*Arid Zone Journal of Engineering, Technology and Environment. August, 2013; Vol. 9: 17-26*
*Copyright© Faculty of Engineering, University of Maiduguri, Nigeria.*
Print ISSN: 1596-2490, Electronic ISSN: 2545-5818
www.azojete.com.ng

# DESIGN AND IMPLEMENTATION OF MOD-6 SYNCHRONOUS COUNTER USING VHDL

**Dibal, P.Y.**
(*Department of Computer Engineering, University of Maiduguri, Maiduguri, Nigeria*)
**e-mail address:** yoksa77@gmail.com

## Abstract

This paper deals with the design of a MOD-6 synchronous counter using VHDL (VHSIC Hardware Description Language). The VHSIC stands for Very High Speed Integrated Circuit. Using this approach, the behaviour of the counter is the most important aspect of the design. In the first section, the paper introduced counters in general, and their areas of specialization, like frequency synthesizers. The synchronous counter was then introduced, stating the behaviour of the flip-flops that make the counter. The modulus of a counter was defined. In the second section, the Xilinx ISE (Integrated Simulation Environment) and the ISIM (Integrated Simulator) were presented and briefly described with their respective snapshots. The structure of a typical VHDL code was presented, which included LIBRARY, ENTITY, and ARCHITECTURE. Each of these structures was then briefly described. The main work in this paper was then presented. The count sequence steps were stated as $0 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 6 \rightarrow 4$. VHDL was used to model the counter to count through six steps, outputting count values according to desired steps. The hardware implementation of the design was presented, where the implementation process was described, with a supporting diagram, followed by the floor-planning technique, in which the PORTS described in the VHDL design were assigned to the physical pins of the XC3S1000 FPGA (Field Programmable Gate Array) chip. The final steps of the hardware implementation process were then presented. These include bitstream generation and download to target device. The third section of the paper presented the results obtained. Simulation/timing results of the design were presented, showing the output of the counter at each state with respect to the clock signal. The result of the synthesis of the design was presented, which showed the FPGA area with the exact location of the pins on the FPGA chip. Finally, the fourth section presented the conclusion arrived at, in respect of the design that was carried out.

Keywords: VHDL, MOD-6 Synchronous counter, FPGA, Xilinx ISE, XC3S1000, Spartan-3

## 1. Introduction

In almost all digital systems, counters are extensively used in areas such as frequency synthesizers, analog-to-digital converters and circuits used in communication systems (Gifty and Kashwan, 2012). A synchronous counter in particular, is the counter type in which all the flip-flops in the counter change their state in synchronism with the input clock signal (Anil, 2007). The clock signal in a synchronous counter is simultaneously applied to all the clock input of the flip-flops (Rabiul and Jabayer, 2012).

The modulus (MOD number) of a counter according to Anil (2007) is defined as the number of different logic states the counter goes through before it comes back to the initial state in order to repeat the count sequence. Also, from (Anil, 2007), an n-bit counter is said to have a modulus of $2^n$, when it goes through all its natural state, without skipping any of the states. Obviously, such a counter is said to have a modulus that is an integral power of 2. The information provided by the modulus number of a counter can be used to determine the

number of flip-flops required to build such a counter. However, since this paper focuses on the application of VHDL (VHSIC Hardware Description Language) to the design of a MOD-6 counter, the use of the MOD number of a counter to determine the number of flip-flops required will not be looked at in details.

The objective of this paper is to design a highly and easily modifiable MOD-6 synchronous counter using VHDL in the Xilinx ISE. The choice of VHDL is because circuits designed in hardware description languages like VHDL are highly portable in the sense that the count sequence of the counter can be easily changed from the source code to suite different applications of the same counter.

