Edge-Triggered Flip-Flop with Preset.




```
ENTITY ttl113 IS
    PORT(j: IN BIT ;
         k: IN BIT ;
         clk: IN BIT ;
         pre: IN BIT ;
         q: OUT BIT ;
         qn: OUT BIT );
END ttl113 ;
```

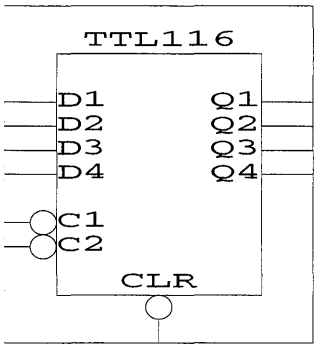
DESCRIPTION

INPUTS				OUTPUTS	
PRE	**CLK**	**J**	**K**	**Q**	**QN**
L	X	X	X	H	L
H	∀	L	L	Q_0	$\overline{Q}_0$
H	∀	H	L	H	L
H	∀	L	H	L	H
H	∀	H	H	TOGGLE	
H	H	X	X	Q_0	$\overline{Q}_0$

NOTE: This library element is only available through schematic entry.

11.25. TTL116

4-Bit Latch with Clear.




```
ENTITY ttl116 IS
    PORT(clr: IN BIT ;
         c1: IN BIT ;
         c2: IN BIT ;
         d1: IN BIT ;
         d2: IN BIT ;
         d3: IN BIT ;
         d4: IN BIT ;
         q1: OUT BIT ;
         q2: OUT BIT ;
         q3: OUT BIT ;
         q4: OUT BIT );
END ttl116 ;
```

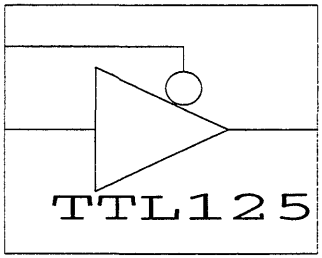
DESCRIPTION

INPUTS				OUTPUT
CLR	**C2**	**C1**	**D(i)**	**Q(i)**
H	L	L	L	L
H	L	L	H	H
H	X	H	X	$Q(i)_0$
H	H	X	X	$Q(i)_0$
L	X	X	X	L

NOTE: This library element is only available through schematic entry.

11.26. TTL125

Bus Buffer with Low Enable 3-State Output.




```
ENTITY ttl125 IS
    PORT(g: IN BIT ;
         a: IN BIT ;
         y: OUT X01Z);
END ttl125 ;
```

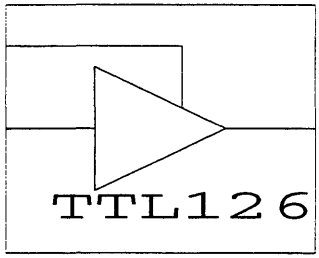
DESCRIPTION

INPUTS		OUTPUTS
G	A	Y
H	X	Z
L	L	L
L	H	H

NOTE: This library element is only available through schematic entry.

11.27. TTL126

Bus Buffer with High Enable 3-State Output.




```
ENTITY ttl126 IS
    PORT(g: IN BIT ;
         a: IN BIT ;
         y: OUT X01Z);
END ttl126 ;
```

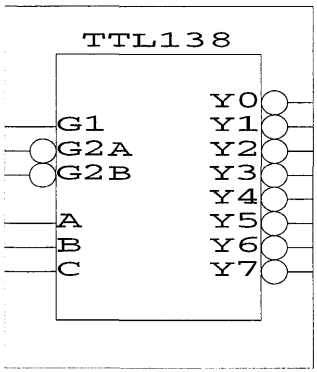
DESCRIPTION

INPUTS		OUTPUTS
G	A	Y
L	X	Z
H	L	L
H	H	H

NOTE: This library element is only available through schematic entry.

11.28. TTL138

3-Line to 8-Line Decoder/Demultiplexer.




```
ENTITY ttl138 IS
    PORT(g1: IN BIT ;
         g2a: IN BIT ;
         g2b: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         y0: OUT BIT ;
         y1: OUT BIT ;
         y2: OUT BIT ;
         y3: OUT BIT ;
         y4: OUT BIT ;
         y5: OUT BIT ;
         y6: OUT BIT ;
         y7: OUT BIT );
END ttl138 ;
```

DESCRIPTION

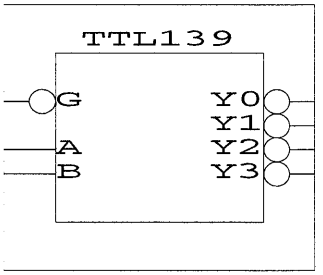
INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

* G2 = G2A + G2B

NOTE: This library element is only available through schematic entry.

11.29. TTL139

2-Line to 4-Line Decoder/Demultiplexer.




```
ENTITY ttl139 IS
    PORT(g: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         y0: OUT BIT ;
         y1: OUT BIT ;
         y2: OUT BIT ;
         y3: OUT BIT );
END ttl139 ;
```

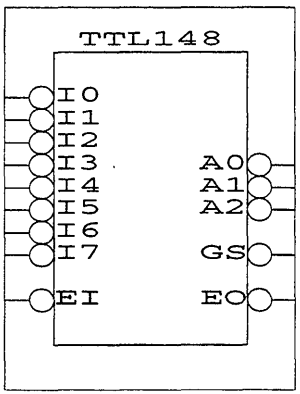
DESCRIPTION

INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

NOTE: This library element is only available through schematic entry.

11.30. TTL148

8-Line to 3-Line Priority Encoder.




```
ENTITY ttl148 IS
    PORT(ei: IN BIT ;
         i0: IN BIT ;
         i1: IN BIT ;
         i2: IN BIT ;
         i3: IN BIT ;
         i4: IN BIT ;
         i5: IN BIT ;
         i6: IN BIT ;
         i7: IN BIT ;
         a0: OUT BIT ;
         a1: OUT BIT ;
         a2: OUT BIT ;
         gs: OUT BIT ;
         eo: OUT BIT );
END ttl148 ;
```

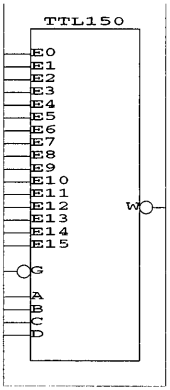
DESCRIPTION

INPUTS									OUTPUTS				
EI	**I0**	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**A2**	**A1**	**A0**	**GS**	**EO**
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H

INPUTS									OUTPUTS				
EI	**I0**	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**A2**	**A1**	**A0**	**GS**	**EO**
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

11.31. TTL150

1 of 16 Data Selector/Multiplexer.




```
ENTITY ttl150 IS
    PORT(g: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         e0: IN BIT ;
         e1: IN BIT ;
         e2: IN BIT ;
         e3: IN BIT ;
         e4: IN BIT ;
         e5: IN BIT ;
         e6: IN BIT ;
         e7: IN BIT ;
         e8: IN BIT ;
         e9: IN BIT ;
         e10: IN BIT ;
         e11: IN BIT ;
         e12: IN BIT ;
         e13: IN BIT ;
         e14: IN BIT ;
         e15: IN BIT ;
         w: OUT BIT );
END ttl150 ;
```

