

Something about the CPEN 311 Lab 2

Something about the SOF File

It is located here: [./rtl/simple_ipod_solution.sof](#)

Something about the status

All parts complete, including music playback, pause, resume, speedup, speed down, speed reset, forward, backward.

R key is implemented

44KHz is implemented

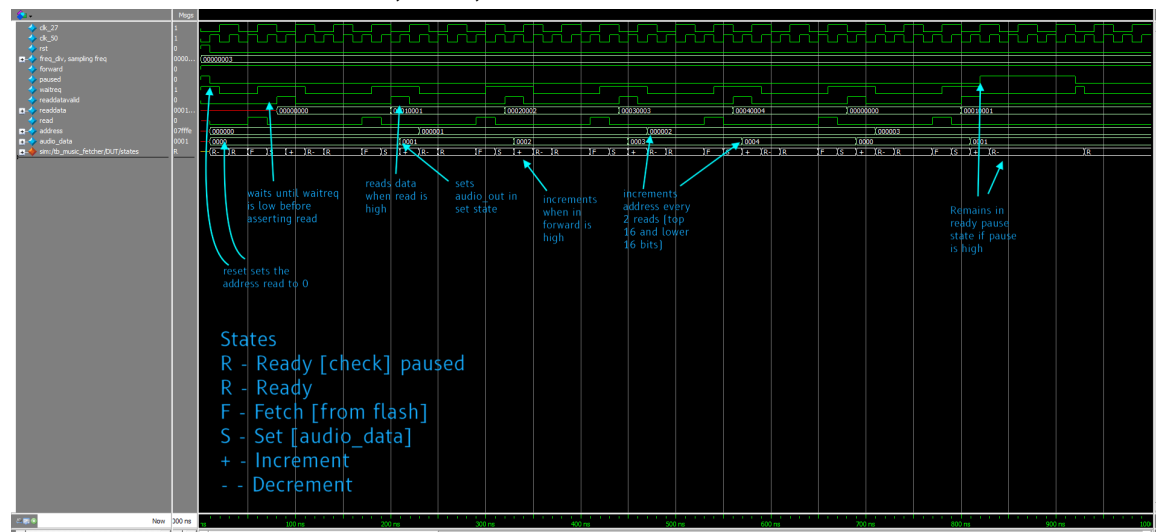
Additional titanic.jic for your listening pleasure.

Annotated Screenshots

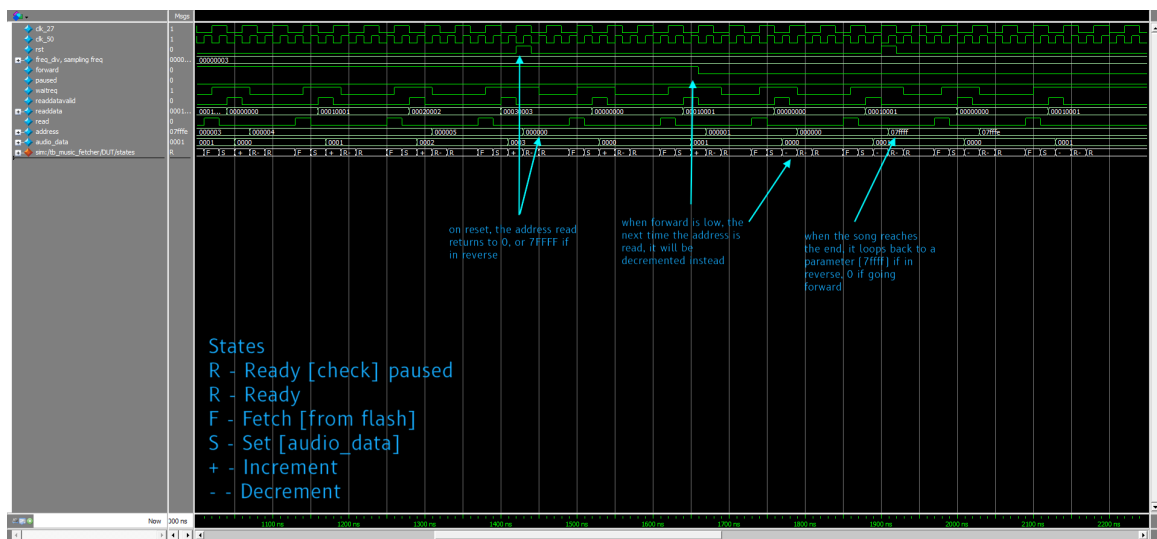
Full sized screenshots are located in the [./doc](#) folder

