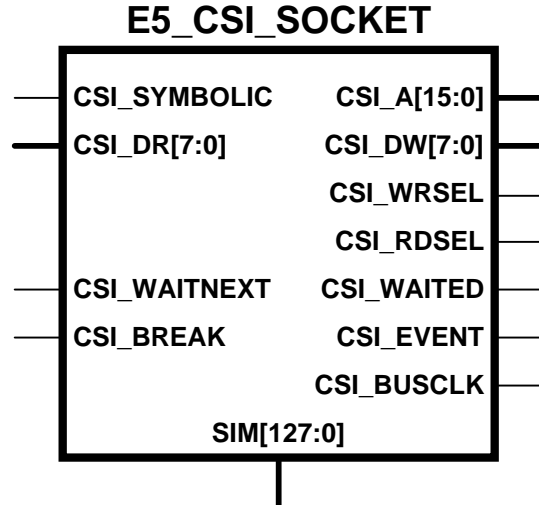


## E5\_CSI\_SOCKET VHDL Function

### Functional Diagram



### Port Connections

Port Name	Direction	Type	Description
CSI_SYMBOLIC	Input	std_logic	The name of the signal connected to this port defines the symbolic name for the decoded address. The symbolic name is passed to the 8051 compiler using the FastChip Generate function.
CSI_DR[7:0]	Input	std_logic_vector (7 downto 0)	The data passed back to the CSI bus when the current bus master accesses the address ranged defined by this E5_CSI_SOCKET instance. The data is presented on the bus when the CSI_RDSEL signal is asserted.
CSI_WAITNEXT	Input	std_logic	If the current bus master is accessing the selected address range, and the generic isAutoWaited=true, the application can request additional wait-states by asserting the CSI_WAITNEXT signal.
CSI_BREAK	Input	std_logic	Asserting this signal forces a breakpoint event on the next rising edge of Bus Clock.
CSI_A[15:0]	Output	std_logic_vector (15 downto 0)	Presents the lower 16 bits of the 32-bit CSI address bus. Only 16 bits are presented because the 8051 only supports a 64K-byte (16-bit) address range. In most applications, these address signals are unused. Decoded and qualified addresses are presented on the CSI_WRSEL and CSI_RDSEL outputs.
CSI_DW[7:0]	Output	std_logic_vector (7 downto 0)	Presents the write data presented by the current CSI bus master. Valid data is qualified by the CSI_WRSEL output.

Port Name	Direction	Type	Description
CSI_WRSEL	Output	std_logic	Indicates that the current CSI bus master is performing a write transaction to the address range defined by this instance of the E5_CSI_SOCKET.
CSI_RDSEL	Output	std_logic	Indicates that the current CSI bus master is performing a read transaction to the address range defined by this instance of th E5_CSI_SOCKET.
CSI_WAITED	Output	std_logic	Indicates that a wait-state occurred during the last CSI bus cycle. This signal is typically only used by FIFO control logic.
CSI_EVENT	Output	std_logic	Indicates that a breakpoint event has occurred. This signal is typically not used in most applications.
CSI_BUSCLK	Output	std_logic	The CSI bus clock. Most transactions to and from the CSI bus should be synchronized to this clock signal.
SIM[127:0]	Bidirectional	std_logic (127 downto 0)	Triscend CSI bus simulation port. Connects to the Triscend bus functional model to simulate bus transactions.

## Generics

Port Name	Default	Type	Description
isAutoWaited	false	boolean	If true, then the Select will automatically assert one wait-state when the current bus master addresses this instance of the E5_CSI_SOCKET. Additional wait-states can be asserted using the CSI_WAITNEXT signal.
isExternal	false	boolean	If true, then the Selector will automatically send and receive CSI bus signals to the Memory Interface Unit (MIU). For example the CSI address bus will appear on the MIU's external address pins. Read and write data go to the MIU's external data pins. During write operations, the WE- pin is asserted. During read operations, the OE- pin is asserted. The address decoding and external timing is controlled by the E5_CSI_SOCKET function and associated logic.
isSFR	false	boolean	If true, then the decoded address actually resides within the 8051's internal Special Function Register (SFR) address space. Only single-byte address decodes are allowed. The isSFR generic also controls how the variable is declared in the 8051 header file created by FastChip Generate.

