g04\_FSM\_Controller

# Description of System’s Features

The purpose of the system designed in this lab was to provide user control of the music box system through the use of a finite state machine. The state machine was comprised of three states: stopped, played and paused. Playback was inactive in the stopped and paused state, while in the played state the song information was read from the LUT ROM, fed to the flash reader and finally played through the audio interface. See for the state diagram of the FSM.

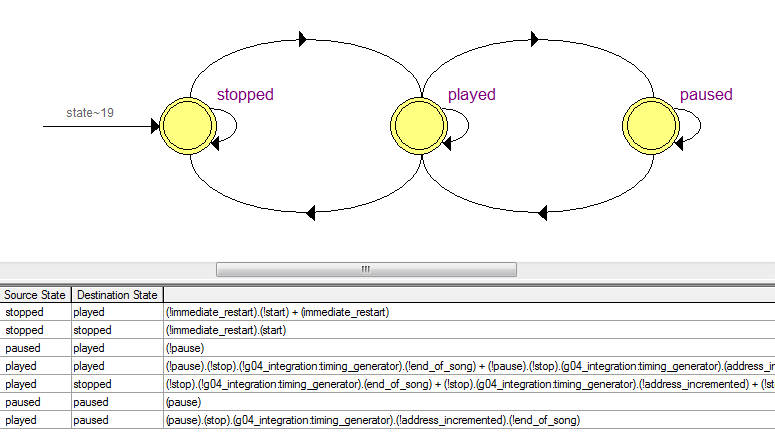


Figure Mealy-Type State Diagram for Controller

# Block Diagram of Entire System

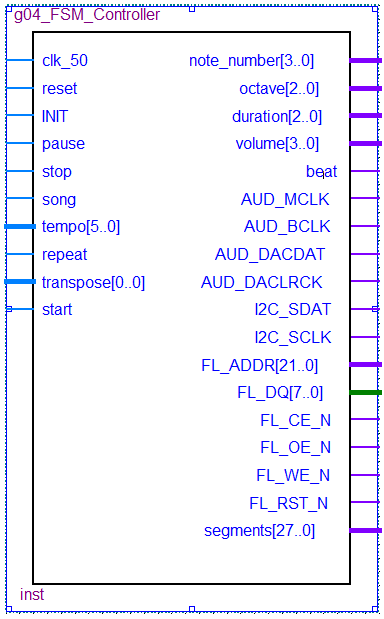


Figure Pinout Symbol of Controller

# Description of System

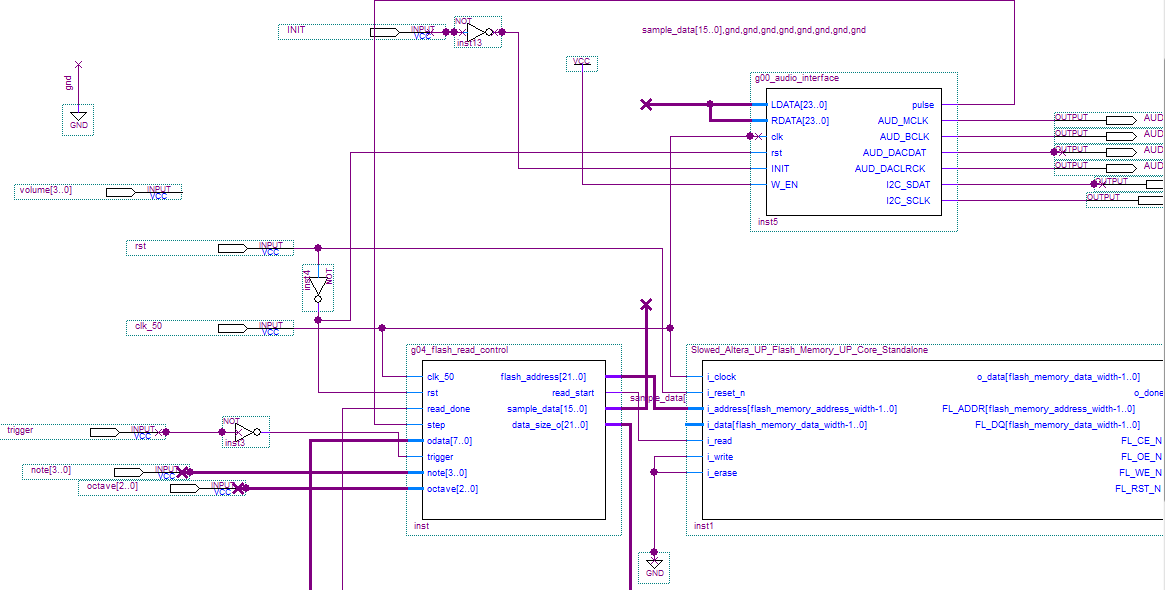


Figure Block Diagram of Audio and Flash Interface

The overall music box control system made use of multiple components, including both hardware interface components and user control logic.

On the hardware level, there are modules that communicate with the flash memory and the audio interface. The flash read control module takes as inputs the music data comprised of note number, octave and volume for each note as well as a trigger signal, which tells the system when to begin a new note. The trigger signal is generated by the note timer component which in turn depends on the output of the tempo unit. The outputs of the flash read control are fed to the flash memory which retrieves the guitar strum, and to the audio interface which plays the note through the Altera board’s audio output. See Figure 3 for the block diagram depicting the communication between these modules.

The user control of the system handles user input to generate the note information and trigger signal needed by the flash read control. First the timing of the song must be established. The user inputs the desired tempo in beats per minute on the slide switches which is then processed by the tempo circuit to output both a beat