Simulations

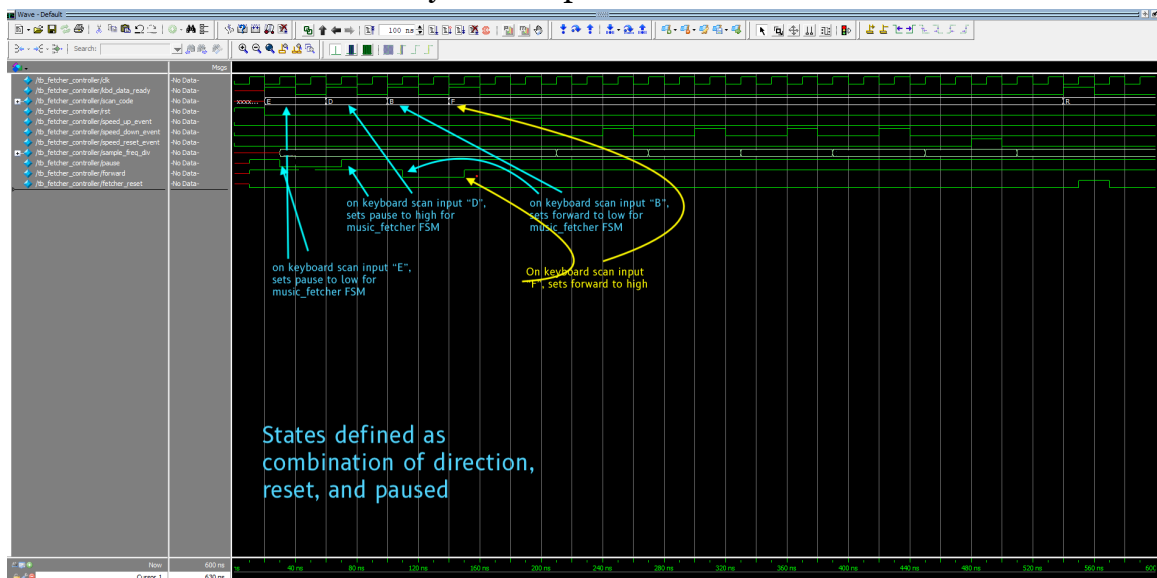
Music fetcher FSM 1 - Wait, read, increment



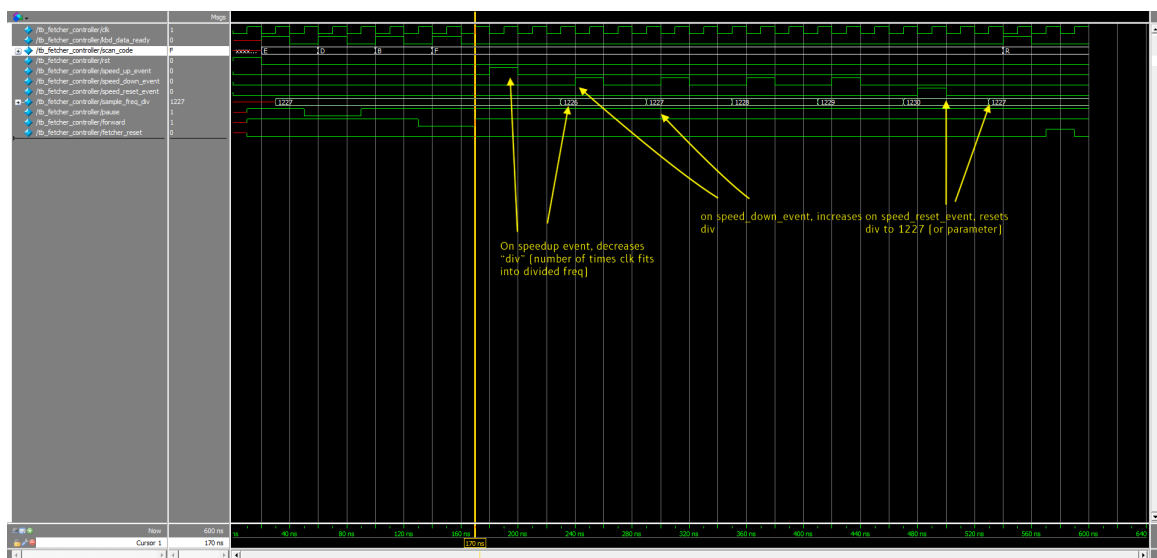
Music fetcher FSM 2 - Reset, reverse, and song end