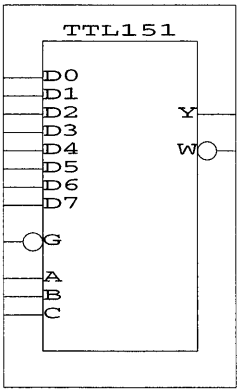
DESCRIPTION

INPUTS					OUTPUT
ENABLE	SELECT				
G	D	C	B	A	W
H	X	X	X	X	H
L	L	L	L	L	$\overline{E0}$
L	L	L	L	H	$\overline{E1}$
L	L	L	H	L	$\overline{E2}$
L	L	L	H	H	$\overline{E3}$
L	L	H	L	L	$\overline{E4}$
L	L	H	L	H	$\overline{E5}$
L	L	H	H	L	$\overline{E6}$
L	L	H	H	H	$\overline{E7}$
L	H	L	L	L	$\overline{E8}$
L	H	L	L	H	$\overline{E9}$
L	H	L	H	L	$\overline{E10}$
L	H	L	H	H	$\overline{E11}$
L	H	H	L	L	$\overline{E12}$
L	H	H	L	H	$\overline{E13}$
L	H	H	H	L	$\overline{E14}$
L	H	H	H	H	$\overline{E15}$

NOTE: This library element is only available through schematic entry.

11.32. TTL151

1 of 8 Data Selector/Multiplexer.




```
ENTITY ttl151 IS
    PORT(d0: IN BIT ;
         d1: IN BIT ;
         d2: IN BIT ;
         d3: IN BIT ;
         d4: IN BIT ;
         d5: IN BIT ;
         d6: IN BIT ;
         d7: IN BIT ;
         g: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         y: OUT BIT ;
         w: OUT BIT );
END ttl151 ;
```

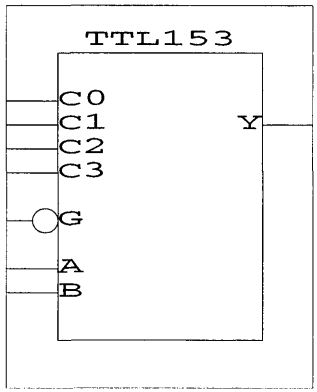
DESCRIPTION

INPUTS				OUTPUTS	
ENABLE	SELECT				
G	C	B	A	Y	W
H	X	X	X	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	L	H	D1	$\overline{D1}$
L	L	H	L	D2	$\overline{D2}$
L	L	H	H	D3	$\overline{D3}$
L	H	L	L	D4	$\overline{D4}$
L	H	L	H	D5	$\overline{D5}$
L	H	H	L	D6	$\overline{D6}$
L	H	H	H	D7	$\overline{D7}$

NOTE: This library element is only available through schematic entry.

11.33. TTL153

4-Line to 1-Line Data Selector/Multiplexer.




```
ENTITY ttl153 IS
    PORT(c0: IN BIT ;
         c1: IN BIT ;
         c2: IN BIT ;
         c3: IN BIT ;
         g: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         y: OUT BIT );
END ttl153 ;
```

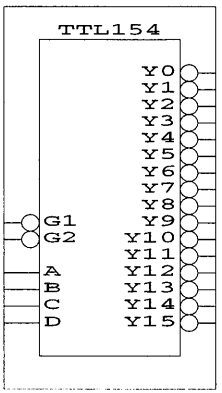
DESCRIPTION

INPUTS							OUTPUT
SELECT		DATA				STROBE	
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

NOTE: This library element is only available through schematic entry.

11.34. TTL154

4-Line to 16-Line Decoder/Demultiplexer.




```
ENTITY ttl154 IS
    PORT(g1: IN BIT ;
         g2: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         y0: OUT BIT ;
         y1: OUT BIT ;
         y2: OUT BIT ;
         y3: OUT BIT ;
         y4: OUT BIT ;
         y5: OUT BIT ;
         y6: OUT BIT ;
         y7: OUT BIT ;
         y8: OUT BIT ;
         y9: OUT BIT ;
         y10: OUT BIT ;
         y11: OUT BIT ;
         y12: OUT BIT ;
         y13: OUT BIT ;
         y14: OUT BIT ;
         y15: OUT BIT );
END ttl154 ;
```

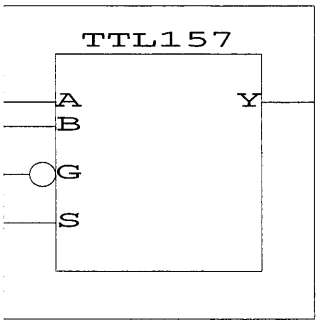
NOTE: This library element is only available through schematic entry.

DESCRIPTION

INPUTS						OUTPUTS														
ENA		SELECT																		
G1	G2	D	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12	Y13	Y14	Y15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

11.35. TTL157

2-Line to 1-Line Data Selector/Multiplexer.




```
ENTITY ttl157 IS
    PORT(a: IN BIT ;
         b: IN BIT ;
         g: IN BIT ;
         s: IN BIT ;
         y: OUT BIT );
END ttl157 ;
```

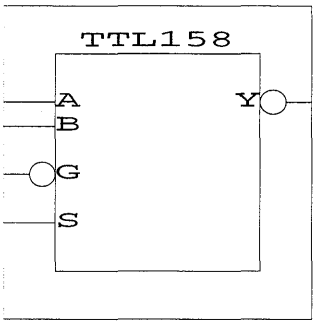
DESCRIPTION

INPUTS				OUTPUT
STROBE	SELECT	DATA		
G	S	A	B	Y
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

NOTE: This library element is only available through schematic entry.

11.36. TTL158

2-Line to 1-Line Data Selector/Multiplexer.




```
ENTITY ttl158 IS
    PORT(a: IN BIT ;
         b: IN BIT ;
         g: IN BIT ;
         s: IN BIT ;
         y: OUT BIT );
END ttl158 ;
```

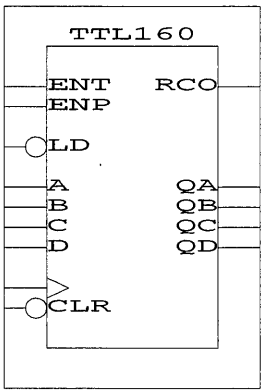
DESCRIPTION

INPUTS				OUTPUT
STROBE	SELECT	DATA		
G	S	A	B	Y
H	X	X	X	H
L	L	L	X	H
L	L	H	X	L
L	H	X	L	H
L	H	X	H	L

NOTE: This library element is only available through schematic entry.

