

FPGA-BASED MICROPROGRAMMED AUDIO SYNTHESIZER

Kelompok 11:

Syifa Aulia Azhim (2406413445)

Steven (2406359600)

Tomas Warren Wuisang (2406423963)

Irgy Rabbani Sakti (2406438290)

LATAR BELAKANG



Dalam dunia sistem digital, pembangkitan suara (audio synthesis) adalah aplikasi klasik yang menggabungkan konsep pewaktuan (timing) dan pemrosesan sinyal. Pada umumnya, pemutar musik sederhana pada FPGA diimplementasikan menggunakan hardcoded counter yang kaku. Pendekatan ini memiliki kelemahan: untuk mengubah lagu, kita harus mengubah keseluruhan struktur perangkat keras (re-synthesize).

Untuk mengatasi hal tersebut, perlu pendekatan Microprogrammed Architecture. Dengan metode ini, FPGA tidak sekadar menghitung waktu, melainkan bertindak layaknya prosesor sederhana yang membaca "instruksi musik" (opcode dan operand) dari memori. Hal ini membuat sistem lebih fleksibel dan modular, serta memenuhi standar perancangan sistem digital yang kompleks yang mencakup penggunaan Finite State Machine (FSM), memori (ROM), dan aritmatika biner.

TUJUAN

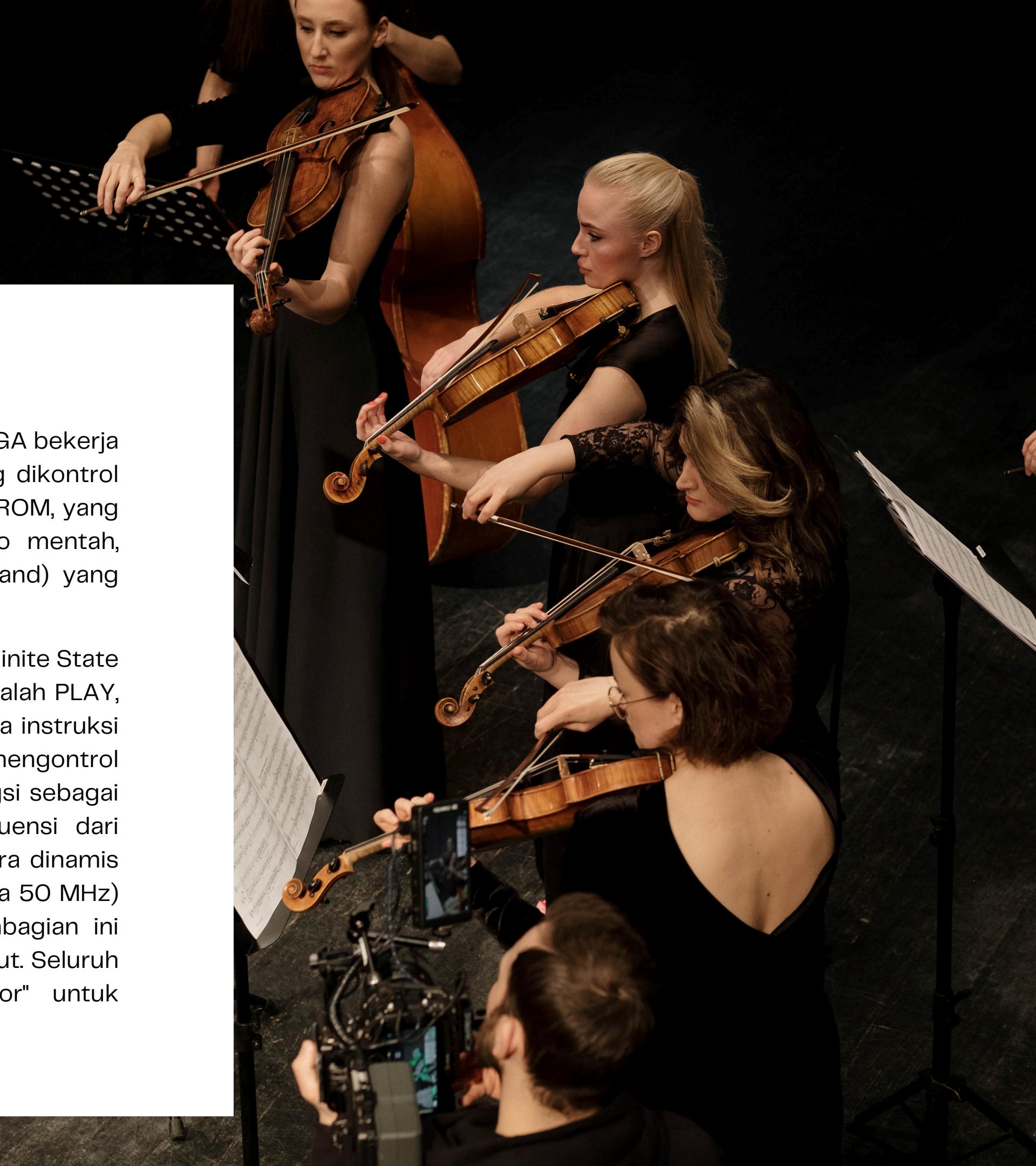
- Mengimplementasikan konsep Microprogramming dan Finite State Machine (FSM) dalam desain hardware.
- Menerapkan struktur kode VHDL yang modular menggunakan Structural Programming dan Package.
- Memahami prinsip pembagi frekuensi (Frequency Divider) untuk menghasilkan nada musik yang akurat dari clock sistem FPGA.
- Memenuhi syarat kelulusan praktikum Perancangan Sistem Digital dengan mencakup minimal 6 modul praktikum.



