INTRANEX

INTRANEX is a **programmable interconnect network** that accepts a N bit input W and produces a N bit output Z. The interconnect can be programmed to realize any mapping from W to Z.

University of Cincinnati - EECE6080
Fall 2013

Max Thrun
973 919 6593
max.thrun@gmail.com
(Coordinator)

Xiaohu Qi
513 652 2075
qixiaohuihaha@gmail.com

Contents

List of Figures

List of Tables

Listings

1 Pinout Diagram

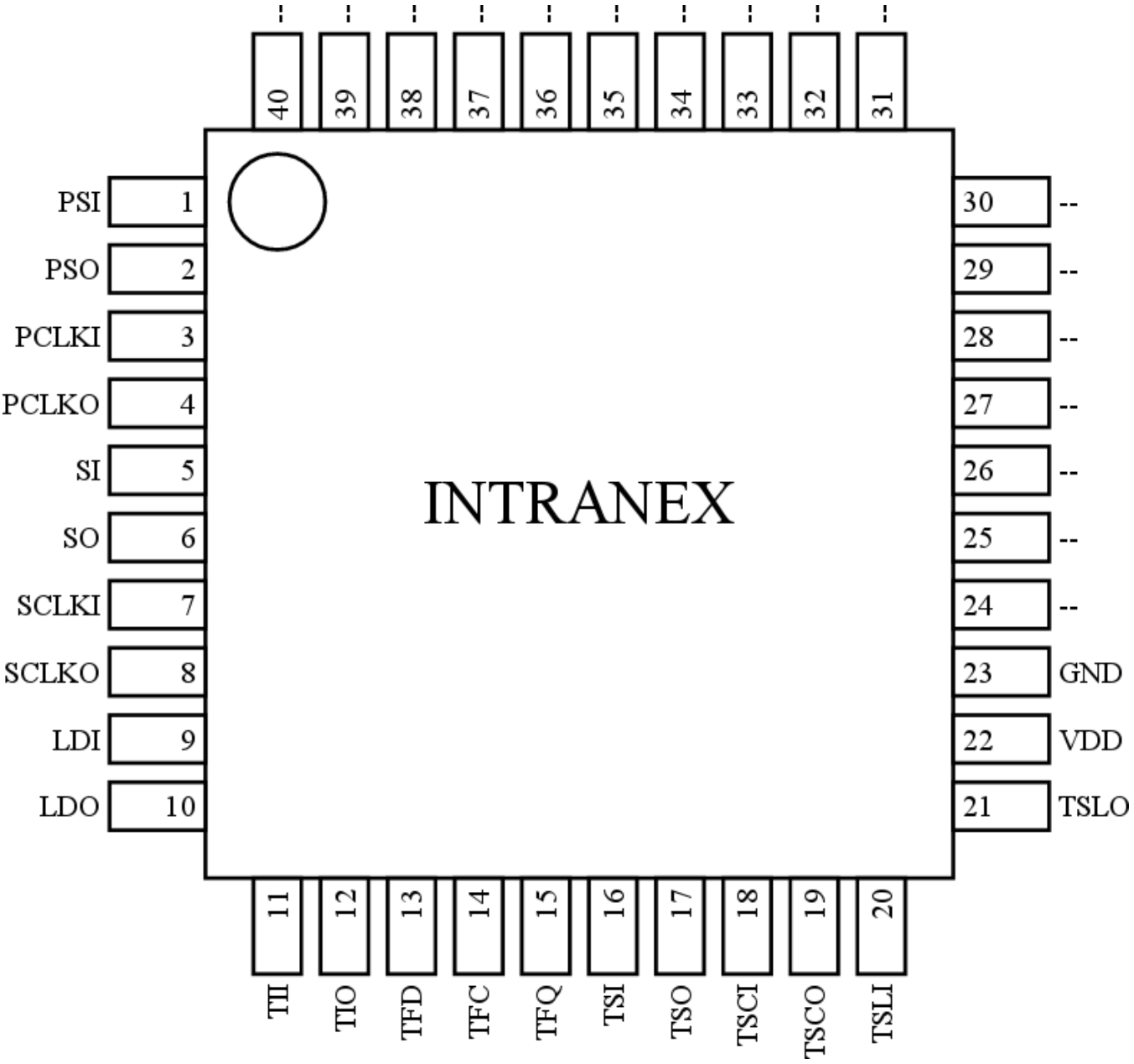

Figure 1: Pinout Diagram

Pin #	Name	Type	Description
1	PSI	I	PIN serial input
2	PSO	O	PIN serial output
3	PCLKI	I	PIN clock input
4	PCLKO	O	PIN clock output
5	SI	I	Serial input
6	SO	O	Serial output
7	SCLKI	I	Serial clock input
8	SCLKO	O	Serial clock output
9	LDI	I	Parallel load input (active low)
10	LDO	I	Parallel load output (active low)
11	TII	I	Test inverter input
12	TIO	O	Test interter output
13	TFD	I	Test flip-flop D input
14	TFC	I	Test flip-flop clock input
15	TFQ	O	Test flop-flop Q output
16	TSI	I	Test shift slice serial input
17	TSO	O	Test shift slice serial output
18	TSCI	I	Test shift slice clock input
19	TSCO	O	Test shift slice clock output
20	TSLI	I	Test shift slice load input
21	TSLO	O	Test shift slice load output
22	VDD	P	–
23	GND	P	–
24	–	-	–
25	–	-	–
26	–	-	–
27	–	-	–
28	–	-	–
29	–	-	–
30	–	-	–
31	–	-	–
32	–	-	–
33	–	-	–
34	–	-	–
35	–	-	–
36	–	-	–
37	–	-	–
38	–	-	–
39	–	-	–
40	–	-	–