## 2. Materials and Methods

The VHDL was used to design a MOD-6 synchronous counter in the Xilinx ISE. Drawing from the definition of the MOD number of a counter, a MOD-6 counter goes through six logic states before going back to the initial state to repeat the count sequence, as shown in Figure 1:

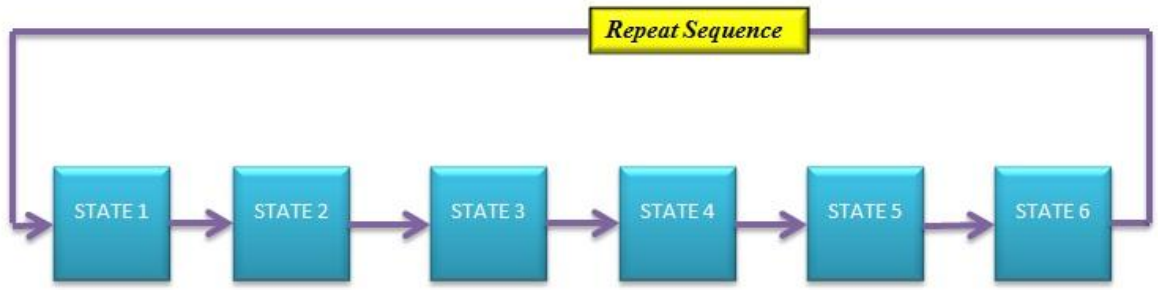


Figure 1: Behaviour of MOD-6 counter

The behaviour of the circuit is such that the circuit starts from state 1 (which is actually binary 0), and then moves to the next state at the next appropriate clock signal. For the synchronous counter design presented in this paper, the counter will be able to go through the following steps: $\mathbf{0 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 6 \rightarrow 4.}$

### 2.1 Xilinx ISE

The Xilinx ISE (Integrated Simulation Environment) provides the environment for VHDL, in which the design of the synchronous counter is usually carried out using the text based approach of hardware description (Xilinx , 2011). The environment allows for checking of syntax error in the model; it also allows for simulation using the ISIM (Integrated Simulator) plug-in, where the design of the model can be verified using waveform analysis. A snapshot of the Xilinx ISE and the ISIM are shown in Figures 2 and 3

