F VHDL Code of the Programmable Coincidence Logic

Below is the VHDL code for implementing the coincidence logic for a static entanglement swapping experiment with a more-complete Bell-state analyzer (see Section 8.3). This logic was implemented in a XC9536XL CPLD device from Xilinx, using the Foundation Express softwar from Xilinx.

F.1 Main Routine: telecoincall.vhd

This routine takes care of the inputs and outputs of the device, and also calls the required sub routines.

```
library IEEE; use IEEE.std_logic_1164.all;


entity telecoincall is
    port (
        dlong: in STD_LOGIC_VECTOR (0 to 7);
        clk: in STD_LOGIC;
        sel: in integer range 0 to 2;
        data: out STD_LOGIC_VECTOR (0 to 7)
    );

end telecoincall;

architecture coinc_arch of telecoincall is


component inplatch is
    port (
        d: in STD_LOGIC;
        clk: in STD_LOGIC;
        dd: out STD_LOGIC
    );
```

```
end component inplatch;


component compar is
    port (
        d: in STD_LOGIC_VECTOR (0 to 7);
        clk: in STD_LOGIC;
        sel: in integer range 0 to 2;
        data: out STD_LOGIC_VECTOR (0 to 7)
    );
end component compar;


signal dt: STD_LOGIC_VECTOR (0 to 7);

begin

  l0: inplatch port map(dlong(0),clk,dt(0));
  l1: inplatch port map(dlong(1),clk,dt(1));
  l2: inplatch port map(dlong(2),clk,dt(2));
  l3: inplatch port map(dlong(3),clk,dt(3));
  l4: inplatch port map(dlong(4),clk,dt(4));
  l5: inplatch port map(dlong(5),clk,dt(5));
  l6: inplatch port map(dlong(6),clk,dt(6));
  l7: inplatch port map(dlong(7),clk,dt(7));

  c1: compar port map(dt,clk,sel,data);


end coinc_arch;
```

F.2 Sub Routine: inplatch.vhd

This sub routine transforms an input detection pulse which can be in the "high" state for times longer than one clock period to an internal variable, which is only "high" for the first clock period directly after the input pulse started. This is necessary, because the logic only compares the status of the inputs at a clock edge, which would lead to multiple counts for an input pulse which lasts for several clock edges.

```
library IEEE; use IEEE.std_logic_1164.all;

entity inplatch is
    port (
        d: in STD_LOGIC;
```

```
        clk: in STD_LOGIC;
        dd: out STD_LOGIC
    );
end inplatch;

architecture inplatch_arch of inplatch is

signal temp: STD_LOGIC;

begin
  -- <<enter your statements here>>
  prcss_inplatch: process (clk)
    begin
      if (CLK = '1' and CLK'Event) then
        if (d='1' and temp='0') then
          dd <= '1' ;
          temp <= '1';
        elsif (d='1' and temp='1') then
          dd <= '0' ;
          temp <= '1';
        else
          dd <= '0' ;
          temp <= '0';
      end if;
    end if;
  end process;


end inplatch_arch;
```

F.3 Sub Routine: compar.vhd

This subroutine performs the actual correlation of the detection pulses. Depending on the setting of the two select inputs, the routine checks for certain bit patterns on the eight detectors, and generates a pulse at the corresponding output.

```
library IEEE; use IEEE.std_logic_1164.all;

-- get all possible teleportation combinations for 8 detectors and 2 bell
states

entity compar is
    port (
        d: in STD_LOGIC_VECTOR (0 to 7);
```

```
        sel: in integer range 0 to 2;
        clk: in STD_LOGIC;
        data: out STD_LOGIC_VECTOR (0 to 7)
    );

end compar;

architecture compar_arch of compar is

begin

  p1: process (clk)
  begin

  if (clk'event and clk='1') then
      -- input data structure: 8 Bits: (Bob0:) D0+,D0-, (BSA:) D1+, D1-. D2+, D2-,
      --                               (Bob3:) D3+, D3-

    if sel=0 then

      -- determine all 8 4-fold coincidences
      -- output data meaning: bits 0-3: 0+ psi- 0+, 0+ psi- 0-, 0- psi- 0+, 0- psi- 0-
      --                      bits 4-7: 0+ psi+ 0+, 0+ psi+ 0-, 0- psi+ 0+, 0- psi+ 0-

      if (d="10100110" or
          d="10011010" or
          d="10101010" or
          d="10010110") then
             data <= "10000000"; -- 0+ psi- 0+

      elsif  (d="10100101" or
              d="10011001" or
              d="10101001" or
              d="10010101") then
             data <= "01000000";  -- 0+ psi- 0-

      elsif  (d="01100110" or
              d="01011010" or
              d="01010110" or
              d="01101010") then
             data <= "00100000";   -- 0- psi- 0+

      elsif  (d="01100101" or
              d="01011001" or
              d="01010101" or
              d="01101001") then
```

```
        data <= "00010000";   -- 0- psi- 0-

  elsif  (d="10110010" or
          d="10001110") then
        data <= "00001000";   -- 0+ psi+ 0+

  elsif  (d="10110001" or
          d="10001101") then
        data <= "00000100";   -- 0+ psi+ 0-

  elsif  (d="01110010" or
          d="01001110") then
        data <= "00000010";   -- 0- psi+ 0+

  elsif  (d="01110001" or
          d="01001101") then
        data <= "00000001";   -- 0- psi+ 0-

  else
        data <= "00000000";
  end if;

elsif sel=1 then

  -- determine several 2-folds for adjustement
  -- outputs 0 to 4: 2-folds D0+^D1* D0+^D2*, D0-^D1* D0-^D2*,
  -- D3+^D1* D3+^D2*, D3-^D1* D3-^D2*, D1*^D2*

       if (d="10100000" or
           d="10010000" or
           d="10001000" or
           d="10000100") then
         data <= "10000000"; -- 2-fold D0+^D1* or D0+^D2*

       elsif (d="01100000" or
              d="01010000" or
              d="01001000" or
              d="01000100") then
         data <= "01000000"; -- 2-fold D0-^D1* or D0-^D2*

       elsif (d="00100010" or
              d="00010010" or
              d="00001010" or
              d="00000110") then
         data <= "00100000"; -- 2-fold D3+^D1* or D3+^D2*
```

```
            elsif (d="00100001" or
                   d="00010001" or
                   d="00001001" or
                   d="00000101") then
               data <= "00010000"; -- 2-fold D3-^D1* or D3-^D2*

            elsif (d="00101000" or
                   d="00011000" or
                   d="00100100" or
                   d="00010100") then
               data <= "00001000"; -- 2-fold D1*^D2*

            else
               data <= "00000000";
            end if;

    elsif sel=2 then

            -- all signals mapped through for singles

            data <= d;
    end if;
  end if;

  end process p1;

end compar_arch;
```