Table 1: Pin Descriptions

2 Chip Functionality

2.1 Configuring the Programmable Interconnect Network

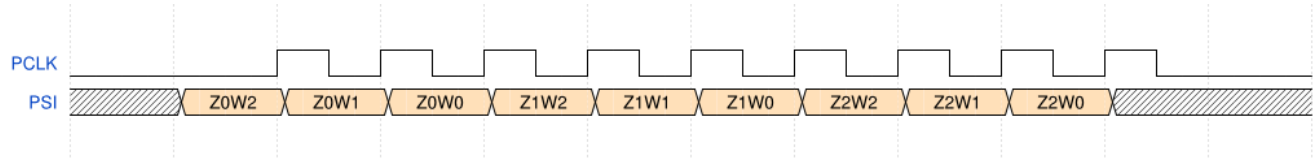

Figure 2: PIN Configuration

2.2 Loading and reading a value

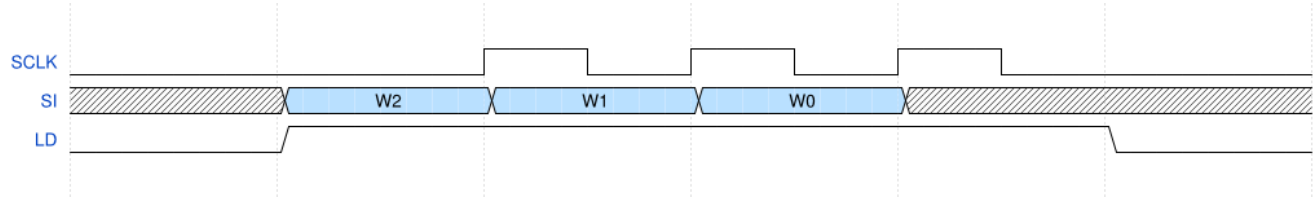

Figure 3: Loading a value

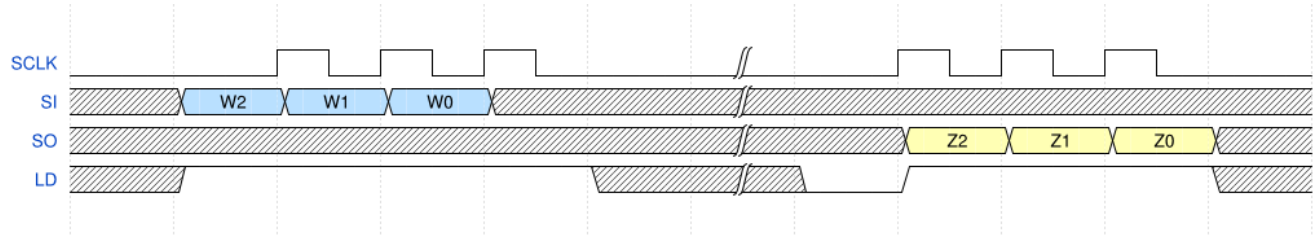