CARA KERJA

Sistem Audio Synthesizer berbasis Arsitektur Microprogramming pada FPGA bekerja dengan memecah pemrosesan musik menjadi siklus fetch-execute yang dikontrol oleh sebuah Instruction Decoder (FSM). Inti dari sistem ini adalah Music ROM, yang bertindak sebagai "The Librarian" dan tidak menyimpan sampel audio mentah, melainkan menyimpan urutan instruksi mikro (opcode) dan data (operand) yang merepresentasikan partitur lagu, seperti perintah PLAY, WAIT, dan STOP.

Ketika sistem berjalan, Instruction Decoder ("The Architect") yang berupa Finite State Machine (FSM) akan membaca instruksi dari Music ROM. Jika instruksi adalah PLAY, FSM akan meneruskan data frekuensi yang sesuai ke Tone Generator. Jika instruksi adalah WAIT atau loop, FSM akan mengatur Program Counter untuk mengontrol durasi atau pengulangan. Tone Generator ("The Sound Engineer") berfungsi sebagai Programmable Frequency Divider. Generator ini menerima data frekuensi dari Decoder dan menggunakan counter yang batas hitungannya diatur secara dinamis untuk membagi frekuensi clock sistem FPGA yang sangat tinggi (misalnya 50 MHz) menjadi frekuensi audio yang terdengar (20Hz - 20kHz). Proses pembagian ini menghasilkan gelombang kotak yang kemudian menjadi sinyal audio output. Seluruh modul ini digabungkan pada top-level entity oleh "The Integrator" untuk memverifikasi fungsionalitasnya.