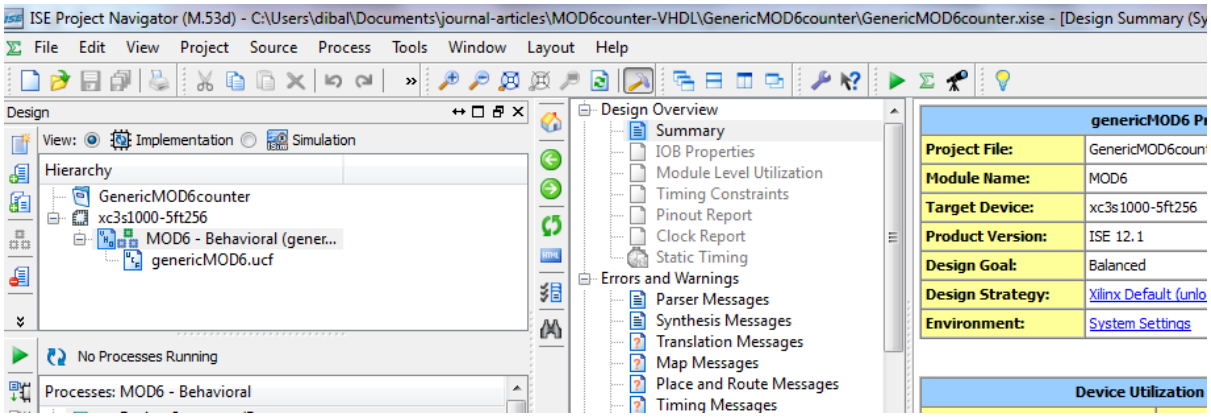


Figure 2: Xilinx ISE

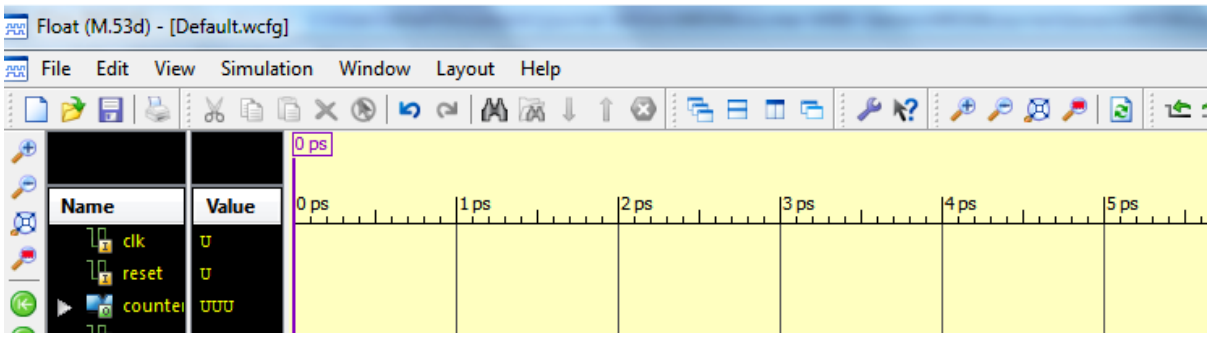


Figure 3: ISIM Plug-in

## 2.2 Structure of VHDL code

A typical VHDL code is made of the following sections (Pedroni, 2004):

- LIBRARY
- ENTITY
- ARCHITECTURE

The Library is where commonly used codes are placed; and by this means, they become available and reusable in other designs. The library structure is shown in Figure 4.

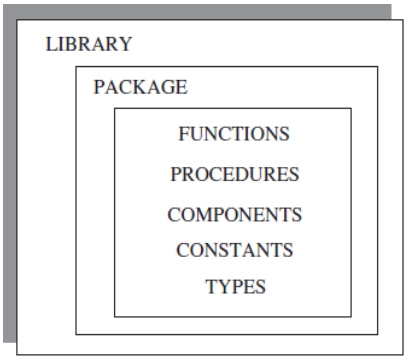


Figure 4: VHDL library structure

The Entity is where the design specifications of all input and output pins (PORTS) are made. That is, it is here the interface of the circuit to other circuit components or the outside world is made (Pedroni, 2004). The Architecture is where the description of how the circuit should behave or function is implemented. It has two parts: a declarative part, which is optional, where signals and constants are declared; and a code part, which describes the actual behaviour of the circuit (Pedroni, 2004)

### *2.3 Design implementation*

The synchronous counter will be designed to go through the following steps as stated earlier:

$$\mathbf{0 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 6 \rightarrow 4}$$

The implementation of the design is achieved by modelling the counter using VHDL, as shown in Figure 5.

From the code listing for the design implementation, it can be seen that the model has two processes. The first process outputs the counter value when the clock is active high, as the count goes through the sequence of count by moving from one state to the next state. The second process moves the counter from the present state to the next state by transferring the content of the user input to the next state variable, and pointing the counter to the next state to go to. The VHDL design shown in (Figure 5) has three parts: the library part, the entity part, and the architecture part.

In the Library part of the design, the IEEE library was declared, which contains functions, procedures as defined by the IEEE standard. Under the ieee library, the design will use std_logic_1164 package which specifies a multilevel logic system design. In the Entity part, the design will have only output ports because the synchronous counter designed have no data input. The output PORTS are ***clock, reset,*** and ***counterOUT*** (defined as 3-bit wide).

The Architecture of this design has both declarative and code parts. In the declarative part, a user-defined data type called ***sequence_step*** is declared. It stores the six states the counter will pass through. Next, two signals, namely ***sequence_reg*** and ***sequence_next,*** both them being of the ***sequence_step*** step type, are declared***.*** They store the present and next sequence of the counter. Finally, a next state signal, which is a type of ***standard_logic_vector*** is declared. It stores the next state of the count sequence.