Figure 4: Loading a value and reading the result

2.3 Test Mode

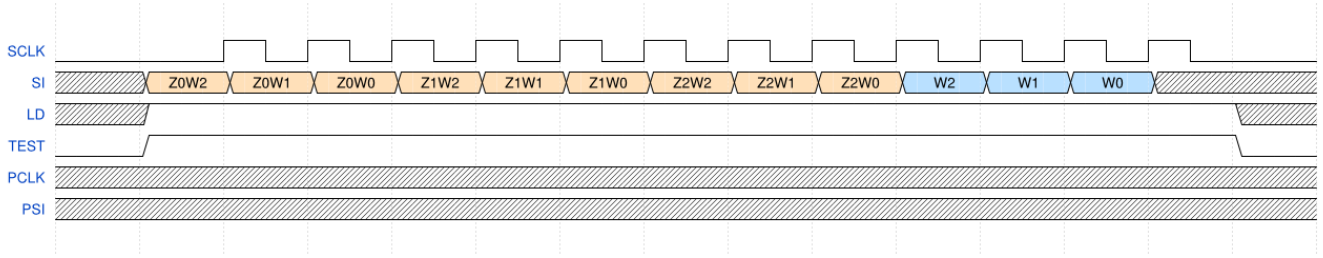

Figure 5: Enabling test mode and loading all DFFs

3 Design Decisions

4 Block Diagrams

4.1 Top Level

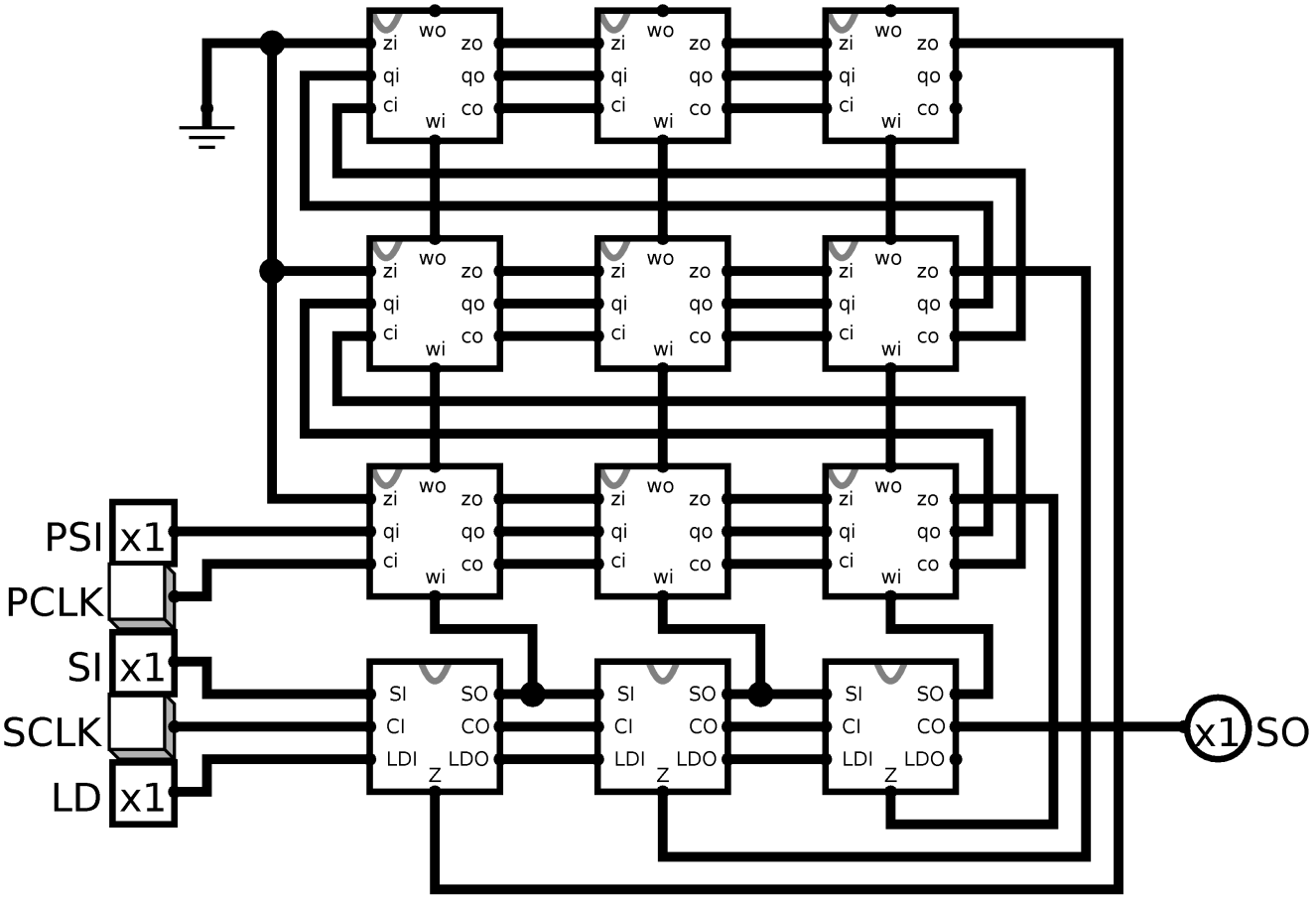

Figure 6: Top Level Block Diagram (3-Bit Configuration)