signal and a tempo\_enable signal (a subdivision of the beat). The tempo\_enable is sent to the note timer along with the note duration for a specific note which then produces the aforementioned trigger signal. At the same time, the note information needs to be retrieved. The note information is stored in a .mif file where each 16 bit entry contains all the data for a single note. This includes the note number, note duration, triplet bit, loudness, octave, and a final bit indicating the termination of the song. This structure requires a separate .mif file for each song. These .mif files are accessed via a look-up-table module provided by the LPM library. Iteration over the data entries is directly controlled by the finite state machine.

In order for all the above components to function synchronously a robust controller is needed. This is implemented in the form of a Mealy finite state machine, in which the output is dependent on both the inputs to the system and its current state. Our controller has three states. The stopped state is the initial state of the system. In this state the address signal is set to 0, pointing to the first entry of the appropriate LUT, and all sound playback is suppressed. Jumping to the played state is achieved by either a user-activated start signal or a high repeat signal. Once in the played state, playback can begin! For each pulse of the trigger in this state, the LUT address is incremented by one and note data is read and fed to the hardware components. There are three ways to leave this state. If the stop button is pressed by the user, or the end of song marker is reached the state will switch back to stopped. If the pause slide switch is activated, the state will change to paused. The paused state is an idle state wherein playback halts and the address signal is no longer incremented. The state will revert to played when the pause switch goes low. See Figure 1 for a state diagram of the FSM and Figure 4 for a block diagram of the FSM.

