**Description of Alarm Clock**  
  
  
Upon examining the code, the first point of interest to look at is the alarm\_top file. This file essentially explains the operations of all the sub-modules and explains the transfer of signals needed. I have provided quick illustrations of block-diagrams for the alarm-clock. There are certain modules used from the Xilinx library such as counters whose code cannot be included within the repository. They are generated from Xilinx in Vivado.

**Alarm\_Top**  
Firstly a general view of the alarm\_top module is portrayed. It must be noted that the board used a 7 segment display to display all numbers. Thus, any outputted values needed to be converted to a 7-segment value which is the purpose of the LUT (convert decimal to binary) and the number modules (convert the binary number to respective 7-segments). For example, the number 8 would be a “1000” in binary. This 8 would have all 7 segments off, (logic-low).

The alarm essentially uses two loader modules when setting the real time and the alarm time. The alarm time loader stores the alarm time, the other regular loader feeds into a time\_top module which hold the time and continuously increments it based on seconds and minutes and hours. The comparator compares the real time to the alarm time and determines when necessary to set off the alarm.   
The mux\_driver displays which info is displayed depending on what button or switch are held (ie: when changing the alarm time with the enable\_A button, the alarm time will be displayed). The big\_mux flashes the anodes of the display at a fast rate which cannot be tracked by the human eye.

**Wack-a-mole game**  
The alarm also sends the signal to the Wack amole games which sets off the game. This game uses a state machine. The state idle goes into the switch state when hitting a start button. A random number is generated which corresponds to a light or “mole”. The user is given a small amount of time to match a corresponding button to the light which goes off. This successfully scores a point. When the threshold is reached for points to score, the state machine goes back to idle and turns the game off. The individual has successfully turned off the alarm after completing the game.   
  
**Timer\_top**  
The timer\_top module performs much of the computation for the alarm clock. It uses a set of Xilinx IP counters which hold the counts for the seconds, minutes and hours. The seconds works off a 1Hz clock, when this number reaches 60, it resets and also goes to increment the minute counter. This minute counter does the same for the hour counter when it reaches 60. A time-zone module and decoder will fix the hours based on a selected time zone switch picked. Certain cities include Florence and Dubai. These time offsets are based on the time entered at the real time (eastern time). This also adjusts the time based on if 24 hour or 12 hour format is specified. A light is outputted when 12 hour format is selected for a PM light.

**Calendar**

The calendar expands the idea of the hour counter to days and months. A month fixer module is used to reset the days and increment the months when the end of the month is reached, for example when October 31st, 11:59 PM and 60 seconds is seen. The LUT for the calendar converts the months into proper 7-segment display values needed.