Fetcher Controller FSM 1 - Keyboard inputs



Fetcher controller FSM 2 - Pushbutton events



SignalTap screenshots

The screenshot displays the SignalTap II Logic Analyzer interface. At the top, the 'Instance Manager' shows three instances: 'auto_signaltap_0' (Not running, 3783 cells), 'music_fetcher_tap' (Not running, 2753 cells), and 'controller_tap' (Not running, 1527 cells). The main area shows a timing diagram for a JTAG chain configuration. The diagram includes signals for 'auto_signaltap_0', 'music_fetcher_tap', and 'controller_tap'. Annotations highlight 'speed down event high for 1 clk' and 'Div increases (Sampling period increases)'. The bottom panel shows the hierarchy of the solution, including 'simple_ipod_solution' and 'controller_tap'.

The screenshot shows the SignalTap II Logic Analyzer interface. The top menu bar includes File, Edit, View, Project, Processing, Tools, Window, and Help. The left pane shows the Instance Manager with three instances: auto_signaltap_0, music_fetcher_tap, and controller_tap. The right pane shows the JTAG Chain Configuration, indicating the hardware is DE-Soc [USB-1] and the device is 02:5C5E(BAS)MAS. The main area displays a timing diagram for a system reset event. The diagram shows various signals over time, with a red vertical line indicating a reset event. Annotations highlight the 'speed reset event high for 1 clk' and the 'Reset to default value 0x4cb (d1227)'. The bottom pane shows the Hierarchy Display and Data Log.

The screenshot shows the SignalTap II Logic Analyzer interface. At the top, the File menu and project path are visible. Below the menu is the Instance Manager table:

Instance	Status	Enabled	LEs: 8063	Memory: 103	Small: 0/0	Medium: 17/	Large: 0/0
auto_insttap_0	Not running	3783 cells	27120 bits	0 blocks	8 blocks	0 blocks	0 blocks
music_fetcher_tap	Not running	2753 cells	17408 bits	0 blocks	8 blocks	0 blocks	0 blocks
controller_tap	Not running	1527 cells	49152 bits	0 blocks	5 blocks	0 blocks	0 blocks