11.37. TTL160

Synchronous 4-Bit Decade Counter with Direct Clear.




```
ENTITY ttl160 IS
    PORT(ent: IN BIT ;
         enp: IN BIT ;
         ld: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         rco: OUT BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT );
END ttl160 ;
```

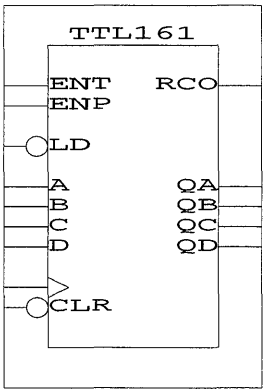
DESCRIPTION

Decade counter with cascade in and out signals. Both count enable inputs (ENT and ENP) must be high to count, and input ENT is fed forward to enable the ripple carry output (RCO). The RCO then produces a high-level output pulse with a duration approximately equal to the high-level portion of the QA output. This pulse is used to enable successive cascaded stages. The counter is loaded when the load (LD) is low during a high-going clock (CLK). The counter is cleared asynchronously when CLR is asserted low.

NOTE: This library element is only available through schematic entry.

11.38. TTL161

Synchronous 4-Bit Binary Counter with Direct Clear.




```
ENTITY ttl161 IS
    PORT(ent: IN BIT ;
         enp: IN BIT ;
         ld: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         rco: OUT BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT );
END ttl161 ;
```

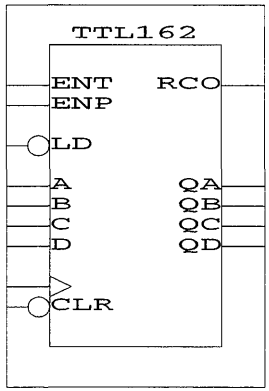
DESCRIPTION

Binary counter with cascade in and out signals. Both count enable inputs (ENT and ENP) must be high to count, and input ENT is fed forward to enable the ripple carry output (RCO). The RCO then produces a high-level output pulse with a duration approximately equal to the high-level portion of the QA output. This pulse is used to enable successive cascaded stages. The counter is loaded when the load (LD) is low during a high-going clock (CLK). The counter is cleared asynchronously when CLR is asserted low.

NOTE: This library element is only available through schematic entry.

11.39. TTL162

Fully Synchronous 4-Bit Decade Counter.




```
ENTITY ttl162 IS
    PORT(ent: IN BIT ;
         enp: IN BIT ;
         ld: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         rco: OUT BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT );
END ttl162 ;
```

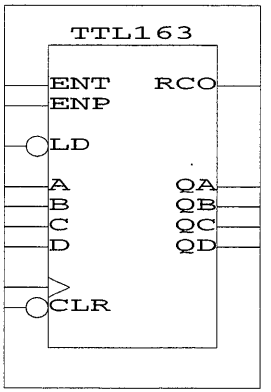
DESCRIPTION

Decade counter with cascade in and out signals. Both count enable inputs (ENT and ENP) must be high to count, and input ENT is fed forward to enable the ripple carry output (RCO). The RCO then produces a high-level output pulse with a duration approximately equal to the high-level portion of the QA output. This pulse is used to enable successive cascaded stages. The counter is loaded when the load (LD) is low during a high going clock (CLK). The counter is cleared synchronously when CLR is low during a high going CLK.

NOTE: This library element is only available through schematic entry.

11.40. TTL163

Fully Synchronous 4-Bit Binary Counter.




```
ENTITY ttl163 IS
    PORT(ent: IN BIT ;
         enp: IN BIT ;
         ld: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         rco: OUT BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT );
END ttl163 ;
```

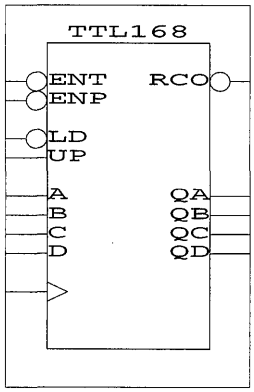
DESCRIPTION

Binary counter with cascade in and out signals. Both count enable inputs (ENT and ENP) must be high to count, and input ENT is fed forward to enable the ripple carry output (RCO). The RCO then produces a high-level output pulse with a duration approximately equal to the high-level portion of the QA output. This pulse is used to enable successive cascaded stages. The counter is loaded when the load (LD) is low during a high-going clock (CLK). The counter is cleared synchronously when CLR is low during a high going CLK.

NOTE: This library element is only available through schematic entry.

11.41. TTL168

Synchronous 4-Bit Up/Down Decade Counter.




```
ENTITY ttl168 IS
    PORT(ent: IN BIT ;
         enp: IN BIT ;
         ld: IN BIT ;
         up: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         rco: OUT BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT );
END ttl168 ;
```

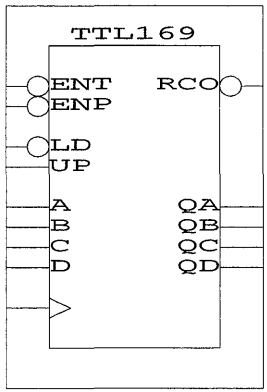
DESCRIPTION

Decade up/down counter with cascade in and out signals. Both count enable inputs (ENT and ENP) must be high to count, and input ENT is fed forward to enable the ripple carry output (RCO). The RCO then produces a high-level output pulse with a duration approximately equal to the high-level portion of the QA output. This pulse is used to enable successive cascaded stages. The counter is loaded when the load (LD) is low during a high-going clock (CLK). The counter counts up if UP is high and down if UP is low.

NOTE: This library element is only available through schematic entry.

11.42. TTL169

Synchronous 4-Bit Up/Down Binary Counter.




```
ENTITY ttl169 IS
    PORT(ent: IN BIT ;
         enp: IN BIT ;
         ld: IN BIT ;
         up: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         rco: OUT BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT );
END ttl169 ;
```

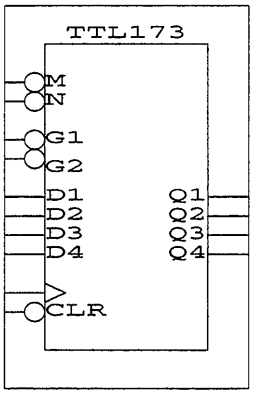
DESCRIPTION

Binary up/down counter with cascade in and out signals. Both count enable inputs (ENT and ENP) must be high to count, and input ENT is fed forward to enable the ripple carry output (RCO). The RCO then produces a high-level output pulse with a duration approximately equal to the high-level portion of the QA output. This pulse is used to enable successive cascaded stages. The counter is loaded when the load (LD) is low during a high-going clock (CLK). The counter counts up if UP is high and down if UP is low.

NOTE: This library element is only available through schematic entry.

11.43. TTL173

4-Bit D-Type register with 3-State Outputs.




```
ENTITY ttl173 IS
    PORT(m: IN BIT ;
         n: IN BIT ;
         g1: IN BIT ;
         g2: IN BIT ;
         d1: IN BIT ;
         d2: IN BIT ;
         d3: IN BIT ;
         d4: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         q1: OUT X01Z;
         q2: OUT X01Z;
         q3: OUT X01Z;
         q4: OUT X01Z);
END ttl173 ;
```

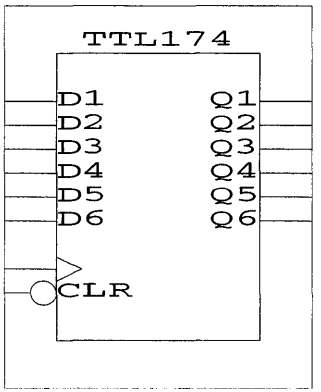
DESCRIPTION

INPUTS					OUTPUT
CLR	**CLK**	**G1**	**G2**	**D(i)**	**Q(i)**
H	X	X	X	X	L
L	L	X	X	X	Q0
L	Λ	H	X	X	Q0
L	Λ	X	H	X	Q0
L	Λ	L	L	L	L
L	Λ	L	L	H	H
When either M or N are high the output is disabled to the high impedance state; however sequential operation is not affected.					