4.2 Parallel Load Shift Register

4.2.1 Bit-slicing Scheme

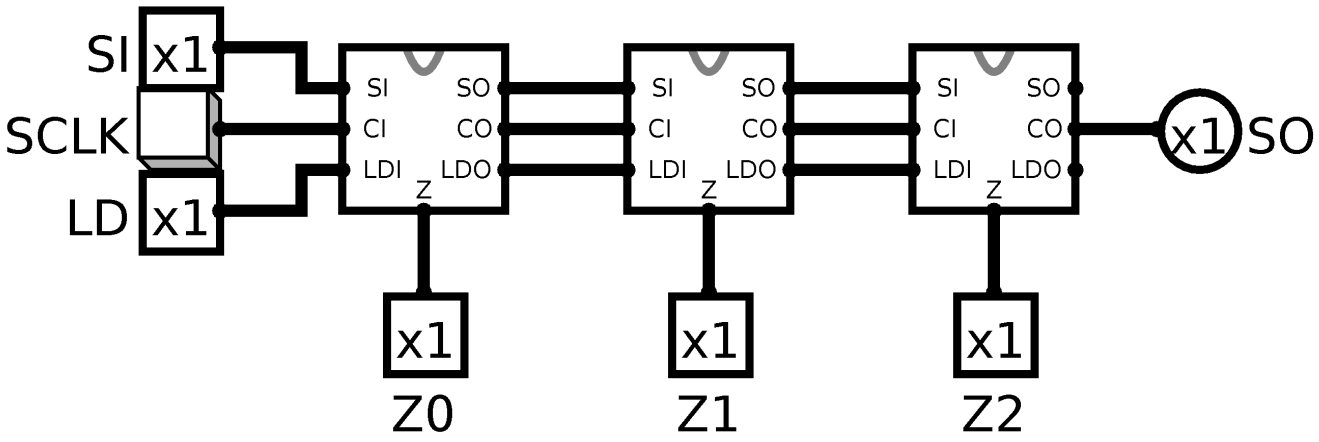

Figure 7: Parallel Load Bit-Sliced Shifter Register (3-Bit Configuration)