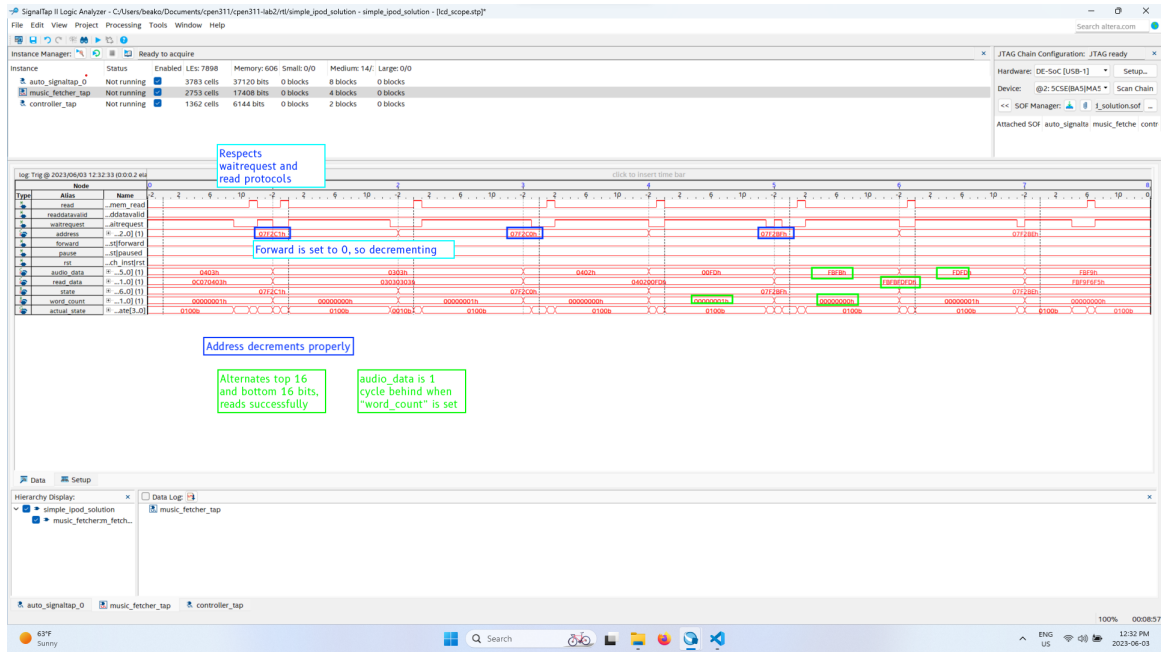
The main window displays a timing diagram for the JTAG chain configuration. The diagram shows signals like controller_inst, inst_speed, and inst_speed_up_event. Annotations highlight speed-up events and the relationship between 'div' and clock frequency.

Annotations in the diagram:

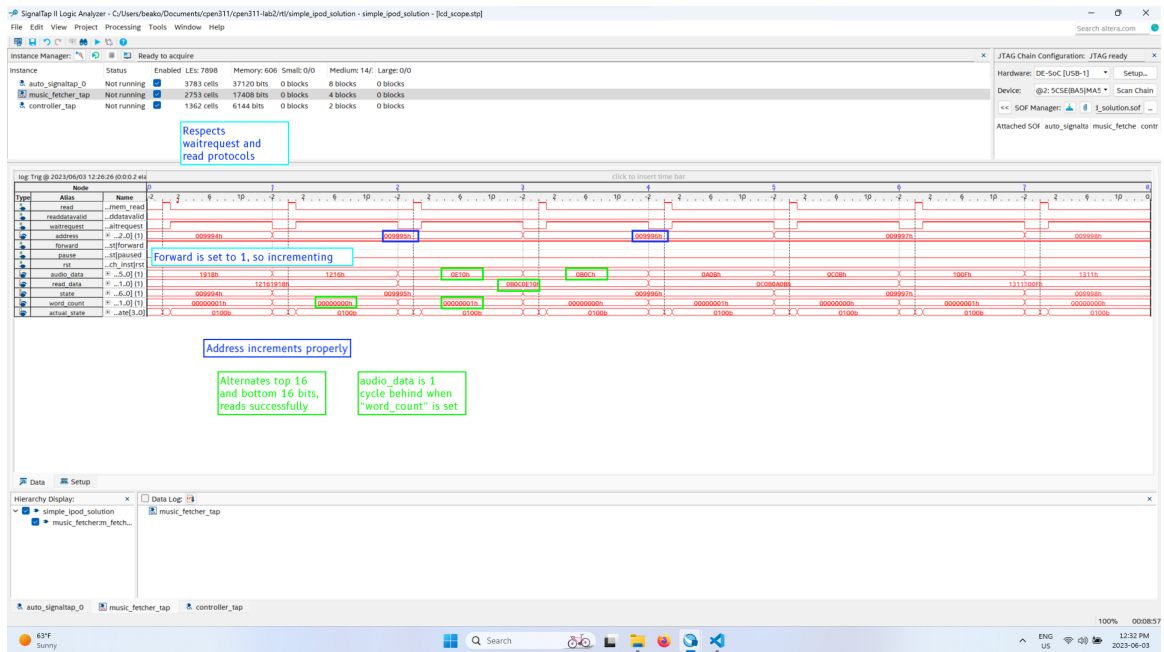
- Speed up event high for 1 clk
- Speed up event high for 1 clk
- div decreases period increases, speeds up
- div decreases period increases, speeds up
- More speed up events
- "div" is number of times clk will fit into desired sampling freq