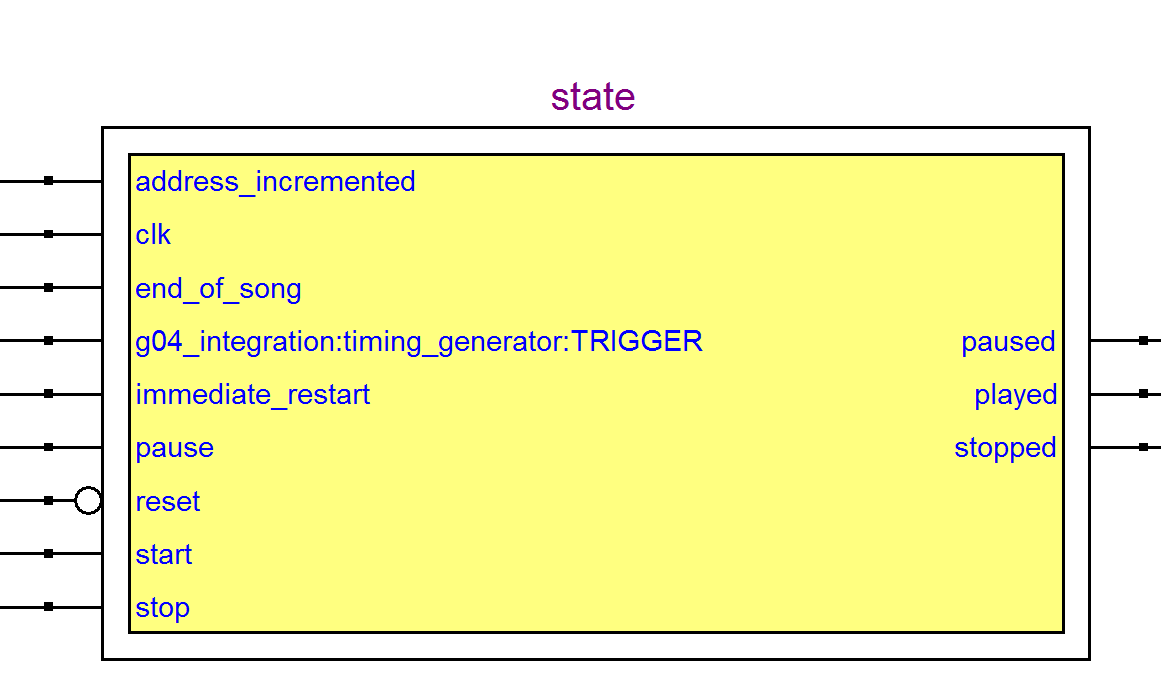


Figure Block diagram of controller

# Description of User Interface

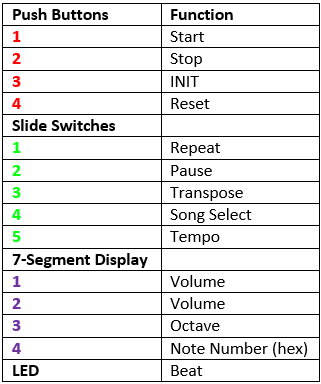
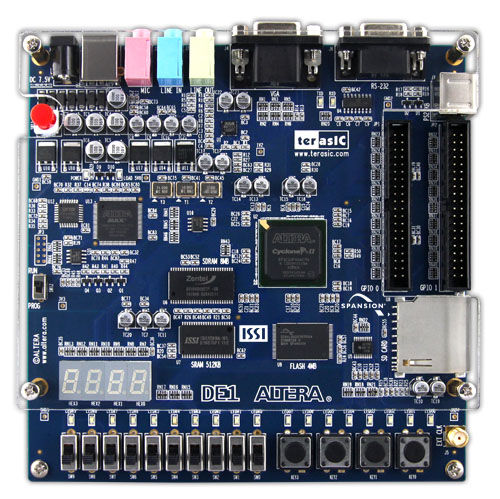
A user communicates to the system through the use of push buttons, slide switches, 7-segment displays, and an LED. See Figure 4 for the detailed user guide.

Figure User Interface Guide



**1**

**2**

**3**

**4**

**1**

**2**

**3**

**4**

**1**

**2**

**3**

**4**

**5**

# Testing and Simulation

The music box controller system is composed of a large number of interacting components. As such, it was important to thoroughly test each feature individually to ensure it was behaving correctly. Only once the individual features were confirmed to be working satisfactorily, a complete integration test was performed to verify the communication between the elements of the system.

