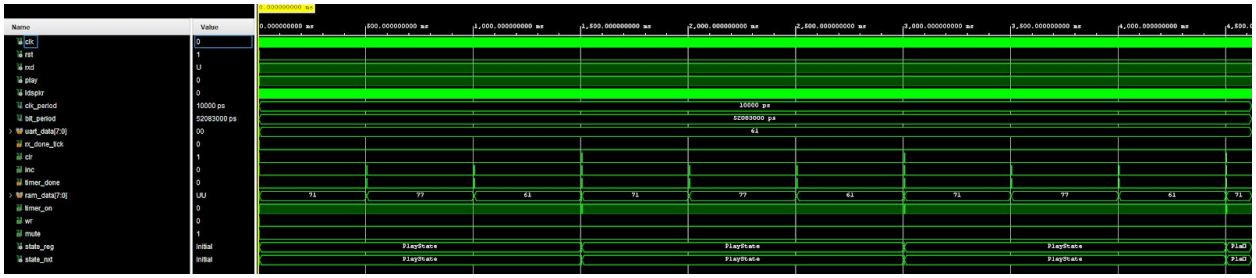
Data Path
The datapath has been programmed with qwertyu as musical notes. Space is mute and delete functions as ENTER.

UART Receiver
There are no transmitting subsystems in our design. We have chosen to make a buad generator using a mod-m counter, and just the UART subsystem by using the concept of finite state machine with idle, start, data and stop. We can easily receive data from the keyboard and send it to the ram for storing.

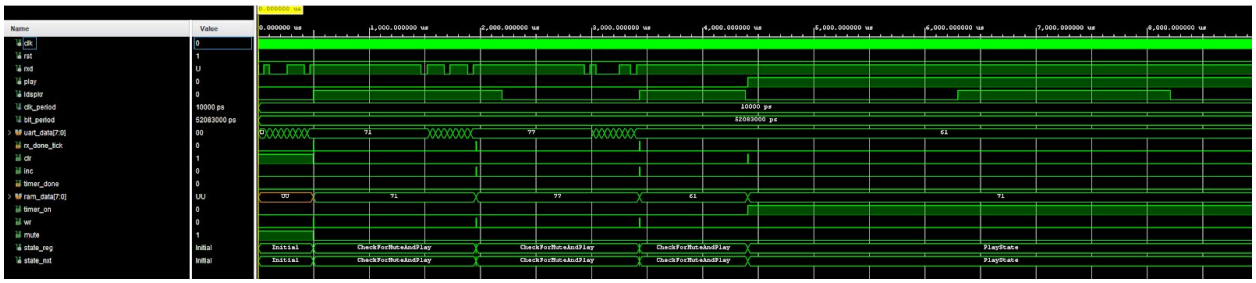
Toggle Flip Flop
The TFF controls the output signal to the speaker. It is controlled by a timer which is operating at the frequency of the desired note. The mute input signal immediately mutes the speaker and no new sound will be played until it is off.