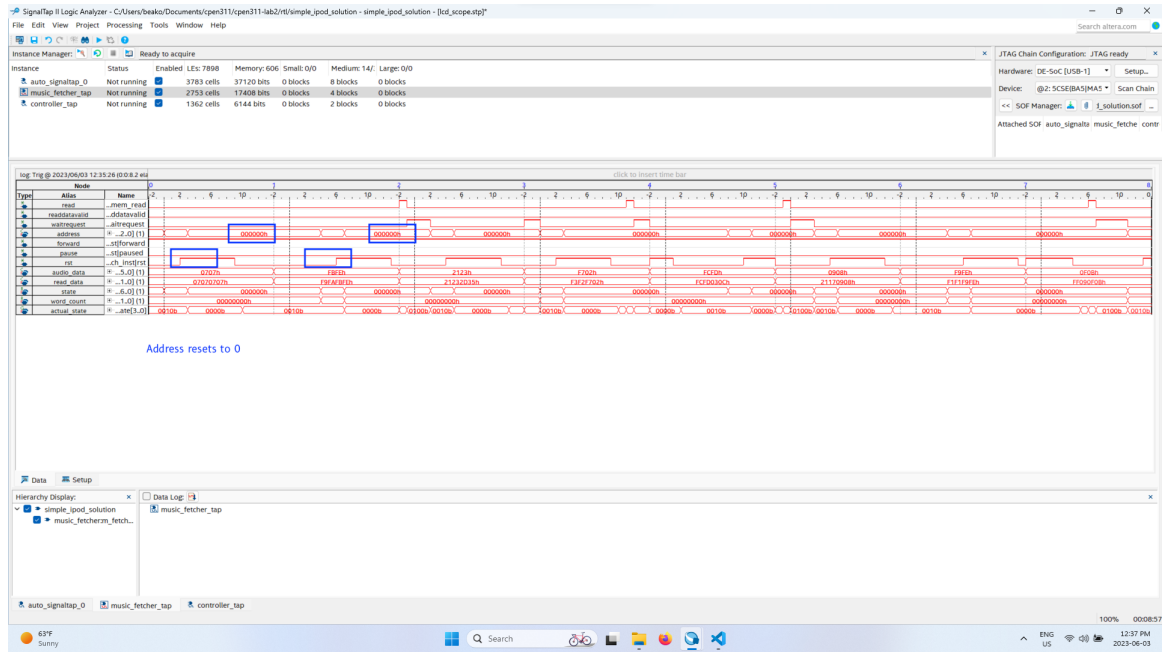
The bottom of the interface shows the Hierarchy Display and Data Log sections.

Music fetcher FSM 1, Decrementing addresses



Music fetcher FSM 2 - incrementing addresses





Something about the simulations

They are located in `./sim`. They were created using modelsim 10.5b. There is the fetcher controller and music fetcher, which each have their own simulations.

The test benches are prefixed to the device under test by “tb”. You should load that file into modelsim when you start your simulation.

A waves.do file is included so you can see the waves as I intended. There are also some custom radices included so you can see the states and button presses more clearly like my screenshots.

A `vsim.wlf` is included so you can load the simulations in if you don't want to simulate them again.

FSM Diagrams

[./doc/FSMs](#), or see PDF version below.

Additional Information

How to customize your songs?

1. Download an mp3 file
2. export as RAW binary
3. Using the intelHex Utility, convert binary to hex
4. using the quartus programmer file converter (or something similar, on file)
5. Set EPCS128
6. Add the Hex file in, the device, and generate.

7. Use JIC file as usual

RTL_44 contains the version that works with 44KHz 8 bit samples.