NOTE: This library element is only available through schematic entry.

11.44. TTL174

Hex D-Type Flip-Flops with Clear.




```
ENTITY ttl174 IS
    PORT(d1: IN BIT ;
         d2: IN BIT ;
         d3: IN BIT ;
         d4: IN BIT ;
         d5: IN BIT ;
         d6: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         q1: OUT BIT ;
         q2: OUT BIT ;
         q3: OUT BIT ;
         q4: OUT BIT ;
         q5: OUT BIT ;
         q6: OUT BIT );
END ttl174 ;
```

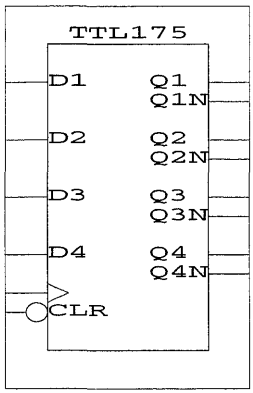
DESCRIPTION

INPUTS			OUTPUTS
CLR	**CLK**	**D(i)**	**Q(i)**
L	X	X	L
H	Λ	H	H
H	Λ	L	L
H	L	X	$Q(i)_0$

NOTE: This library element is only available through schematic entry.

11.45. TTL175

Quadruple D-Type Flip-Flops with Clear.




```
ENTITY ttl175 IS
    PORT(d1: IN BIT ;
         d2: IN BIT ;
         d3: IN BIT ;
         d4: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         q1: OUT BIT ;
         q1n: OUT BIT ;
         q2: OUT BIT ;
         q2n: OUT BIT ;
         q3: OUT BIT ;
         q3n: OUT BIT ;
         q4: OUT BIT ;
         q4n: OUT BIT );
END ttl175 ;
```

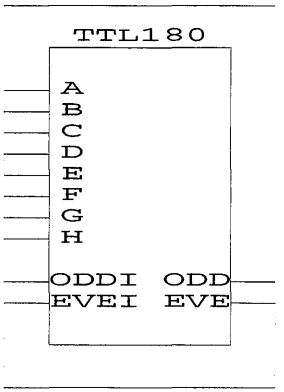
DESCRIPTION

INPUTS			OUTPUTS	
CLR	**CLK**	**D(i)**	**Q(i)**	**Q(i)N**
L	X	X	L	H
H	Λ	H	H	L
H	Λ	L	L	H
H	L	X	$Q(i)_0$	$\overline{Q}(i)_0$

NOTE: This library element is only available through schematic entry.

11.46. TTL180

9-Bit Odd/Even Parity Generator/Checker.



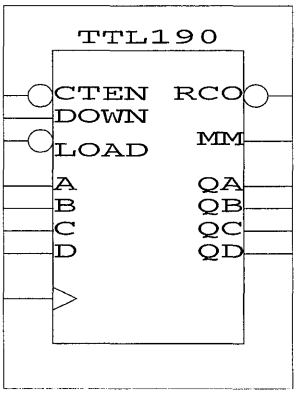

```
ENTITY ttl180 IS
    PORT(a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         e: IN BIT ;
         f: IN BIT ;
         g: IN BIT ;
         h: IN BIT ;
         oddi: IN BIT ;
         evei: IN BIT ;
         odd: OUT BIT ;
         eve: OUT BIT );
END ttl180 ;
```

DESCRIPTION

INPUTS			OUTPUTS	
Σ of H's AT A THRU H	**EVEI**	**ODDI**	**EVE**	**ODD**
EVEN	H	L	H	L
ODD	H	L	L	H
EVEN	L	H	L	H
ODD	L	H	H	L
X	H	H	L	L
X	L	L	H	H

11.47. TTL190

Synchronous Decade Up/Down Counter with Down/Up Mode Control.




```
ENTITY ttl190 IS
    PORT(cten: IN BIT ;
         down: IN BIT ;
         load: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         rco: OUT BIT ;
         mm: OUT BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT );
END ttl190 ;
```

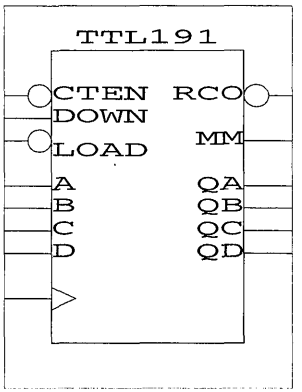
DESCRIPTION

Decade up/down counter with cascade in and out signals. CTEN must be low to count, and is fed forward to enable the ripple carry output (RCO). The RCO then produces a low-level output pulse with a duration approximately equal to the low-level portion of the clock during a min/max (MM) count. This pulse is used to enable successive cascaded stages. The counter is loaded when the LOAD is low during a high-going clock (CLK). The counter counts up if DOWN is high and down if DOWN is low. Output MM is high if the count is at 0000 when counting down or if the count is 1001 when counting up.

NOTE: This library element is only available through schematic entry.

11.48. TTL191

Synchronous Binary Up/Down Counter with Down/Up Mode Control.




```
ENTITY ttl191 IS
    PORT(cten: IN BIT ;
         down: IN BIT ;
         load: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         clk: IN BIT ;
         rco: OUT BIT ;
         mm: OUT BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT );
END ttl191 ;
```

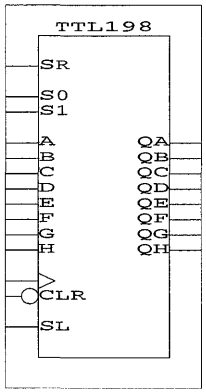
DESCRIPTION

Binary up/down counter with cascade in and out signals. CTEN must be low to count, and is fed forward to enable the ripple carry output (RCO). The RCO then produces a low-level output pulse with a duration approximately equal to the low-level portion of the clock during a min/max (MM) count. This pulse is used to enable successive cascaded stages. The counter is loaded when the LOAD is low during a high-going clock (CLK). The counter counts up if DOWN is high, and down if DOWN is low. Output MM is high if the count is at 0000 when counting down, or if the count is 1001 when counting up.

NOTE: This library element is only available through schematic entry.

11.49. TTL198

8-Bit Bi-directional Shift Register.




```
ENTITY ttl198 IS
    PORT(sr: IN BIT ;
         s0: IN BIT ;
         s1: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         e: IN BIT ;
         f: IN BIT ;
         g: IN BIT ;
         h: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         sl: IN BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT ;
         qe: OUT BIT ;
         qf: OUT BIT ;
         qg: OUT BIT ;
         qh: OUT BIT );
END ttl198 ;
```

NOTE: This library element is only available through schematic entry.