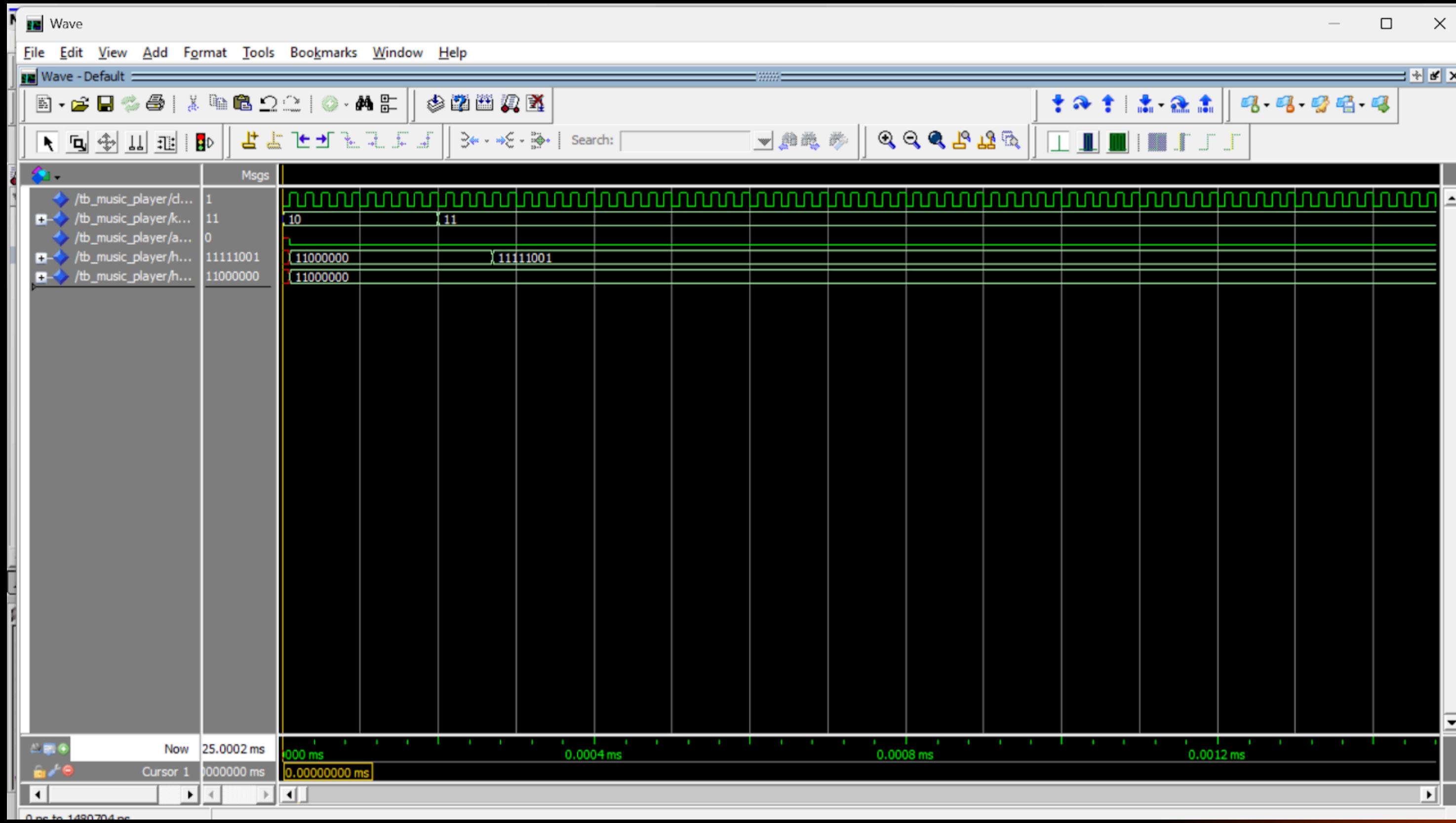




HASIL & TES

Testing dilakukan dengan menggunakan simulasi yang tersedia di ModelSim. Untuk melakukan simulasi, kami membuat sebuah file test bench untuk mempermudah proses simulasi yaitu pada file tb_music_player.vhd.