Reset. Set address to 0, or
maximum depending on whether
input forward is true or false

If not paused

Ready state

- Wait for some inputs

Input:
Pulse from frequency
divider that pulses
each period of the
sampling frequency

Fetch state

- Set outputs to flash memory
to fetch data current address
register
- Wait for flash to be ready

Input: When
memory is done
and ready

Set state

- Set the output to the
audio to whatever you
fetch

Input:
forward =
true

Increment

- Add 1 to the current
address register

Always

Ready paused

- wait to be unpaused

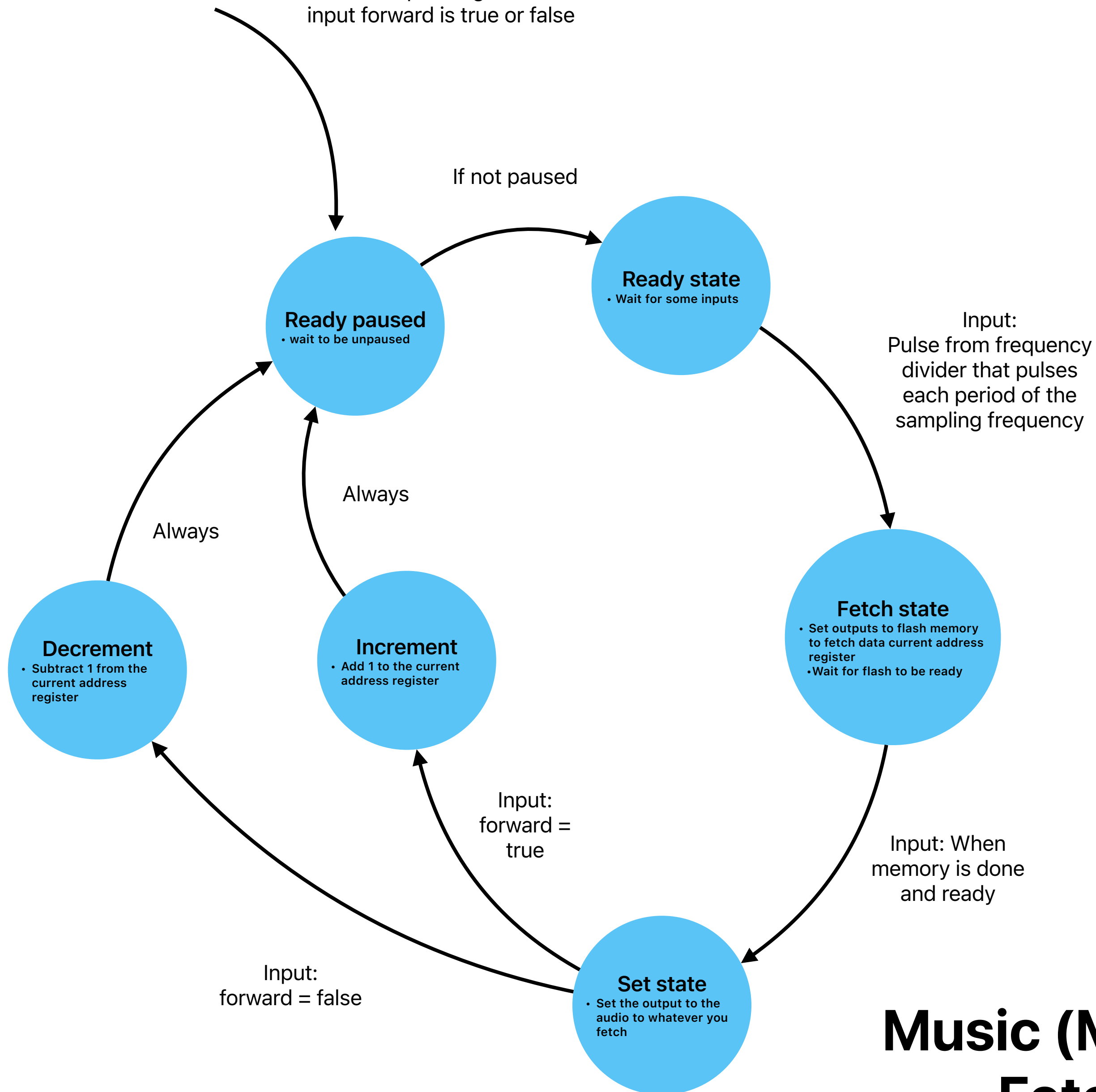
Always