DESCRIPTION

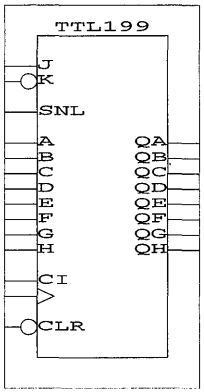
INPUTS							OUTPUTS							
CLR	**S1**	**S0**	**CLK**	**SL**	**SR**	**a...h**	**QA**	**QB**	**QC**	**QD**	**QE**	**QF**	**QG**	**QH**
L	X	X	X	X	X	X	L	L	L	L	L	L	L	L
H	X	X	L	X	X	X	QA_0	QB_0	QC_0	QD_0	QE_0	QF_0	QG_0	QH_0
H	H	H	Λ	X	X	a...h	A	B	C	D	E	F	G	H
H	L	H	Λ	X	H	X	H	QA_N	QB_N	QC_N	QD_N	QE_N	QF_N	QG_N
H	L	H	Λ	X	L	X	L	QA_N	QB_N	QC_N	QD_N	QE_N	QF_N	QG_N
H	H	L	Λ	H	X	X	QB_N	QC_N	QD_N	QE_N	QF_N	QG_N	QH_N	H
H	H	L	Λ	L	X	X	QB_N	QC_N	QD_N	QE_N	QF_N	QG_N	QH_N	L
H	L	L	X	X	X	X	QA_0	QB_0	QC_0	QD_0	QE_0	QF_0	QG_0	QH_0

QA_0, QB_0, etc. = the level of QA, QB, etc. before the indicated steady state input conditions were established.

QA_N, QB_N, etc. = the level of QA, QB, etc. before the most recent Λ transition of the clock.

11.50. TTL199

8-Bit Shift Register with J-$\overline{K}$ Serial Inputs.




```
ENTITY ttl199 IS
    PORT(j: IN BIT ;
         k: IN BIT ;
         snl: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         d: IN BIT ;
         e: IN BIT ;
         f: IN BIT ;
         g: IN BIT ;
         h: IN BIT ;
         ci: IN BIT ;
         clk: IN BIT ;
         clr: IN BIT ;
         qa: OUT BIT ;
         qb: OUT BIT ;
         qc: OUT BIT ;
         qd: OUT BIT ;
         qe: OUT BIT ;
         qf: OUT BIT ;
         qg: OUT BIT ;
         qh: OUT BIT );
END ttl199 ;
```

NOTE: This library element is only available through schematic entry.

DESCRIPTION

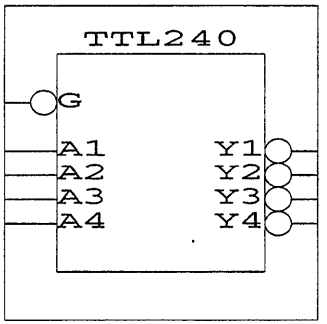
INPUTS							OUTPUTS							
CLR	SNL	CI	CLK	J	K	a...h	QA	QB	QC	QD	QE	QF	QG	QH
L	X	X	X	X	X	X	L	L	L	L	L	L	L	L
H	X	X	L	X	X	X	QA_0	QB_0	QC_0	QD_0	QE_0	QF_0	QG_0	QH_0
H	L	L	Λ	X	X	a...h	A	B	C	D	E	F	G	H
H	H	L	Λ	L	H	X	QA_0	QA_0	QB_N	QC_N	QD_N	QE_N	QF_N	QG_N
H	H	L	Λ	L	L	X	L	QA_N	QB_N	QC_N	QD_N	QE_N	QF_N	QG_N
H	H	L	Λ	H	H	X	H	QA_N	QB_N	QC_N	QD_N	QE_N	QF_N	QG_N
H	H	L	Λ	H	L	X	$\overline{QA}_N$	QA_N	QB_N	QC_N	QD_N	QE_N	QF_N	QG_N
H	X	H	Λ	X	X	X	QA_0	QB_0	QC_0	QD_0	QE_0	QF_0	QG_0	QH_0

QA_0, QB_0, etc. = the level of QA, QB, etc. before the indicated steady state input conditions were established.

QA_N, QB_N, etc. = the level of QA, QB, etc. before the most recent Λ transition of the clock.

11.51. TTL240

Buffer/Driver with Inverted Output and Low True Output Enable.




```
ENTITY ttl240 IS
    PORT(g: IN BIT ;
         a1: IN BIT ;
         a2: IN BIT ;
         a3: IN BIT ;
         a4: IN BIT ;
         y1: OUT X01Z ;
         y2: OUT X01Z ;
         y3: OUT X01Z ;
         y4: OUT X01Z );
END ttl240 ;
```

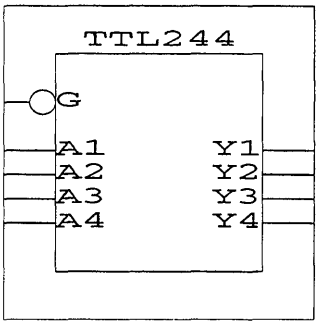
DESCRIPTION

INPUTS		OUTPUT
G	A(i)	Y(i)
H	X	Z
L	L	H
L	H	L

NOTE: This library element is only available through schematic entry.

11.52. TTL244

Buffer/Driver with Low True Output Enable.




```
ENTITY ttl244 IS
    PORT(g: IN BIT ;
         a1: IN BIT ;
         a2: IN BIT ;
         a3: IN BIT ;
         a4: IN BIT ;
         y1: OUT X01Z;
         y2: OUT X01Z ;
         y3: OUT X01Z ;
         y4: OUT X01Z );
END ttl244 ;
```

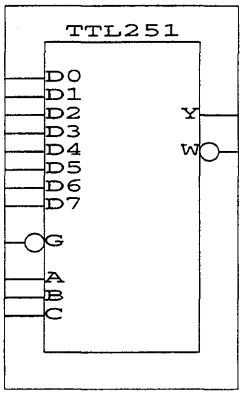
DESCRIPTION

INPUTS		OUTPUT
G	A(i)	Y(i)
H	X	Z
L	L	L
L	H	H

NOTE: This library element is only available through schematic entry.