Port Name	Default	Type	Description
decode_pwr2	0	positive integer	<p>Controls the size of the address region decoded by this instance of the E5_CSI_SOCKET. The size, in bytes, is two raised to the specified power, as shown in the equation below. For example, the default value, zero, specifies to decode a single byte.</p> $SIZE = 2^{decode\_pwr2}$

## Example Usage

In this example, a simple binary up counter connects to the E5\_CSI\_SOCKET module. The counter connects to the CSI bus and any CSI bus master can be both read and write to the counter. The counter is preloaded whenever a bus master writes to the counter, other the counter increments.

Whenever the counter is read, the RDSEL signal is asserted. Whenever the register is written, the WRSEL signal is asserted. The CSI bus master writes data into the register by writing to the FastChip assigned address. The counter value is visible to other CSL logicon the 'Q' output. The CSI bus master can also read the counter value over the CSI bus.

```

library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity rw_counter is
    port (COUNTER_SYMBOLIC_ADDRESS : in std_logic;
          COUNT      : inout std_logic_vector(7 downto 0);
          RDSEL      : inout std_logic;
          WRSEL      : inout std_logic;
          RESET      : in    std_logic;
          CSIBUS     : inout std_logic_vector(127 downto 0)
    );
end rw_counter;

architecture structural of rw_counter is

    signal D      : std_logic_vector(7 downto 0);
    signal CLK    : std_logic;
    signal NC1, NC2 : std_logic;
    signal NC3    : std_logic_vector(15 downto 0);

    component e5_csi_socket is
        generic (
            -- isAutoWaited = true, then Selector asserts initial wait-state
            isAutoWaited : boolean := false;
            -- isExternal = true, then Selector enables MIU access through MIU
            isExternal   : boolean := false;
            -- isSFR = true, the Selector exports locations for 8051 SFR space
            isSFR        : boolean := false;
            -- Specifies number of bytes decoded, size = 2**decode_pwr2
            decode_pwr2  : positive := 0
        );

```

```

port (CSI_A      : out  std_logic_vector(15 downto 0);
      CSI_DW     : out  std_logic_vector(7  downto 0);
      CSI_DR     : in   std_logic_vector(7  downto 0);
      CSI_SYMBOLIC : in   std_logic;
      CSI_WRSEL  : out  std_logic;
      CSI_RDSEL  : inout std_logic;
      CSI_WAITED : out  std_logic;
      CSI_EVENT  : out  std_logic;
      CSI_WAITNEXT : in   std_logic;
      CSI_BREAK  : in   std_logic;
      CSI_BUSCLK : out  std_logic;
      SIM        : inout std_logic_vector(127 downto 0)
);

end component;

begin

CSI_SOCKET: e5_csi_socket
  generic map (
    isAutoWaited => false, -- zero wait-state
    isSFR        => true,  -- place counter in the 8051's SFR region
    decode_pwr2  => 2      -- set decoder to respond to a 4-byte
                          -- region, 2**2 (counter is only a
                          -- single byte, done for testing only)
  )

  port map (
    CSI_A      => NC3,
    CSI_SYMBOLIC => COUNTER_SYMBOLIC_ADDRESS,
    CSI_DW     => D,
    CSI_DR     => COUNT,
    CSI_WRSEL  => WRSEL,
    CSI_RDSEL  => RDSEL,
    CSI_WAITED => NC1,
    CSI_EVENT  => NC2,
    CSI_BREAK  => '0',
    CSI_WAITNEXT => '0',
    CSI_BUSCLK => CLK,
    SIM        => CSIBUS
  );

COUNTER: process (CLK, RESET, WRSEL)
begin
  if (RESET = '1') then
    COUNT <= "00000011"; -- reload counter with 0x3
                       -- when reset or power-up
  elsif (rising_edge(CLK)) then
    if (WRSEL = '1') then
      COUNT <= D; -- load with CSI data on write
    else
      COUNT <= COUNT + 1; -- otherwise increment
    end if;
  end if;
end process;

end structural;

```

## Post-Synthesis View

The following is the Synplicity HDL Analyst view of the RTL-level code.

