MAX 10 ADC with VHDL / Quartus Prime Lite Code Optimisation

Asked 7 years, 7 months ago Modified 7 years, 7 months ago Viewed 2k times

0

First of all I have to say that I am a complette beginner in VHDL so if its a really stupid question I want to apologize in advance. Im trying to get the ADC soft IP to work. I want to use just the ADC so no FIFO or anything else. Therefore I generated the IP core with the qsys file and included it into my project. I also activated channel 8 with prescaler. Im trying to read the value from a variable resistor that is connnected to channel 8 and print out the 5 most significant bits with the led. The case statement should create the pattern that is needed to activate the adc, as it is given in MAX 10 ADC guide.

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



entity main is
    port ( CLK_50   : in  std_logic;
           LEDR  : out std_logic_vector(4 downto 0));
end;


architecture behave of main is

    signal Cnt : integer := 0;
    signal pCnt : integer := 0;

    signal lock : std_logic;
    signal CLK_10 : std_logic;
    signal CLK_1 : std_logic;
    signal set : std_logic ;

    signal RESET : std_logic ;

    signal CMDVal : std_logic;
    signal CMDCH : std_logic_vector (4 downto 0);
    signal CMDSOP : std_logic;
    signal CMDEOP : std_logic;
    signal CMDRDY : std_logic;
    signal RESVal : std_logic;
    signal RESCH : std_logic_vector (4 downto 0);
    signal RESData : std_logic_vector (11 downto 0);
    signal RESSOP : std_logic;
    signal RESEOP : std_logic;

    component myadc is
        port (
            clock_clk              : in  std_logic                     := 'X';
-- clk
            reset_sink_reset_n     : in  std_logic                     := 'X';
-- reset_n
            adc_pll_clock_clk      : in  std_logic                     := 'X';
-- clk
            adc_pll_locked_export  : in  std_logic                     := 'X';
-- export
            command_valid          : in  std_logic                     := 'X';
-- valid
            command_channel        : in  std_logic_vector(4 downto 0)  := (others =>
'X'); -- channel
```

```
            command_startofpacket  : in  std_logic                     := 'X';
-- startofpacket
            command_endofpacket    : in  std_logic                     := 'X';
-- endofpacket
            command_ready          : out std_logic;
-- ready
            response_valid         : out std_logic;
-- valid
            response_channel       : out std_logic_vector(4 downto 0);
-- channel
            response_data          : out std_logic_vector(11 downto 0);
-- data
            response_startofpacket : out std_logic;
-- startofpacket
            response_endofpacket   : out std_logic
-- endofpacket
        );
    end component myadc;

begin

    CMDCH <= "01000";
    RESET <= '0';
    set <= '1';

    mPLL : entity work.pll
        port map(
        areset => set,
        inclk0 => CLK_50,
        c0 => CLK_10,
        c1 => CLK_1,
        locked => lock
    );


    mADC : component myadc
        port map (
            clock_clk              => CLK_50,                  --          clock.clk
            reset_sink_reset_n     => RESET,                     --
reset_sink.reset_n
            adc_pll_clock_clk      => CLK_10,                   --  adc_pll_clock.clk
            adc_pll_locked_export  => lock,                          --
adc_pll_locked.export
            command_valid          => CMDVal,                    --
command.valid
            command_channel        => CMDCH,                    --
.channel
            command_startofpacket  => CMDSOP,                    --
.startofpacket
            command_endofpacket    => CMDEOP,                   --
.endofpacket
            command_ready          => CMDRDY,                  --                .ready
            response_valid         => RESVal,                   --       response.valid
            response_channel       => RESCH,                     --
.channel
            response_data          => RESData,             --                .data
            response_startofpacket => RESSOP,                    --
.startofpacket
            response_endofpacket   => RESEOP                    --
.endofpacket
        );
```

```
process
begin

    wait until rising_edge(CLK_50);

    pCnt <= pCnt + 1;

    case pCnt is
        when 1 => CMDSOP <= '1';
                     CMDVal <= '1';
        when 114 => CMDRDY <= '1';
        when 115 => CMDSOP <= '0';
                        CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 214 => CMDRDY <= '1';
        when 215 => CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 314 => CMDRDY <= '1';
        when 315 => CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 414 => CMDRDY <= '1';
        when 415 => CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 514 => CMDRDY <= '1';
        when 515 => CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 614 => CMDRDY <= '1';
        when 615 => CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 714 => CMDRDY <= '1';
        when 715 => CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 814 => CMDRDY <= '1';
        when 815 => CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 914 => CMDRDY <= '1';
        when 915 => CMDRDY <= '0';
                        LEDR <= RESData(11 downto 7);
        when 1014 => CMDRDY <= '1';
        when 1015 => CMDRDY <= '0';
                         LEDR <= RESData(11 downto 7);
        when 1114 => CMDRDY <= '1';
        when 1115 => CMDRDY <= '0';
                         CMDEOP <= '1';
        when 1116 => CMDEOP <= '0';
                         CMDVal <= '0';


        when 2000 => pCnt <= 0;
        when others => Cnt <= pCnt ;
    end case;
end process;

end;
```

However while compiling Quartus always removes all of my code. So in the end it pulls the LED's to GND and neither uses the ADC or the PLL. If anyone has an idea I would be really thankfull if you could tell me what exaclty im doing wrong.

Best regards.

Edit: I wasnt clear enough with describing the problem I have. It does synthetzise correctly however it thinks that the pll is not necessary and so removes it, leaving the adc ip core without the clock and so removeing it as well. The errors are:

```
Warning (14284): Synthesized away the following node(s):
    Warning (14285): Synthesized away the following RAM node(s):
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
Warning (14284): Synthesized away the following node(s):
    Warning (14285): Synthesized away the following RAM node(s):
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
        Warning (14320): Synthesized away node
"madc:myadc|madc_modular_adc_0:modular_adc_0|altera_modular_adc_control:control_internal|
Warning (14284): Synthesized away the following node(s):
    Warning (14285): Synthesized away the following PLL node(s):
        Warning (14320): Synthesized away node
"mpll:myPLL|altpll:altpll_component|mpll_altpll:auto_generated|wire_pll1_clk[0]"
```

vhdl **fpga** **adc** **quartus**

Share Improve this question Follow

edited Feb 20, 2017 at 19:16

asked Feb 20, 2017 at 14:47
nuclear
105 7

When your logic gets eaten during optimization and your outputs ties to a rail it's usually a sign of a logic design error or tool usage error. in this case centered on ResData from the myadc used to drive LEDR values. In addition to a potential logic problem with myadc it can be unbound in elaboration resulting in ResData not being driven and the entire remainder of the design being gate eaten (LEDR the only output). You need to examine your console output (or logs) for warnings and errors during synthesis. – user1155120 Feb 20, 2017 at 19:06

Yeah i realized that the problem is that the output is not driven, so I tried to save the data of the adc in a seperate std_logic_vector and drive the output with every clockcycle, however that didnt changed the problem. The downside is that I dont know how to change anything inside the adc logic becaue its a altera ip core. I could only find tutorials for the adc control with sample storrage. As far as i understood the altera documentation the logic to drive the adc should be correct but sadly i cant confirm that it is. – nuclear Feb 20, 2017 at 19:24

It doesn't seem possible for your readers to peer into a black box and tell you what's wrong without having been similarly bitten. You appear to be dealing with IP core configuration during generation. You've said your aren't using the FIFO, there's a PLL showing in the myadc component declaration. – user1155120 Feb 20, 2017 at 19:42

The first question is, did it work correctly in simulation? Most likely, when you get it to simulate as expected, the mistake that's allowing synthesis to trim all the logic out will have gone. – user1818839 Feb 20, 2017 at 20:59

1 Answer

Sorted by: Highest score (default)

1

In VHDL (and basically in all hardware description languages), you have to keep in mind that your code must be synthetizable: it has to describe hardware components available in your programmable component. This is not the case in your process.

The following line : `wait until rising_edge(CLK_50);` can't be synthesized because of the `wait` statement.

To create a sequential process you need this :

```
my_seq_proc : process (clk, rst)
begin
     if (rst = '1') then
        ... -- reset your signals
     elsif (rising_edge(clk)) then
        ... -- what you need to do
     end if;
end process;
```

Note that you are not obliged to use a reset signal. Also, note that you need a sensitivity list in the process declaration `(clk, rst)` with your clock (clk_50) and your possible reset signal.

I haven't checked if there is another mistake. You should try to do this first.

Share Improve this answer Follow

answered Feb 20, 2017 at 15:54
A. Kieffer
372 2 13

There's a difference between recommended coding styles and supported styles for inferring registers. Recommended styles are found in a Volume 1 Design and Synthesis Handbook. Supported styles are found in the withdrawn IEEE Std 1076.6-2004 IEEE Standard for VHDL Register Transfer Level (RTL) Synthesis. In this particular case 6.1.3.2 Edge-sensitive storage using a single wait statement, matching the OP's usage. Altera's synthesis tool can be expected to be compliant with the withdrawn IEEE standard. – user1155120 Feb 20, 2017 at 18:51

1 as far as i know wait until rising_edge(clk) is a perfectly valid option as long as you dont want to have a reset. Anyways it did synthetizise before without problems. However while sythetizing it removed the pll for some reason and so of course all of the logic is not working at all . But i also changed it how you recommended it but it still doesnt work. – nuclear Feb 20, 2017 at 19:10

Didn't know that `wait until` was supported by Altera for synthesize. My apologies then. – A. Kieffer Feb 21, 2017 at 8:20

Wait statements in general, only one. At the beginning or end of a process (a sensitivity list has an implicit wait statement `wait on` at the end of a process, 11.3, 10.2). What types of VHDL wait constructs does Quartus II synthesis support? – user1155120 Feb 21, 2017 at 18:55