4.2.2 Bit-Slice

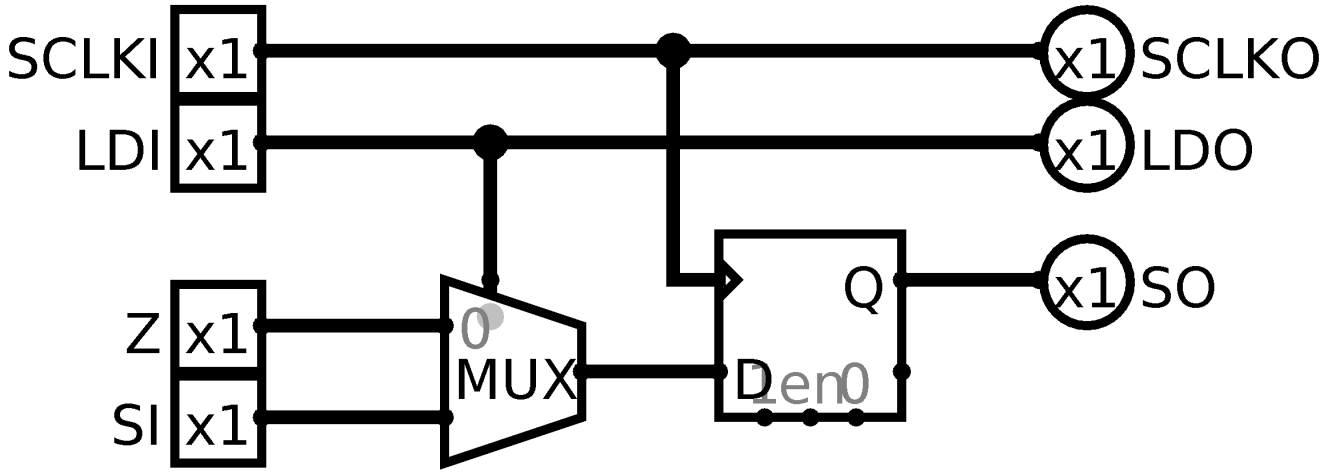

Figure 8: Parallel Load Shifter Register Bit-Slice

4.3 Programmable Interconnect Network

4.3.1 Bit-slicing Scheme

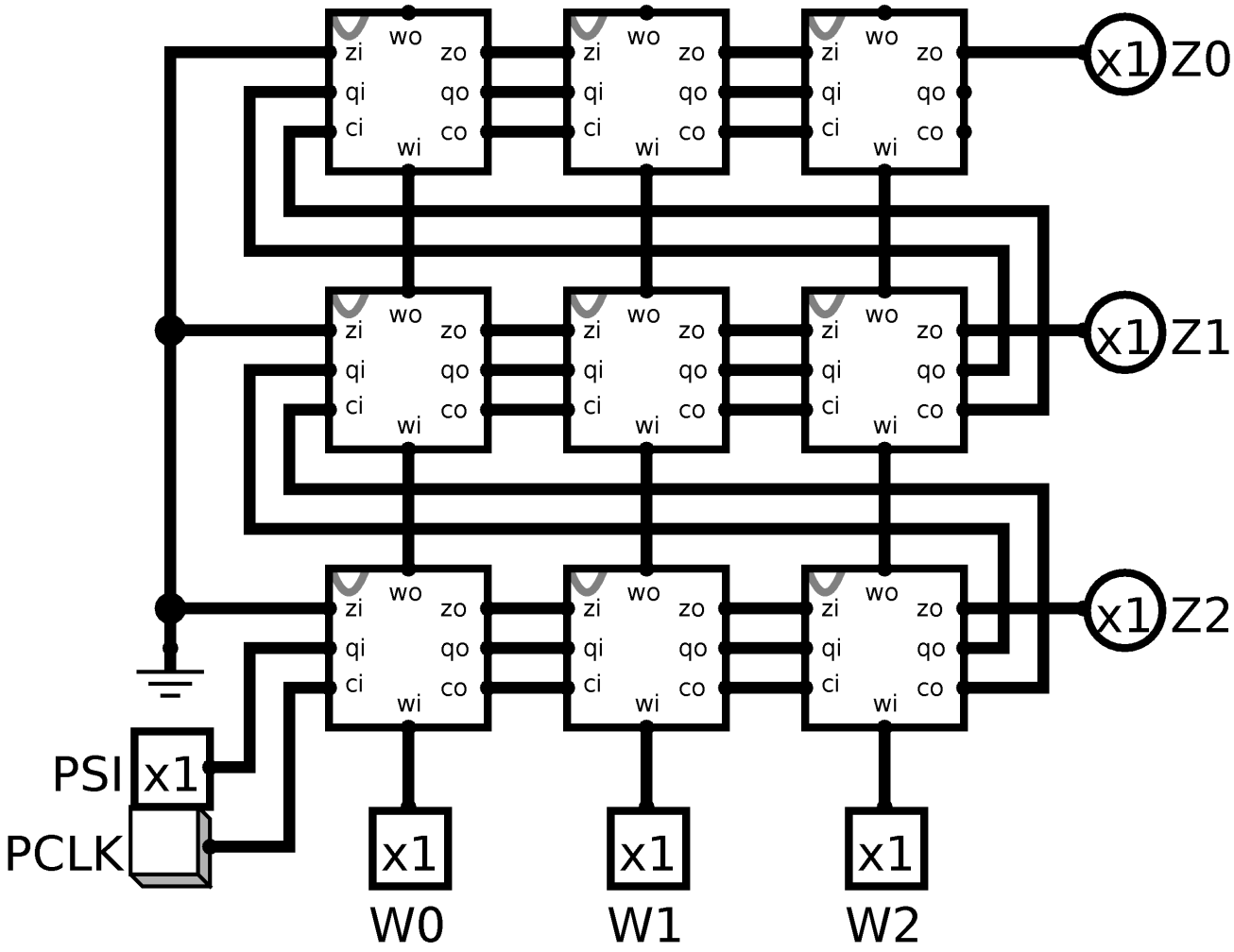

Figure 9: Bit-Sliced Programmable Interconnect Network (3-Bit Configuration)