The code part of the Architecture has two processes. The first process monitors the reset input and the clock input signals. Members of the sensitivity list of this process are ‘clk’, and ‘reset’. The second process has a CASE structure, which handles each of the possible states of the count sequence. Members of the sensitivity list are ***sequence_reg,*** and ***nextstate.***

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MOD6 is
    Port ( clk, reset : in STD_LOGIC;
           counterOUT : out  STD_LOGIC_VECTOR (2 downto 0));
end MOD6;

architecture Behavioral of MOD6 is
type sequence_step is (one, two, three, four, five, six);
signal sequence_reg, sequence_next : sequence_step;
signal nextstate : STD_LOGIC_VECTOR (2 downto 0);

begin

      --State Register
      Process(clk, reset)
      begin
            if (reset = '1') then
                  sequence_reg <= one;
            elsif (clk'event and clk = '1') then
                  sequence_reg <= sequence_next;
                  counterOUT <= nextstate;
            end if;
      end process;

      --Next State
      Process(sequence_reg, nextstate)
      begin
            sequence_next <= sequence_reg;
            case sequence_reg is
                  when one =>
                        nextstate <= "000";
                        sequence_next <= two;
                  when two =>
                        nextstate <= "011";
                        sequence_next <=three;
                  when three =>
                        nextstate <= "101";
                        sequence_next <= four;
                  when four =>
                        nextstate <= "111";
                        sequence_next <= five;
                  when five =>
```

Figure 5: Modelling of MOD-6 synchronous counter

### 2.4 Hardware Implementation

The implementation of this design was carried out on an FPGA (field programmable gate array) chip, specifically on the XC3S1000 FPGA. The procedure for implementing the design on the XC3S1000 FPGA chip is presented in Figure 6:

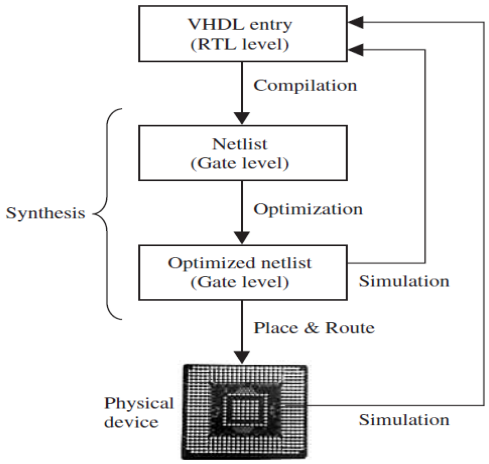


Figure 6: Hardware Implementation process

The first step in the hardware implementation is to floor-map the model to the target hardware. The floor-planning allows the connection of the inputs and outputs of the top level entity to the pins of the XC3S1000 FPGA chip. (Xilinx Spartan-3 FPGA data sheet, 2012). The general purpose I/O (GPIO) pins wereused, and the output wasmapped to a GPIO pin in Bank 0, the **clk** input was mapped to Bank 0, and the **reset** was mapped to Bank 3. Details of the FPGA pins and their corresponding Bank numbers can be found in the Xilinx Spartan-3 FPGA data sheet. The mapping was done according to Table 1 below:

Table 1: Floor-planning table for FT256 pin assignments

| Port Name | Function | XC3S1000 Pin Assignment | BANK NUMBER | FT256 Pin Number |
|---|---|---|---|---|
| clk | input | I/O L32P_0 GCLK6 | 0 | A8 |
| reset | input | I/O L40P_3 | 3 | K16 |
| counterOUT[0] | output | I/O L25P_0 | 0 | B5 |
| counterOUT[1] | output | I/O L28P_0 | 0 | B6 |
| counterOUT[2] | output | I/O L30P_0 | 0 | B7 |

The FPGA chip is usually divided into banks (sections). For the XC3S100, it has eight banks, from bank 0 to bank 7. For this design, four of the PORTS wereon bank 0, while the reset PORT wasin bank 3; the reason being that the pins of an FPGA are usually very close to each, and putting a sensitive signal like the reset signal in the same bank with other signal is usually not advisable. The FT256 pin number is the physical address of each pin on the FPGA chip.

The implementation of the floor-planning (Table 1) is achieved using I/O Planning (Plan Ahead) – Pre-Synthesis module in the Xilinx ISE as shown in Figure 7.

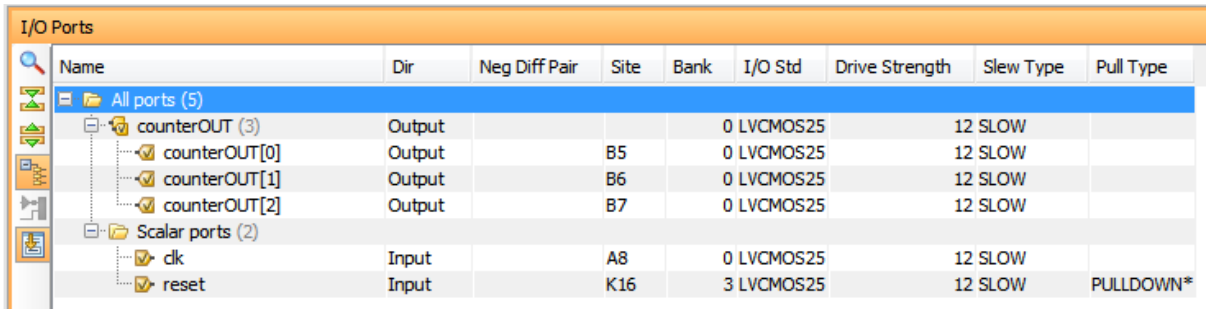


Figure 7: Design floor-mapping