11.53. TTL251

Data Selector/Multiplexer with 3-State Outputs.




```
ENTITY ttl251 IS
    PORT(d0: IN BIT ;
         d1: IN BIT ;
         d2: IN BIT ;
         d3: IN BIT ;
         d4: IN BIT ;
         d5: IN BIT ;
         d6: IN BIT ;
         d7: IN BIT ;
         g: IN BIT ;
         a: IN BIT ;
         b: IN BIT ;
         c: IN BIT ;
         y: OUT X01Z ;
         w: OUT X01Z );
END ttl251 ;
```

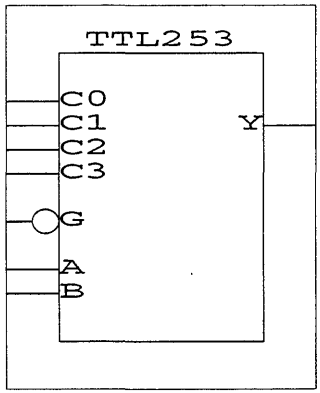
DESCRIPTION

INPUTS				OUTPUTS	
ENABLE	SELECT				
G	C	B	A	Y	W
H	X	X	X	Z	Z
L	L	L	L	D0	$\overline{D0}$
L	L	L	H	D1	$\overline{D1}$
L	L	H	L	D2	$\overline{D2}$
L	L	H	H	D3	$\overline{D3}$
L	H	L	L	D4	$\overline{D4}$
L	H	L	H	D5	$\overline{D5}$
L	H	H	L	D6	$\overline{D6}$
L	H	H	H	D7	$\overline{D7}$

NOTE: This library element is only available through schematic entry.

11.54. TTL253

4-Line to 1-Line Data Selector/Multiplexer with 3-State Outputs.




```
ENTITY ttl253 IS
    PORT(a: IN BIT ;
         b: IN BIT ;
         g: IN BIT ;
         c0: IN BIT ;
         c1: IN BIT ;
         c2: IN BIT ;
         c3: IN BIT ;
         y: OUT X01Z );
END ttl253 ;
```

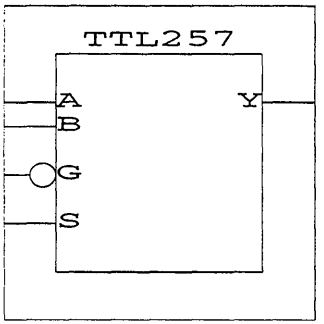
DESCRIPTION

INPUTS							OUTPUT
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	H	Z
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

NOTE: This library element is only available through schematic entry.

11.55. TTL257

2-Line to 1-Line Data Selector/Multiplexer with 3-State Output.




```
ENTITY ttl257 IS
    PORT(a: IN BIT ;
         b: IN BIT ;
         g: IN BIT ;
         s: IN BIT ;
         y: OUT X01Z );
END ttl257 ;
```

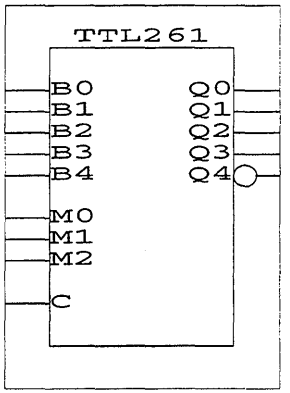
DESCRIPTION

INPUTS				OUTPUT
G	**S**	**A**	**B**	**Y**
H	X	X	X	Z
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

NOTE: This library element is only available through schematic entry.

11.56. TTL261

2-Bit by 4-Bit Parallel Binary Multiplier.




```
ENTITY ttl261 IS
    PORT(c: IN BIT ;
         b0: IN BIT ;
         b1: IN BIT ;
         b2: IN BIT ;
         b3: IN BIT ;
         b4: IN BIT ;
         m0: IN BIT ;
         m1: IN BIT ;
         m2: IN BIT ;
         q0: OUT BIT ;
         q1: OUT BIT ;
         q2: OUT BIT ;
         q3: OUT BIT ;
         q4: OUT BIT );
END ttl261 ;
```

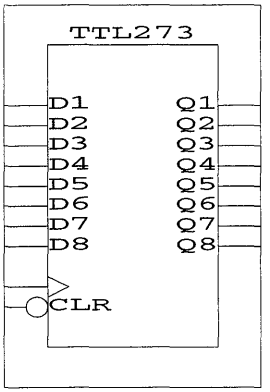
DESCRIPTION

INPUTS				OUTPUTS				
C	**M2**	**M1**	**M0**	**Q4**	**Q3**	**Q2**	**Q1**	**Q0**
L	X	X	X	$Q4_0$	$Q3_0$	$Q2_0$	$Q1_0$	$Q0_0$
H	L	L	L	H	L	L	L	L
H	L	L	H	$\overline{B4}$	B4	B3	B2	B1
H	L	H	L	$\overline{B4}$	B4	B3	B2	B1
H	L	H	H	$\overline{B4}$	B3	B2	B1	B0
H	H	L	L	B4	$\overline{B3}$	$\overline{B2}$	$\overline{B1}$	$\overline{B0}$
H	H	L	H	B4	$\overline{B4}$	$\overline{B3}$	$\overline{B2}$	$\overline{B1}$

INPUTS				OUTPUTS				
C	**M2**	**M1**	**M0**	**Q4**	**Q3**	**Q2**	**Q1**	**Q0**
H	H	H	L	B4	$\overline{B4}$	$\overline{B3}$	$\overline{B2}$	$\overline{B1}$
H	H	H	H	H	L	L	L	L

11.57. TTL273

Octal D-Type Flip-Flop with Clear.




```
ENTITY ttl273 IS
    PORT (d1: IN BIT ;
          d2: IN BIT ;
          d3: IN BIT ;
          d4: IN BIT ;
          d5: IN BIT ;
          d6: IN BIT ;
          d7: IN BIT ;
          d8: IN BIT ;
          clk: IN BIT ;
          clr: IN BIT ;
          q1: OUT BIT ;
          q2: OUT BIT ;
          q3: OUT BIT ;
          q4: OUT BIT ;
          q5: OUT BIT ;
          q6: OUT BIT ;
          q7: OUT BIT ;
          q8: OUT BIT );
END ttl273 ;
```

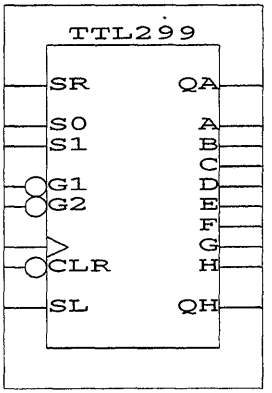
NOTE: This library element is only available through schematic entry.

DESCRIPTION

INPUTS			OUTPUT
CLR	**CLK**	**D(i)**	**Q(i)**
L	X	X	L
H	Λ	H	H
H	Λ	L	L
H	L	X	$Q(i)_0$

11.58. TTL299

8-Bit Universal Shift/Storage Register.




```
ENTITY ttl299 IS
    PORT (sr: IN BIT ;
          s0: IN BIT ;
          s1: IN BIT ;
          g1: IN BIT ;
          g2: IN BIT ;
          clk: IN BIT ;
          clr: IN BIT ;
          sl: IN BIT ;
          qa: OUT BIT ;
          a: INOUT X01Z ;
          b: INOUT X01Z ;
          c: INOUT X01Z ;
          d: INOUT X01Z ;
          e: INOUT X01Z ;
          f: INOUT X01Z ;
          g: INOUT X01Z ;
          h: INOUT X01Z ;
          qh: OUT BIT );
END ttl299 ;
```

NOTE: This library element is only available through schematic entry.