4.3.2 Bit-Slice

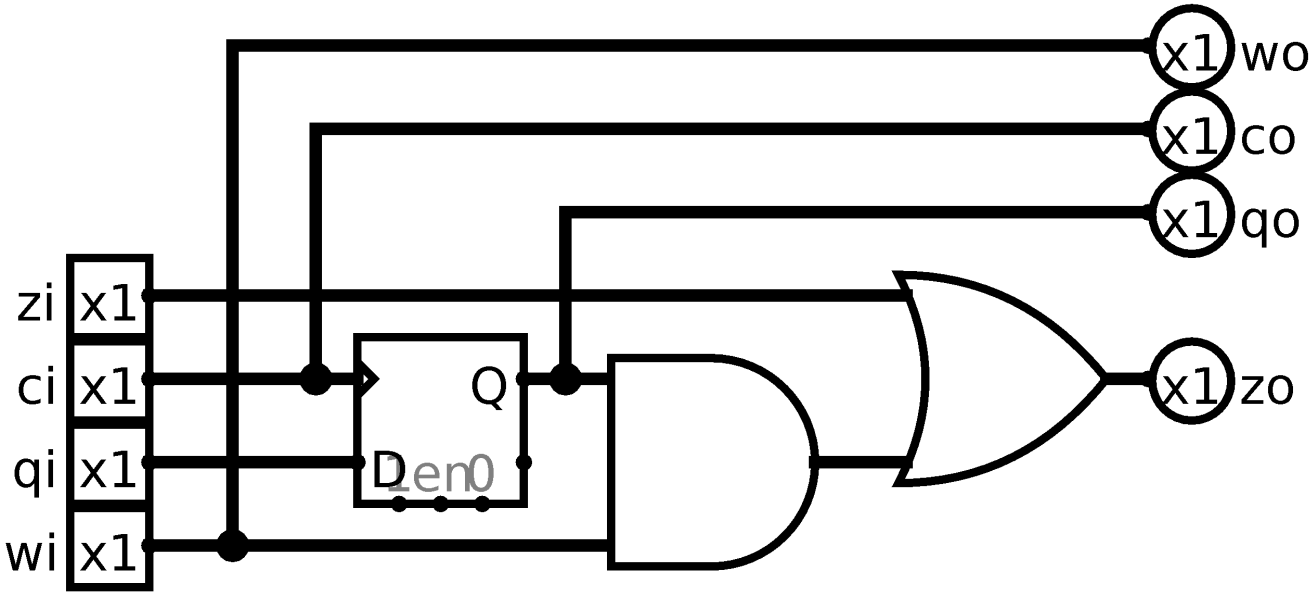

Figure 10: Programmable Interconnect Network Bit-Slice

5 VHDL Models

5.1 Top Level

5.2 PIN

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity pin is
5      generic(
6          n : integer := 3
7      );
8      port(
9          clk : in std_logic;
10         psi  : in std_logic;
11         z : out  std_logic_vector((n-1) downto 0);
12         w : in  std_logic_vector((n-1) downto 0)
13     );
14 end pin;
15
16 architecture rtl of pin is
17
18     component pin_slice is
19         port(
20             zi : in std_logic;
21             qi : in std_logic;
22             wi : in std_logic;
23             ci : in std_logic;
24             zo : out std_logic;
25             qo : out std_logic;
```

```
26              wo : out std_logic;
27              co : out std_logic
28          );
29      end component;
30
31      --type carry_array is array (0 to n) of std_logic_vector(n downto 0);
32      type carry_array is array (0 to n, 0 to n) of std_logic;
33      signal zo : carry_array;
34      signal co : carry_array;
35      signal wo : carry_array;
36      signal qo : carry_array;
37
38      begin
39
40      -- setup first and last inputs for each row
41      z_connect : for i in 0 to n-1 generate
42          zo(i, 0) <= '0';
43          z(i) <= zo(i, n);
44      end generate;
45
46      -- setup first inputs for each column
47      w_connect : for i in 0 to n-1 generate
48          wo(0, i) <= w(i);
49      end generate;
50
51      -- setup row transfer
52      -- (last output of row to first input of next row)
53      r_connect : for i in 0 to n-2 generate
54          qo(i, 0) <= qo(i+1, n);
55          co(i, 0) <= co(i+1, n);
56      end generate;
57
58      -- connect external inputs
59      qo(n-1, 0) <= psi;
60      co(n-1, 0) <= clk;
61
62      -- generate the grid of slices
63      pin_z_gen : for zz in 0 to n-1 generate
64          pin_w_gen : for ww in 0 to n-1 generate
65              pin_i: pin_slice port map(
66                  zi => zo(zz, ww),
67                  qi => qo(zz, ww),
68                  wi => wo(zz, ww),
69                  ci => co(zz, ww),
70                  zo => zo(zz, ww+1),
71                  qo => qo(zz, ww+1),
72                  wo => wo(zz+1, ww),
73                  co => co(zz, ww+1)
74              );
75          end generate;
76      end generate;
77
78  end rtl;
```

Listing 1: PIN VHDL Module

5.3 PIN Slice

```
library ieee;
use ieee.std_logic_1164.all;

entity pin_slice is
    port(
        zi : in std_logic;
        qi : in std_logic;
        wi : in std_logic;
        ci : in std_logic;
        zo : out std_logic;
        qo : out std_logic;
        wo : out std_logic;
        co : out std_logic
    );
end pin_slice;

architecture rtl of pin_slice is

    signal g1_o : std_logic := '0';
    signal g2_o : std_logic := '0';

begin

    g1 : entity work.dffposx1 port map(ci, qi, g1_o);
    g2 : entity work.aoi21x1  port map(wi, g1_o, zi, g2_o);
    g3 : entity work.invx1    port map(g2_o, zo);