After the floor-mapping, the next step is mapping, placing, and routing of the design in the Xilinx ISE; this is then followed by the bit stream generation, which condenses the whole design into ***0s*** and ***1s*** by generating a bitstream file which contains the complete design represented in binary format. The generated bit stream file for the design is shown in Figure 8 below:

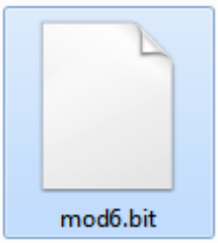


Figure 8: Bitstream generated file

The final step in the hardware implementation is to download design on the FPGA chip. This is achieved using the ISE iMPACT (a tool featuring batch and GUI operations, which allows a designer to perform device configuration and file generation) interface, where the generated bitstream file is attached to the FPGA chip as shown in Figure 9.

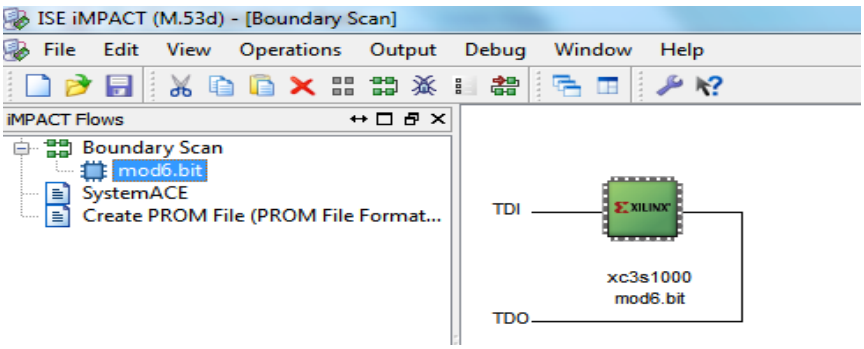


Figure 9: Successfully downloaded bitstream file to target device

## 3. Results and Discussion

Having successfully designed the synchronous counter, and downloaded it to the target hardware, a simulation was carried out, and the following result (Figure 10) was obtained.

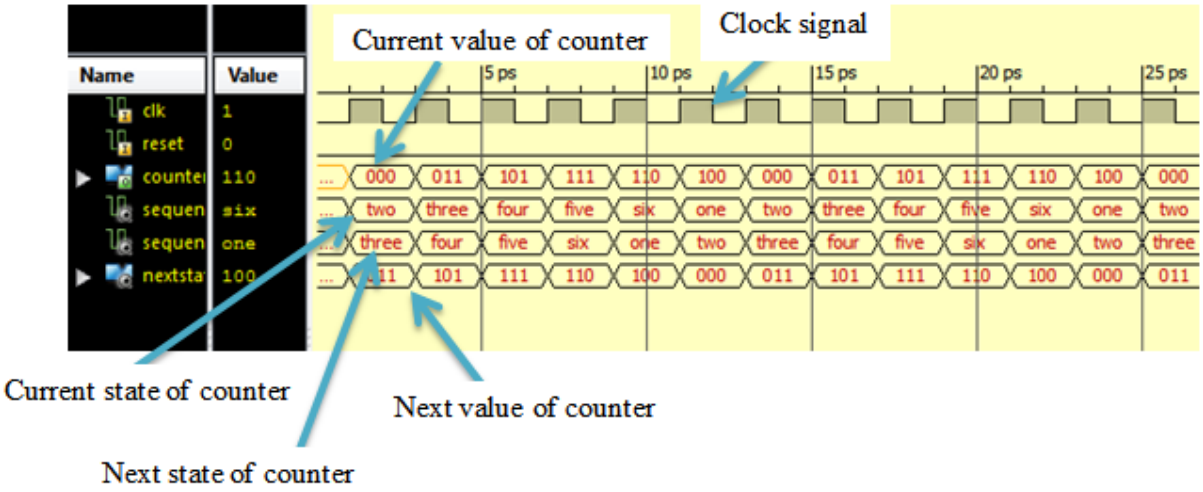


Figure 10: Simulation / Timing result from 0 to 25ps

As seen in figure 10, the counter moves from state1 to state2, to state3, up to state6, exactly as described in Figure 1. In each of these states, the counter outputs the values stated in the design implementation, i.e. $\mathbf{0 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 6 \rightarrow 4.}$ Elaborating further, in state1 the counter outputs 000, in state2 it outputs 011, and in state3 it outputs 101, and so on, till it gets to state6 where it outputs 100. However, a closer look at Figure 10 shows a shift between the current state and output value. For example, instead of the counter outputting 000 in state1, it outputs 000 in state 2. This is because, the lag that is occurring between the current output value and the current state is caused by the time delay it takes the clock signal to attain the first rising edge. This design is a positive-edge clock triggered synchronous counter design.