DESCRIPTION

INPUTS								INPUT/OUTPUT								OUT	
CLR	S1	S0	G1	G2	CLK	SL	SR	A	B	C	D	E	F	G	H	QA	QH
L	X	L	L	L	X	X	X	L	L	L	L	L	L	L	L	L	L
L	L	X	L	L	X	X	X	L	L	L	L	L	L	L	L	L	L
L	H	H	X	X	X	X	X	X	X	X	X	X	X	X	X	L	L
H	L	L	L	L	X	X	X	A_0	B_0	C_0	D_0	E_0	F_0	G_0	H_0	A_0	H_0
H	X	X	L	L	L	X	X	A_0	B_0	C_0	D_0	E_0	F_0	G_0	H_0	A_0	H_0
H	L	H	L	L	∧	X	H	H	A_n	B_n	C_n	D_n	E_n	F_n	G_n	H	G_n
H	L	H	L	L	∧	X	L	L	A_n	B_n	C_n	D_n	E_n	F_n	G_n	L	G_n
H	H	L	L	L	∧	H	X	B_n	C_n	D_n	E_n	F_n	G_n	H_n	H	B_n	H
H	H	L	L	L	∧	L	X	B_n	C_n	D_n	E_n	F_n	G_n	H_n	L	B_n	L
H	H	H	X	X	∧	X	X	a	b	c	d	e	f	g	h	a	h

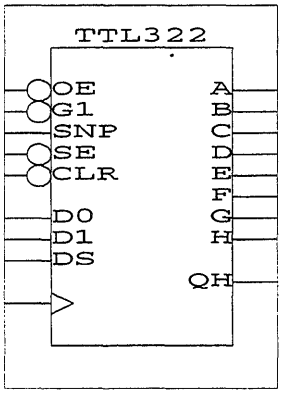
When one or both output controls (G1, G2) are high, the eight input/output terminals are disabled to the high-impedance state; however, sequential operation or clearing of the register is not affected.

The notation "A_0, B_0, C_0, ..." indicates that the A, B, C, etc., outputs retain their current value.

The notation "A_n, B_n, C_n, ..." indicates the output whose previous value is copied over to the current column. For example, when the inputs are set as specified in the sixth row, the A output goes high, the B output gets the previous value of A, the C output gets the previous value of B, etc.

11.59. TTL322

8-Bit Shift Register with Sign Extend.




```
ENTITY ttl322 IS
    PORT (g1: IN BIT ;
          snp: IN BIT ;
          se: IN BIT ;
          d1: IN BIT ;
          ds: IN BIT ;
          d0: IN BIT ;
          clk: IN BIT ;
          clr: IN BIT ;
          oe: IN BIT ;
          a: INOUT X01Z ;
          b: INOUT X01Z ;
          c: INOUT X01Z ;
          d: INOUT X01Z ;
          e: INOUT X01Z ;
          f: INOUT X01Z ;
          g: INOUT X01Z ;
          h: INOUT X01Z ;
          qh: OUT BIT );
END ttl322 ;
```

NOTE: This library element is only available through schematic entry.

DESCRIPTION

INPUTS							INPUT/OUTPUT								OUT
CLR	G	SNP	SE	DS	OE	CLK	A	B	C	D	E	F	G	H	QH
L	H	X	X	X	L	X	L	L	L	L	L	L	L	L	L
L	X	H	X	X	L	X	L	L	L	L	L	L	L	L	L
H	H	X	X	X	L	X	A_0	B_0	C_0	D_0	E_0	F_0	G_0	H_0	H_0
H	L	H	L	L	L	Λ	D0	A_n	B_n	C_n	D_n	E_n	F_n	G_n	G_n
H	L	H	L	H	L	Λ	D1	A_n	B_n	C_n	D_n	E_n	F_n	G_n	G_n
H	L	H	H	X	L	Λ	A_n	A_n	B_n	C_n	D_n	E_n	F_n	G_n	G_n
H	L	L	H	X	X	Λ	a	b	c	d	e	f	g	h	h

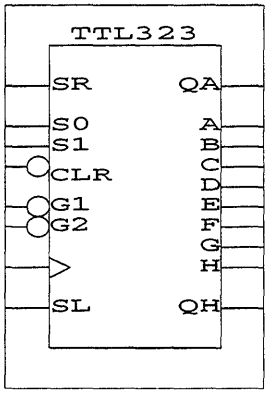
When the output enable is high, the eight input/output ports are disabled to the high-impedance state; however, sequential operation or clearing of the register is not affected. If both the G input and the SNP input are low while the clear input is low, the register is cleared while the eight I/O ports are disabled to the high-impedance state.

The notation "A_0, B_0, C_0, ..." indicates that the A, B, C, etc., outputs retain their current value.

The notation "A_n, B_n, C_n, ..." indicates the output whose previous value is copied over to the current column. For example, when the inputs are set as specified in the fourth row, the A output gets the value of input D0, the B output gets the previous value of A, the C output gets the previous value of B, etc.

11.60. TTL323

8-Bit Universal Shift/Storage Register.




```
ENTITY ttl323 IS
    PORT (sr: IN BIT ;
          s0: IN BIT ;
          s1: IN BIT ;
          clr: IN BIT ;
          g1: IN BIT ;
          g2: IN BIT ;
          clk: IN BIT ;
          sl: IN BIT ;
          qa: OUT BIT ;
          a: INOUT X01Z ;
          b: INOUT X01Z ;
          c: INOUT X01Z ;
          d: INOUT X01Z ;
          e: INOUT X01Z ;
          f: INOUT X01Z ;
          g: INOUT X01Z ;
          h: INOUT X01Z ;
          qh: OUT BIT );
END ttl323 ;
```

NOTE: This library element is only available through schematic entry.

DESCRIPTION

INPUTS								INPUT/OUTPUT								OUT	
CLR	S1	S0	G1	G2	CLK	SL	SR	A	B	C	D	E	F	G	H	QA	QH
L	X	L	L	L	X	X	X	L	L	L	L	L	L	L	L	L	L
L	L	X	L	L	X	X	X	L	L	L	L	L	L	L	L	L	L
L	H	H	X	X	X	X	X	X	X	X	X	X	X	X	X	L	L
H	L	L	L	L	X	X	X	A_0	B_0	C_0	D_0	E_0	F_0	G_0	H_0	A_0	H_0
H	X	X	L	L	L	X	X	A_0	B_0	C_0	D_0	E_0	F_0	G_0	H_0	A_0	H_0
H	L	H	L	L	Λ	X	H	H	A_n	B_n	C_n	D_n	E_n	F_n	G_n	H	G_n
H	L	H	L	L	Λ	X	L	L	A_n	B_n	C_n	D_n	E_n	F_n	G_n	L	G_n
H	H	L	L	L	Λ	H	X	B_n	C_n	D_n	E_n	F_n	G_n	H_n	H	B_n	H
H	H	L	L	L	Λ	L	X	B_n	C_n	D_n	E_n	F_n	G_n	H_n	L	B_n	L
H	H	H	X	X	Λ	X	X	a	b	c	d	e	f	g	h	a	h

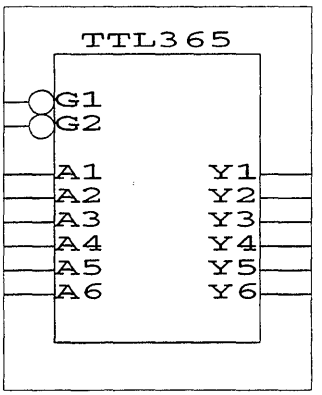
When one or both output controls (G1, G2) are high, the eight input/output terminals are disabled to the high impedance state; however, sequential operation or clearing of the register is not affected. The Clear function is synchronous.