TFF testbench

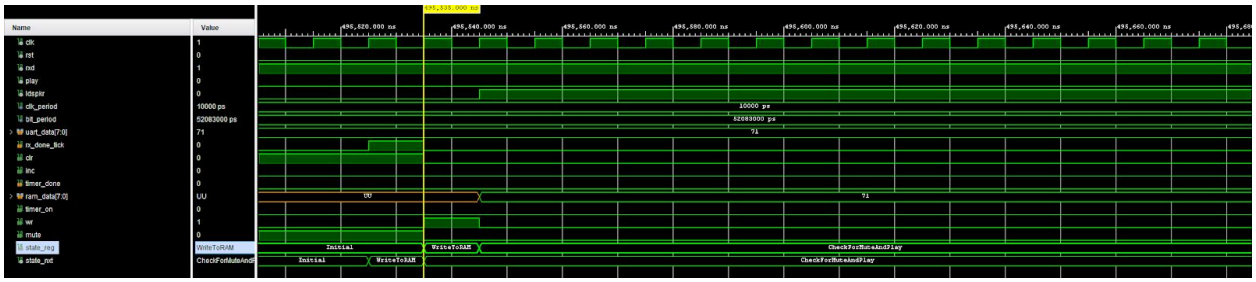
Top level testbench zoom 1



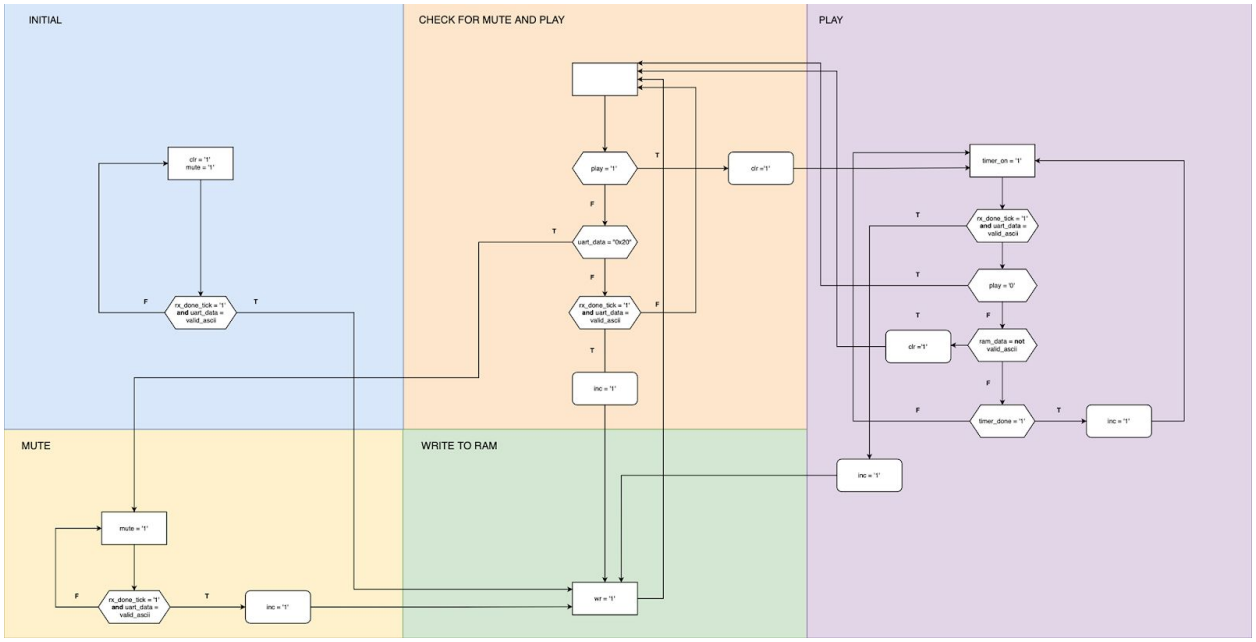
Zoom 2



Zoom 3



Control path ASMD



INITIAL:
Clears the RAM and waits for input from UART.

CHECK FOR MUTE AND PLAY:
Checks if the user wants to play, mute or if new input from UART has arrived. “Valid_ascii” is octave 4 and 5. Before we switch to **WRITE TO RAM** the address counter will be incremented to make sure we do not override past data.

PLAY:
If the user hits a new key or play is disabled we will exit this state. Also, when there is no longer any valid note in RAM we exit this state. A non-valid note marks that we have reached an empty spot in RAM. If these checks are passed, we simply wait until the timer is done, and then we play the next position in RAM.
MUTE:

We are muted until a new note arrives.

WRITE TO RAM:
This is a very simple state with only moore output. We only set “wr” to be 1 (high) so that the current keypress is written to RAM at the address decided by the address counter.

Conclusion:
The waveforms in the simulation performed according to our expectations, however, we were unable to get the playback to work in the synthesised version.