HASIL WAVE



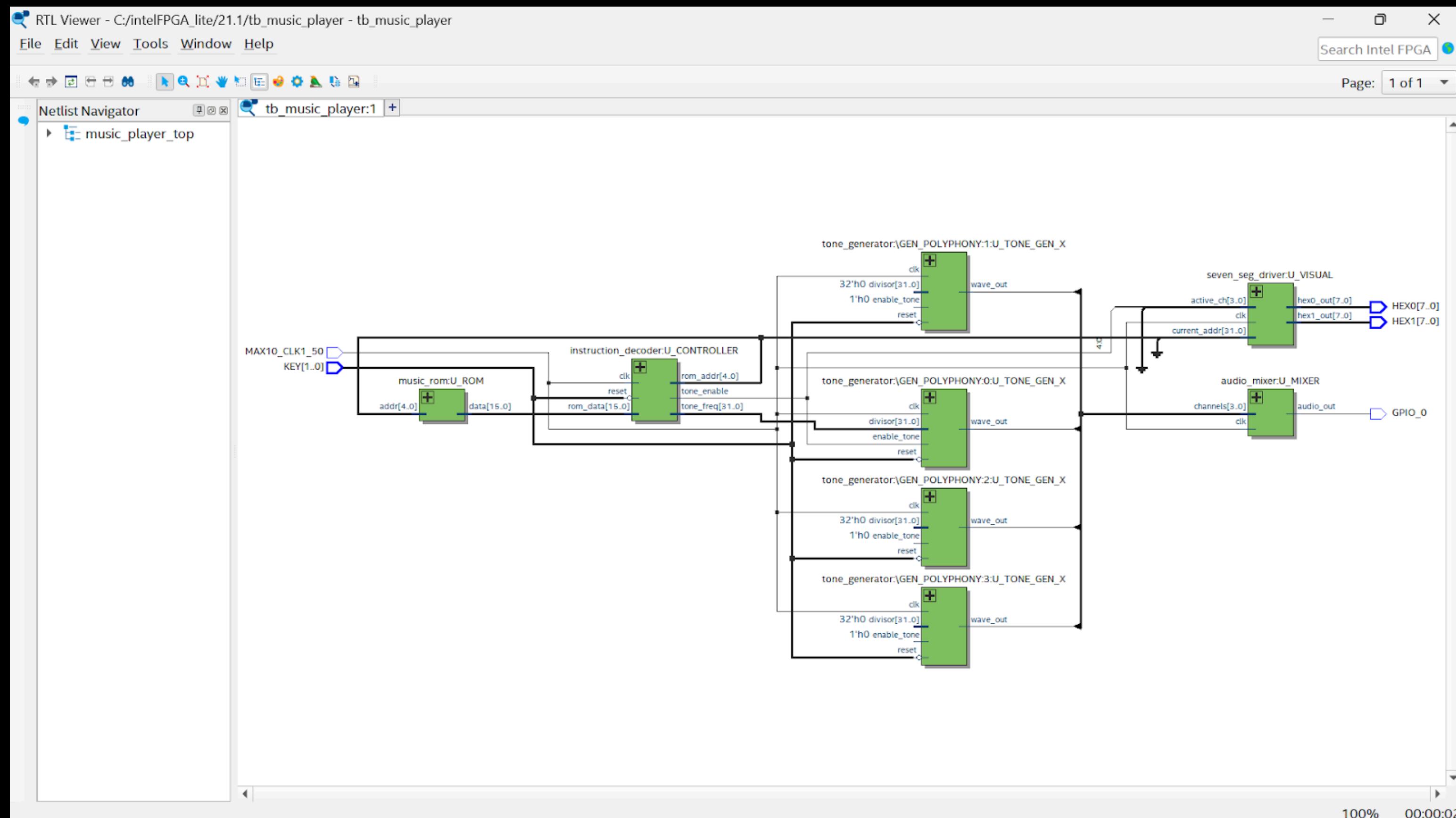
TRANSCRIPT

The screenshot shows the ModelSim simulation environment. The main window title is "ModelSim - INTEL FPGA STARTER EDITION 10.5b". The menu bar includes File, Edit, View, Compile, Simulate, Add, Transcript, Tools, Layout, Bookmarks, Window, and Help. The toolbar contains various simulation and analysis icons. The transcript window displays the following log output:

```
# Time: 0 ps Iteration: 0 Instance: /tb_music_player
# ** Note: Status: [T=0] Simulasi Dimulai. Melakukan Hard Reset...
# Time: 0 ps Iteration: 0 Instance: /tb_music_player
# ** Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
# Time: 0 ps Iteration: 0 Instance: /tb_music_player/uut/U_CONTROLLER
# ** Note: Status: Reset Selesai. Music Player harusnya mulai berjalan.
# Time: 200 ns Iteration: 0 Instance: /tb_music_player
# ** Note:
# Time: 200 ns Iteration: 0 Instance: /tb_music_player
# ** Note: CHECK: Sinyal Audio Valid (Logic 0/1 detected).
# Time: 5000200 ns Iteration: 0 Instance: /tb_music_player
# ** Note: Status: Menjalankan playback untuk 20 ms kedepan...
# Time: 5000200 ns Iteration: 0 Instance: /tb_music_player
# ** Note: Status: Playback progress... 5 ms
# Time: 10000200 ns Iteration: 0 Instance: /tb_music_player
# ** Note: Status: Playback progress... 10 ms
# Time: 15000200 ns Iteration: 0 Instance: /tb_music_player
# ** Note: Status: Playback progress... 15 ms
# Time: 20000200 ns Iteration: 0 Instance: /tb_music_player
# ** Note: Status: Playback progress... 20 ms
# Time: 25000200 ns Iteration: 0 Instance: /tb_music_player
# ** Note:
# Time: 25000200 ns Iteration: 0 Instance: /tb_music_player
# ** Note: Status: Simulasi Selesai. Silakan cek Waveform.
# Time: 25000200 ns Iteration: 0 Instance: /tb_music_player
# ** Note:
# Time: 25000200 ns Iteration: 0 Instance: /tb_music_player
# ** Failure: Simulation Finished Successfully
# Time: 25000200 ns Iteration: 0 Process: /tb_music_player/stim_proc File: C:/Users/fosin/Downloads/FINPRO_PSD_11/tb_music_player.vhd
```

The status bar at the bottom indicates "Project: finpro_psd11 Now: 25,000,200 ns Delta: 0 sim:/tb_music_player/stim_proc".

SINTESIS





HOME

SERVICE

ABOUT US

CONTACT US

THANK YOU