    -- pass through
    co <= ci;
    wo <= wi;
    qo <= g1_o;

end rtl;
```

Listing 2: PIN Slice VHDL Module

5.4 Shifter

```
library ieee;
use ieee.std_logic_1164.all;

entity shift is
    generic(
        n : integer := 3
    );
    port(
        clk : in  std_logic;
        ld  : in  std_logic;
        si  : in  std_logic;
        z   : in  std_logic_vector((n-1) downto 0);
        w   : out std_logic_vector((n-1) downto 0)
    );
end shift;

architecture rtl of shift is

```

```
19      component shift_slice
20          port(
21              sclki : in  std_logic;
22              sclko : out std_logic;
23              ldi   : in  std_logic;
24              ldo   : out std_logic;
25              si    : in  std_logic;
26              so    : out std_logic;
27              z     : in  std_logic
28          );
29      end component;
30
31      -- vector to hold  values between slices
32      signal so : std_logic_vector(n downto 0) := (others => '0');
33      signal clko : std_logic_vector(n downto 0) := (others => '0');
34      signal ldo : std_logic_vector(n downto 0) := (others => '0');
35
36      begin
37
38      -- input of slice 0 comes from module input
39      so(0) <= si;
40      ldo(0) <= ld;
41      clko(0) <= clk;
42
43      -- generate N slices
44      shift_gen : for i in 0 to n-1 generate
45          shift_i: shift_slice port map(
46              sclki => clko(i),
47              sclko => clko(i+1),
48              ldi => ldo(i),
49              ldo => ldo(i+1),
50              si => so(i),
51              so => so(i+1),
52              z => z(i)
53          );
54      end generate;
55
56      -- connect the output of each slice to final output vector
57      connect : for i in 0 to n-1 generate
58          w(i) <= so(i+1);
59      end generate;
60
61  end rtl;
```

Listing 3: Parallel Load Shifter VHDL Module

5.5 Shifter Slice

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity shift_slice is
5       port(
6           sclki : in  std_logic;
7           sclko : out std_logic;
8           ldi   : in  std_logic;
9           ldo   : out std_logic;
```

```
10          si     : in  std_logic;
11          so     : out std_logic;
12          z      : in  std_logic
13      );
14  end shift_slice;
15
16  architecture rtl of shift_slice is
17
18      signal g1_o : std_logic := '0';
19
20  begin
21
22      g1 : entity work.mux2x1   port map(z, si, ldi, g1_o);
23      g2 : entity work.dffposx1 port map(sclki, g1_o, so);
24
25      -- pass through
26      sclko <= sclki;
27      ldo <= ldi;
28
29  end rtl;
```

Listing 4: Parallel Load Shifter Slice VHDL Module

5.6 Gates

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity aoi21x1 is
5       generic(delay : time := 0 ps);
6       port(
7           a : in std_logic;
8           b : in std_logic;
9           c : in std_logic;
10          y : out std_logic
11      );
12  end aoi21x1;
13
14  architecture rtl of aoi21x1 is begin
15      process(a, b, c) begin
16          y <= not ((a and b) or c) after delay;
17      end process;
18  end rtl;
```

Listing 5: AOI21X1 VHDL Module

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity dffposx1 is
5       generic(delay : time := 0 ps);
6       port(
7           c : in std_logic;
8           d : in std_logic;
9           q : out std_logic := '0'
10      );
```

```
end dffposx1;

architecture rtl of dffposx1 is begin
    process(c) begin
        if rising_edge(c) then
            q <= d after delay;
        end if;
    end process;
end rtl;
```

Listing 6: DFFPOSX1 VHDL Module

```
library ieee;
use ieee.std_logic_1164.all;

entity invx1 is
    generic(delay : time := 0 ps);
    port(
        a : in std_logic;
        y : out std_logic
    );
end invx1;

architecture rtl of invx1 is begin
    y <= not a after delay;
end rtl;
```

Listing 7: INVX1 VHDL Module

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2x1 is
    generic(delay : time := 0 ps);
    port(
        a : in std_logic;
        b : in std_logic;
        s : in std_logic;
        y : out std_logic
    );
end mux2x1;

architecture rtl of mux2x1 is begin
    process(a, b, s) begin
        if (s = '1') then
            y <= b after delay;
        else
            y <= a after delay;
        end if;
    end process;
end rtl;
```

Listing 8: MUX2X1 VHDL Module

6 VHDL Test Benches

6.1 Top Level

6.2 PIN

6.3 PIN Slice