The notation "A_0, B_0, C_0, ..." indicates that the A, B, C, etc., outputs retain their current value.

The notation "A_n, B_n, C_n, ..." indicates the output whose previous value is copied over to the current column. For example, when the inputs are set as specified in the sixth row, the A output goes high, the B output gets the previous value of A, the C output gets the previous value of B, etc.

11.61. TTL365

Hex Bus Drivers with 3-State Outputs.




```
ENTITY ttl365 IS
    PORT (g1: IN BIT ;
          g2: IN BIT ;
          a1: IN BIT ;
          a2: IN BIT ;
          a3: IN BIT ;
          a4: IN BIT ;
          a5: IN BIT ;
          a6: IN BIT ;
          y1: OUT X01Z ;
          y2: OUT X01Z ;
          y3: OUT X01Z ;
          y4: OUT X01Z ;
          y5: OUT X01Z ;
          y6: OUT X01Z );
END ttl365 ;
```

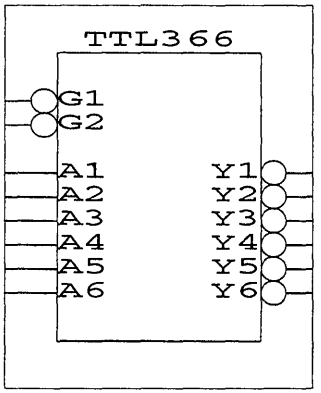
DESCRIPTION

INPUTS			OUTPUT
G1	G2	A(i)	Y(i)
H	X	X	Z
X	H	X	Z
L	L	L	L
L	L	H	H

NOTE: This library element is only available through schematic entry.

11.62. TTL366

Hex Inverted Bus Drivers with 3-State Outputs.




```
ENTITY ttl366 IS
    PORT (g1: IN BIT ;
          g2: IN BIT ;
          a1: IN BIT ;
          a2: IN BIT ;
          a3: IN BIT ;
          a4: IN BIT ;
          a5: IN BIT ;
          a6: IN BIT ;
          y1: OUT X01Z ;
          y2: OUT X01Z ;
          y3: OUT X01Z ;
          y4: OUT X01Z ;
          y5: OUT X01Z ;
          y6: OUT X01Z );
END ttl366 ;
```

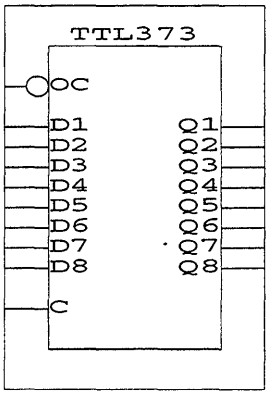
DESCRIPTION

INPUTS			OUTPUT
G1	**G2**	**A(i)**	**Y(i)**
H	X	X	Z
X	H	X	Z
L	L	L	H
L	L	H	L

NOTE: This library element is only available through schematic entry.

11.63. TTL373

Octal D-Type Transparent Latch with 3-State Outputs.




```
ENTITY ttl373 IS
    PORT (oc: IN BIT ;
          d1: IN BIT ;
          d2: IN BIT ;
          d3: IN BIT ;
          d4: IN BIT ;
          d5: IN BIT ;
          d6: IN BIT ;
          d7: IN BIT ;
          d8: IN BIT ;
          c: IN BIT ;
          q1: OUT X01Z ;
          q2: OUT X01Z ;
          q3: OUT X01Z ;
          q4: OUT X01Z ;
          q5: OUT X01Z ;
          q6: OUT X01Z ;
          q7: OUT X01Z ;
          q8: OUT X01Z );
END ttl373 ;
```

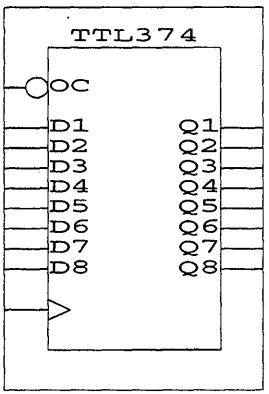
DESCRIPTION

INPUTS			OUTPUTS
OC	**C**	**D(i)**	**Q(i)**
L	H	H	H
L	H	L	L
L	L	X	$Q(i)_0$
H	X	X	Z

NOTE: This library element is only available through schematic entry.

11.64. TTL374

Octal D-Type Edge-Triggered Flip-Flops with 3-State Outputs.




```
ENTITY ttl374 IS
    PORT (oc: IN BIT ;
          d1: IN BIT ;
          d2: IN BIT ;
          d3: IN BIT ;
          d4: IN BIT ;
          d5: IN BIT ;
          d6: IN BIT ;
          d7: IN BIT ;
          d8: IN BIT ;
          clk: IN BIT ;
          q1: OUT X01Z ;
          q2: OUT X01Z ;
          q3: OUT X01Z ;
          q4: OUT X01Z ;
          q5: OUT X01Z ;
          q6: OUT X01Z ;
          q7: OUT X01Z ;
          q8: OUT X01Z );
END ttl374 ;
```

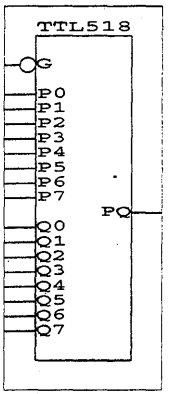
DESCRIPTION

INPUTS			OUTPUTS
OC	C	D(i)	Q(i)
L	Λ	H	H
L	Λ	L	L
L	L	X	$Q(i)_0$
H	X	X	Z

NOTE: This library element is only available through schematic entry.

11.65. TTL518

8-bit magnitude comparator.




```
ENTITY ttl518 IS
    PORT(g: IN BIT ;
         q0: IN BIT ;
         q1: IN BIT ;
         q2: IN BIT ;
         q3: IN BIT ;
         q4: IN BIT ;
         q5: IN BIT ;
         q6: IN BIT ;
         q7: IN BIT ;
         p0: IN BIT ;
         p1: IN BIT ;
         p2: IN BIT ;
         p3: IN BIT ;
         p4: IN BIT ;
         p5: IN BIT ;
         p6: IN BIT ;
         p7: IN BIT ;
         pq: OUT BIT );
END ttl518 ;
```

DESCRIPTION

```
PQ <=   (not G) and
        (not (Q7 xor P7)) and
        (not (Q6 xor P6)) and
        (not (Q5 xor P5)) and
        (not (Q4 xor P4)) and
        (not (Q3 xor P3)) and
        (not (Q2 xor P2)) and
        (not (Q1 xor P1)) and
        (not (Q0 xor P0));
```

PQ is at a high level when input P equals input Q.