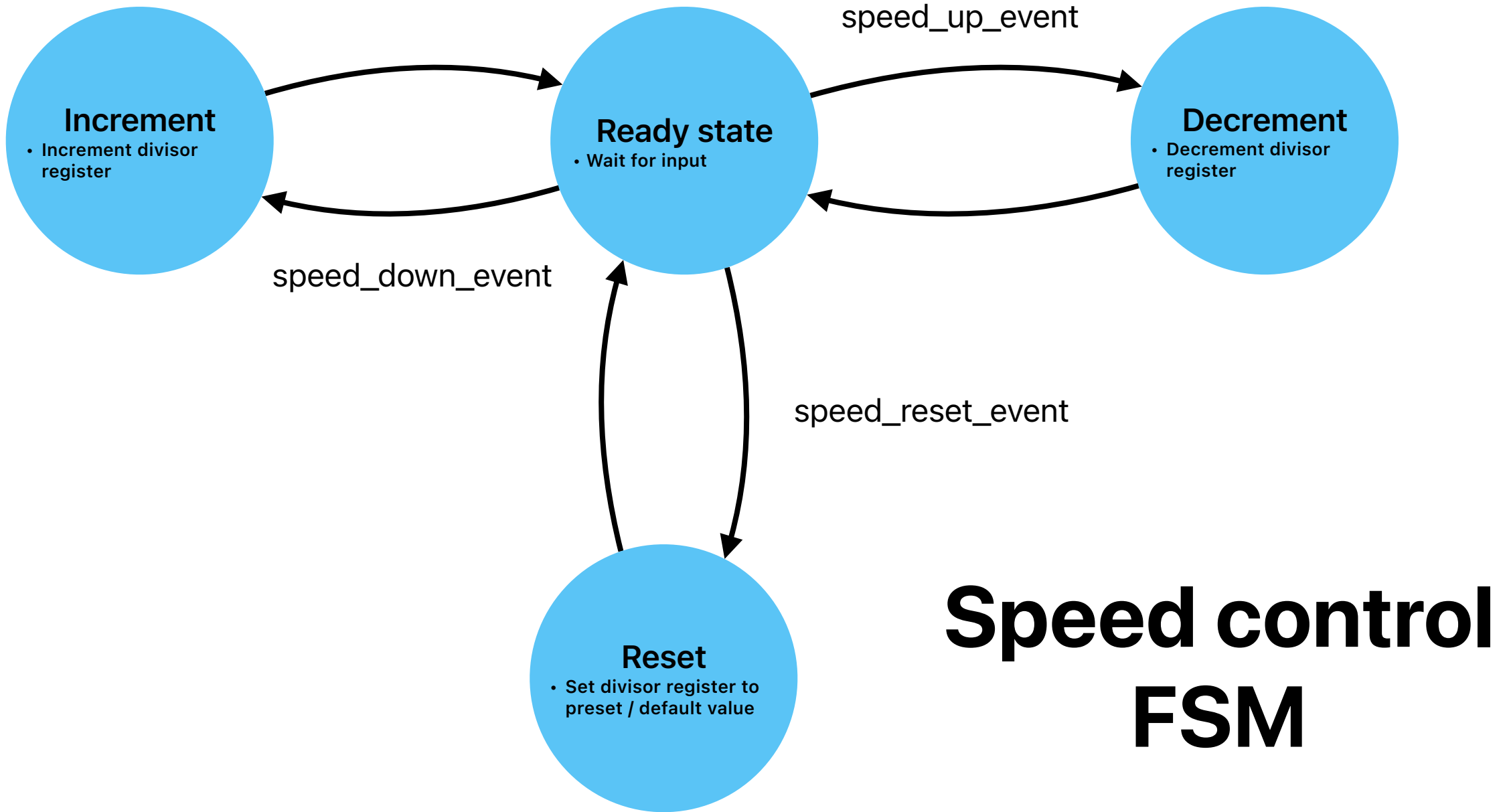
Decrement

- Subtract 1 from the
current address
register

Input:
forward = false

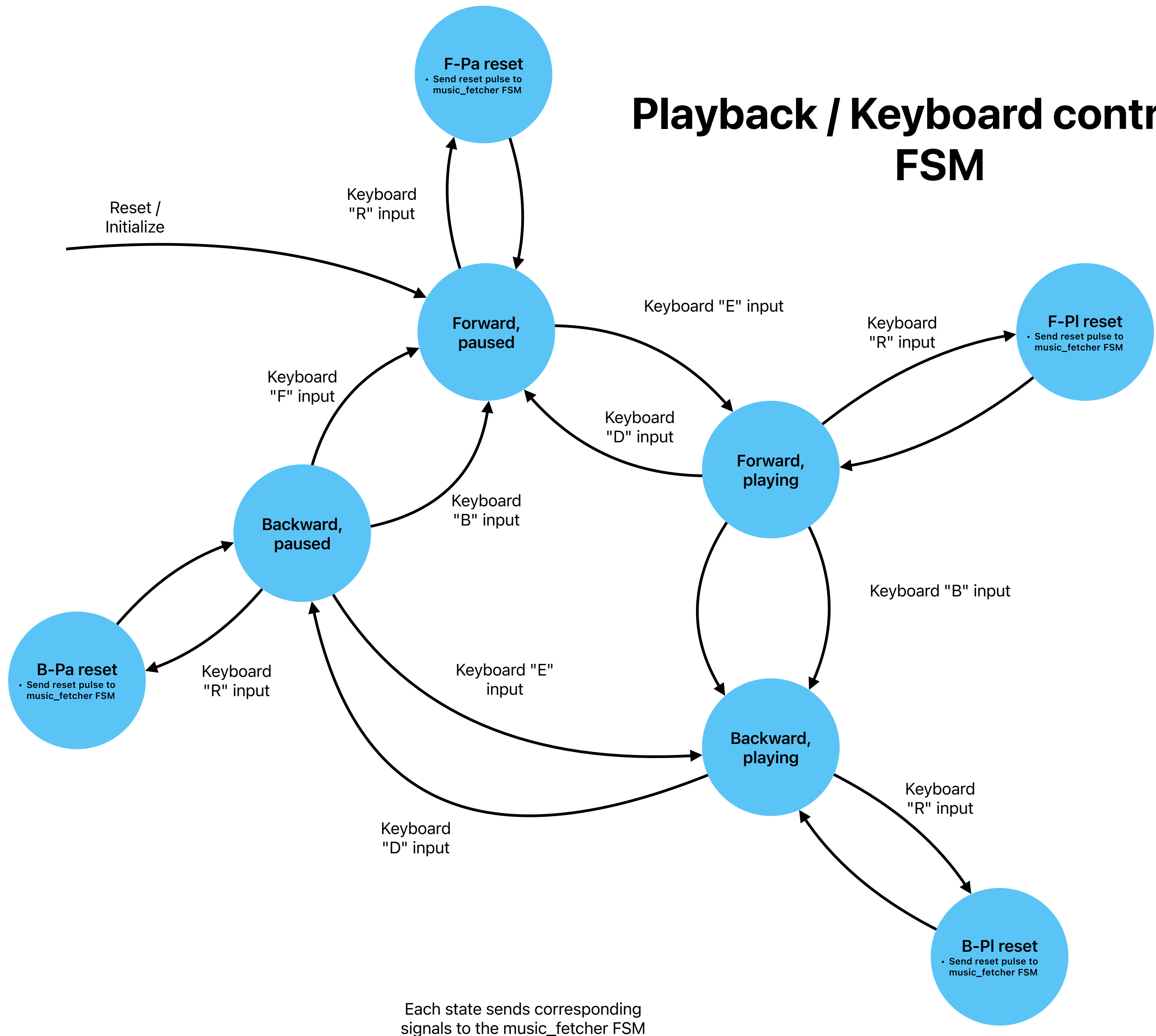
Music (Memory) Fetcher FSM





Speed control FSM

Playback / Keyboard controller FSM



Each state sends corresponding signals to the music_fetcher FSM