```
library ieee;
use ieee.std_logic_1164.all;

entity pin_slice_tb is
end pin_slice_tb;

architecture tb_rtl of pin_slice_tb is

    signal zi : std_logic := '0';
    signal qi : std_logic := '0';
    signal wi : std_logic := '0';
    signal ci : std_logic := '0';
    signal zo : std_logic;
    signal qo : std_logic;
    signal wo : std_logic;
    signal co : std_logic;

    component pin_slice
        port(
            zi : in std_logic;
            qi : in std_logic;
            wi : in std_logic;
            ci : in std_logic;
            zo : out std_logic;
            qo : out std_logic;
            wo : out std_logic;
            co : out std_logic
        );
    end component;

begin

    uut : pin_slice
    port map(
        zi => zi,
        qi => qi,
        wi => wi,
        ci => ci,
        zo => zo,
        qo => qo,
        wo => wo,
        co => co
    );

    process
        type pattern_type is record
            -- inputs
            zi, qi, wi : std_logic;
            -- output
```

```
50              zo : std_logic;
51          end record;
52
53          type pattern_array is array (natural range <>) of pattern_type;
54          constant patterns : pattern_array :=
55          --zi   qi   wi   zo
56          (('0','0','0', '0'),
57           ('0','0','1', '0'),
58           ('0','1','0', '0'),
59           ('0','1','1', '1'),
60           ('1','0','0', '1'),
61           ('1','0','1', '1'),
62           ('1','1','0', '1'),
63           ('1','1','1', '1'));
64
65      begin
66          -- check each pattern
67          for i in patterns'range loop
68
69              -- set the inputs
70              zi <= patterns(i).zi;
71              qi <= patterns(i).qi;
72              wi <= patterns(i).wi;
73              wait for 10 ns;
74
75              -- pulse the clock and check clock passthrough
76              ci <= '1';
77              wait for 10 ns;
78              assert co = '1' report "CO does not equal 1" severity error;
79              ci <= '0';
80              wait for 10 ns;
81              assert co = '0' report "CO does not equal 0" severity error;
82
83              -- check the outputs
84              assert qo = patterns(i).qi report "QI not equal QO" severity error;
85              assert zo = patterns(i).zo report "ZO does not match pattern" severity error;
86
87          end loop;
88
89          report "Test Complete" severity note;
90          wait;
91
92      end process;
93
94  end tb_rtl;
```

Listing 9: PIN Slice VHDL Test Bench

6.4 Shifter

6.5 Shifter Slice

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity shift_slice_tb is
```

```
5  end shift_slice_tb;
6
7  architecture tb_rtl of shift_slice_tb is
8
9      signal sclki : std_logic := '0';
10     signal sclko : std_logic;
11     signal ldi   : std_logic := '0';
12     signal ldo   : std_logic;
13     signal si    : std_logic := '0';
14     signal so    : std_logic;
15     signal z     : std_logic := '0';
16
17     component shift_slice is
18         port(
19             sclki : in  std_logic;
20             sclko : out std_logic;
21             ldi   : in  std_logic;
22             ldo   : out std_logic;
23             si    : in  std_logic;
24             so    : out std_logic;
25             z     : in  std_logic
26         );
27     end component;
28
29 begin
30
31     uut : shift_slice
32     port map(
33         sclki => sclki,
34         sclko => sclko,
35         ldi   => ldi  ,
36         ldo   => ldo  ,
37         si    => si   ,
38         so    => so   ,
39         z     => z
40     );
41
42     process
43         type pattern_type is record
44             -- inputs
45             ldi, z, si : std_logic;
46             -- output
47             so : std_logic;
48         end record;
49
50         type pattern_array is array (natural range <>) of pattern_type;
51         constant patterns : pattern_array :=
52         --ldi  z  si   so
53         (('0','0','0', '0'),
54          ('0','0','1', '0'),
55          ('0','1','0', '1'),
56          ('0','1','1', '1'),
57          ('1','0','0', '0'),
58          ('1','0','1', '1'),
59          ('1','1','0', '0'),
60          ('1','1','1', '1'));
61
62     begin
63         -- check each pattern
```

```
64          for i in patterns'range loop
65
66              -- set the inputs
67              ldi <= patterns(i).ldi;
68              z   <= patterns(i).z;
69              si  <= patterns(i).si;
70              wait for 10 ns;
71
72              -- pulse the clock and check clock passthrough
73              sclki <= '1';
74              wait for 10 ns;
75              assert sclko = '1' report "SCLKO does not equal 1" severity error;
76              assert ldo = patterns(i).ldi report "SCLKO does not equal 1" severity error;
77              sclki <= '0';
78              wait for 10 ns;
79              assert sclko = '0' report "SCLKO does not equal 0" severity error;
80              assert ldo = patterns(i).ldi report "SCLKO does not equal 1" severity error;
81
82              -- check the output
83              assert so = patterns(i).so report "SO is incorrect" severity error;
84
85          end loop;
86
87          report "Test Complete" severity note;
88          wait;
89
90      end process;
91
92  end tb_rtl;
```

Listing 10: Parallel Load Shifter Slice VHDL Test Bench

7 Work Division