Figure 10 also shows that the count sequence is repeated twice from 0 to 25ps (pico seconds). The timing information for this design from 0 to 50ps is shown in Figure 11.

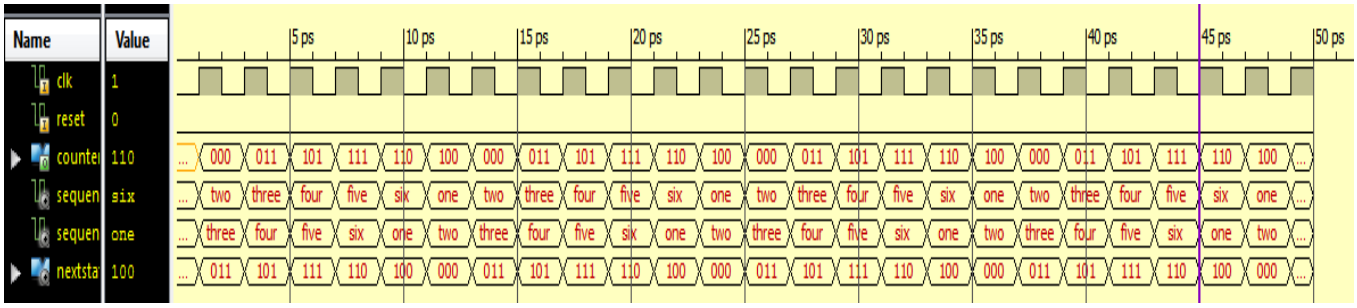


Figure 11: Simulation / Timing result from 0 to 50ps

The synthesis of the design shows the area on the FPGA chip where the mapped-pins are located, based on the floor-mapping presented in Table 1. The locations of the pins on the FPGA are shown in Figure 12.

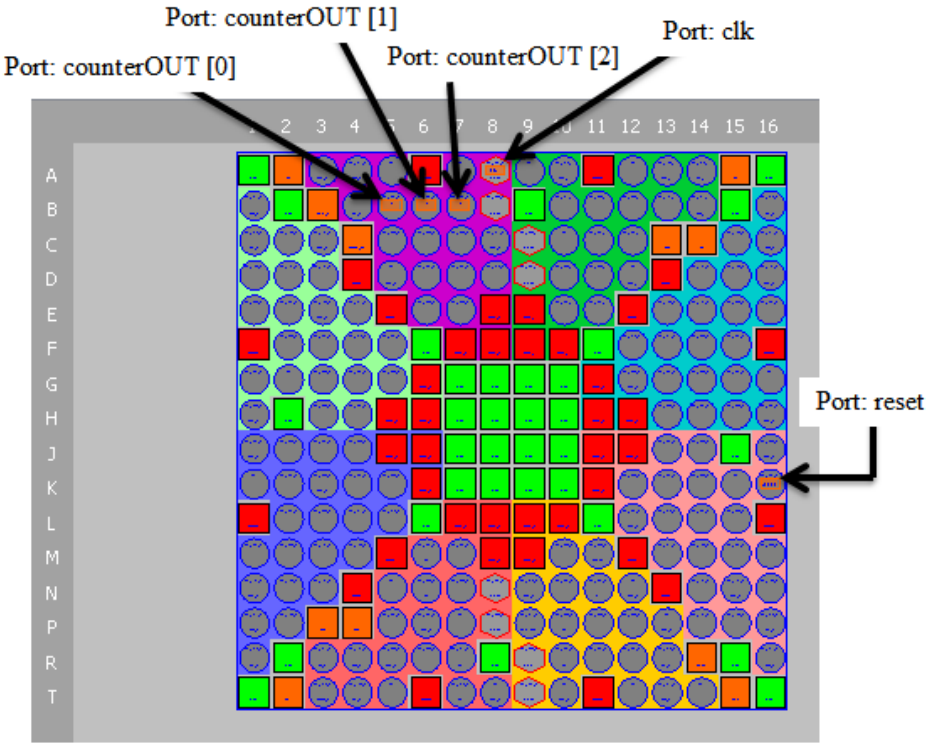


Figure 12: Synthesis area information showing pin locations

## 4. Conclusions

In this paper, the design of a MOD-6 counter was successfully carried out using the Xilinx ISE and the XC3S1000 FPGA chip. The behaviour of the counter was modelled using VHDL in the Xilinx ISE environment, and then verified for accuracy using the ISIM plug-in. The design was then implemented on the FPGA chip, using the technique of floor-planning and bit-stream generation. After generating the bitstream file, the design was then downloaded to the target device using the iMPACT interface.

## References

Anil, KM. 2007. Digital Electronics Principles: Devices and Applications. Sussex: John Wiley & Sons.

Gifty, JB. and Kashwan, KR. 2012. Design and simulation of scalable fast parallel counter. International Journal of Applied Information Systems, **1**(9): 28-33.

Pedroni, VA. 2004. Circuit Design with VHDL. Massachusetts: MIT Press.

Rabiul, SM. and Jobayer, HM. 2012. Gate level design of a digital clock with-asynchronous-synchronous logic. Global Journal of Researches in Engineering Electrical and Electronics Engineering, **12**(4): 16-22.

Xilinx Inc. 2011. *Xilinx ISE In-Depth Tutorial.* [e-book] Colorado: University of Colorado Press. Available at: <http://download.xilinx.com/ISE/ise_indepth_tutorial_ug695.pdf> [Accessed 20 April 2013].

Xilinx Inc. 2012. *Spartan-3 FPGA Family data sheet.* [e-book] Available at: <http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf> [Accessed 22 April 2013].