## Feature Testing

### Display Features

Note information for each note is displayed on the board’s 7 segment displays. See for details of the mapping of each item to be displayed. In order to test that this feature worked correctly, we programmed the device and verified the numbers on the 7 segment displays matched the corresponding entries in the .mif files for our songs.

Another display feature is the beat LED. The LED would flash once per beat, at a rate input by the user on the slide switches. To test this, we mapped the slide switches to our module’s tempo input and again programmed the device. We ensured that any change in the tempo was reflected accurately in the flashing beat LED.

### Control Features

The control of the music box is a key feature of the system. The user has control over the following:

* Starting
* Stopping
* Pausing
* Choosing repeat mode or one-shot mode
* Transposing down one octave
* Selecting one of two songs
* Tempo

See for the mapping onto the board’s inputs.

Each control feature was tested by assigning the appropriate pins, programming the board and using the previously verified display features of the system, confirming that the playback was responding correctly.

# FPGA resource utilization

Resource utilisation on the board is summarised in Figure 5. The quantities of logic elements, combinational functions, and other resources are listed.

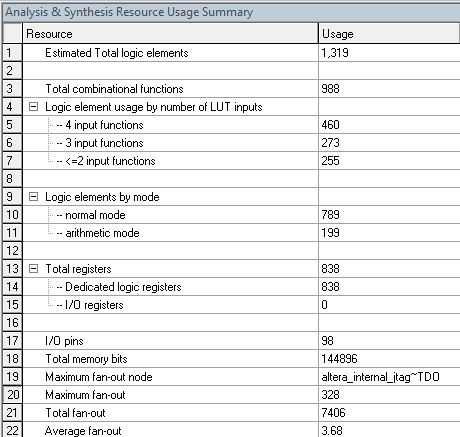


Figure Resource Usage Summary

# Conclusion

## Design Struggles

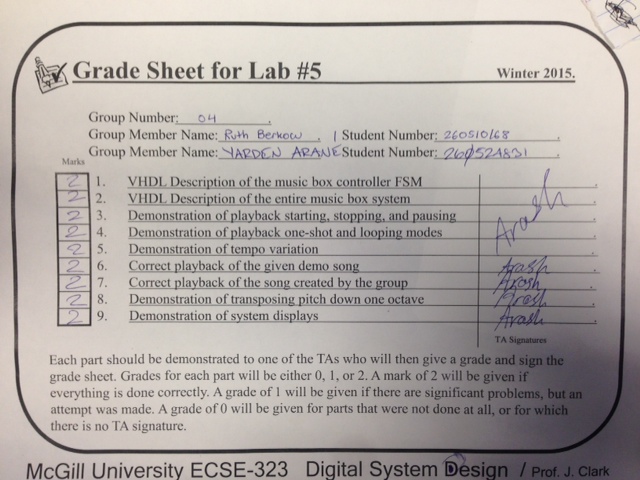
After designing individual components, we encountered some issues with integrating the parts together. Specifically because we did not design the hardware interface modules ourselves, we did not have a full grasp of their functioning and had struggles debugging problems relating to them.

## Future Improvements

The final system currently implemented is only a simple application of a music box. This design has many limitations regarding the musicit is able to play. There are many musical enhancements that are not currently available in our system. These include having the ability to play chords, multiple voices, different instrument sounds and variations in musical intonation.

# Works Cited

"Altera DE1 Board." Terasic.com. Terasic Inc, n.d. Web. 10 Apr. 2015.



|  |  |
| --- | --- |
| **Push Buttons** | **Function** |
| **1** | Start |
| **2** | Stop |
| **3** | INIT |
| **4** | Reset |
| **Slide Switches** |  |
| **1** | Repeat |
| **2** | Pause |
| **3** | Transpose |
| **4** | Song Select |
| **5** | Tempo |
| **7-Segment Display** |  |
| **1** | Volume |
| **2** | Volume |
| **3** | Octave |
| **4** | Note Number (hex) |
| **LED** | Beat |

^Ignore that for now.