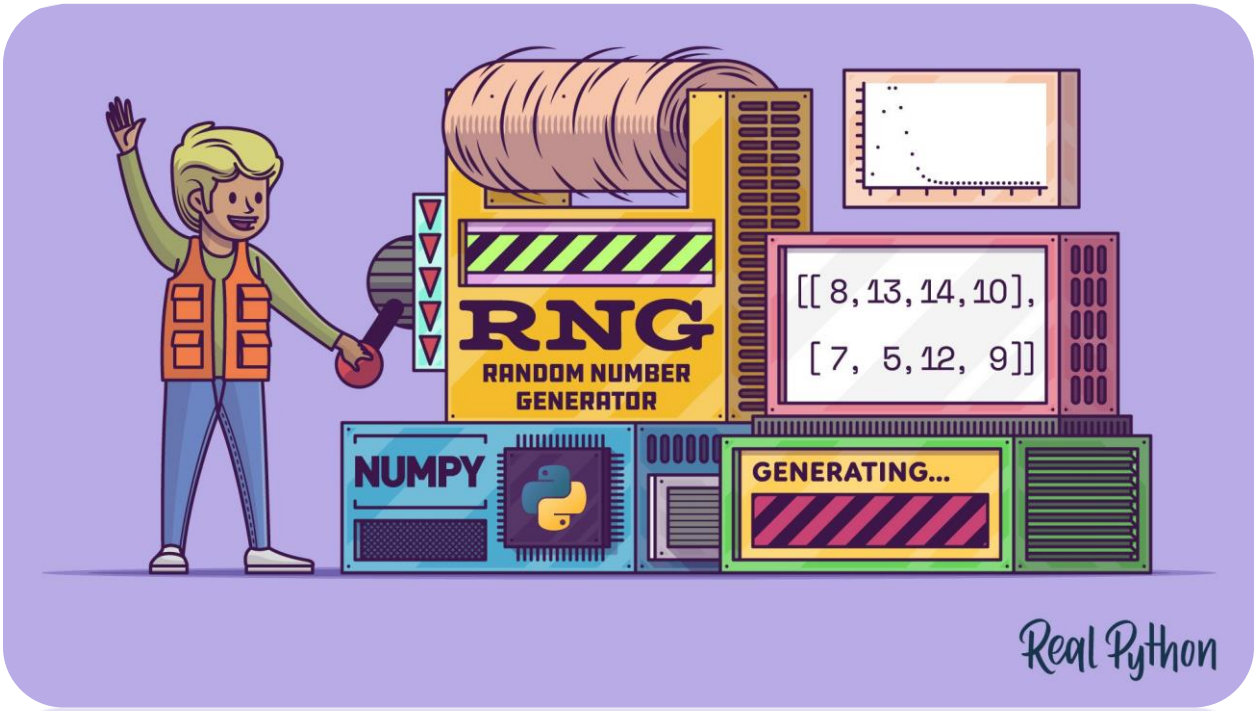
MINI PROJET VHDL :

RNG



Project made by: Amari Saad Nossaer
Bensaadoune Yassine
Oujjeyr Driss

In our project we are going to talk in details about RNGs (random number generator), where we will be showing interest for slot machines, how they work (their RNG algorithm) and how people make money from this type of devices.

Definition :

- Slot machines are gambling devices that consist of spinning reels with various symbols. Players insert money or credits, spin the reels, and win prizes based on the alignment of symbols on paylines. Modern slot machines use random number generators (RNGs) to ensure that each spin's outcome is random and independent. They are popular in casinos due to their simplicity and potential for large jackpots.

A brief explanation of our code:

We will use LSFR as s shift register, it can generate a sequence of bits that appears random and is used in various domains as slot machines, error detection…

Explanation:

- Entity Declaration: The lfsr entity has four ports: clk (clock input), reset (reset input), enable (enable signal for generating new random numbers), and random_number_generated (8 bit output random number).
- Architecture: The rng architecture contains:
  -A signal lfsr_rng representing the 8-bit LFSR register initialized to a non-zero value (00000001 in this case).
  -A process sensitive to the clock and reset signals.
- Process Block:
  -On reset (reset = '1'), the LFSR is set to its initial value (00000001).
  -On the rising edge of the clock (rising_edge (clk)), if the enable signal is high, the LFSR performs a shit right operation:
  -The feedback is created using the XOR operation on specific bits of the LESR (lfsr_rng(7) XOR lfsr_rng (5) XOR lfsr_rng (4) XOR lfsr_rng (3)), which corresponds to the chosen feedback for an 8 bit LFSR.
- Shift Right: 1fsr_rng(6 down to 0) takes the 7 least significant bits of the register and shifts them one bit to the right, A new Bit -Calculated by XOR: calculates the new bit by taking bits 7, 5, 4 and 3 of lfsr_rng and combining them with an XOR operation.
- Concatenation: The result of the right shift is concatenated with the new bit calculated by XOR, thus forming the new value of lfsr_rng.

.

Gameplay Mechanism :

- Bet Insertion: Players insert coins, tokens, or electronic credits to play.
- Activation: By pulling a lever (on older machines) or pressing a button, the reels spin.
- Result: The reels stop, and symbols appear on a payline. If a winning combination is formed, the player receives a payout.

Random Number Generator (RNG) :

- Modern machines use an RNG to determine the outcome of each spin. This ensures that each spin is independent and entirely random.

.

Advantages and Strategies :

Advantages :

- Ease of Play: No need for complex skills or strategies.
- High Potential Payouts: The possibility of winning significant jackpots.
- Variety: A wide range of themes and game styles.

Strategies :

- Budget Management: Determine in advance how much money you are willing to spend and stick to it.
- Choosing Machines with a Good Return to Player (RTP) : Look for machines with a high RTP to maximize your chances of winning.
- Taking Advantage of Bonuses: Use the bonuses and promotions offered by casinos to extend your gameplay.

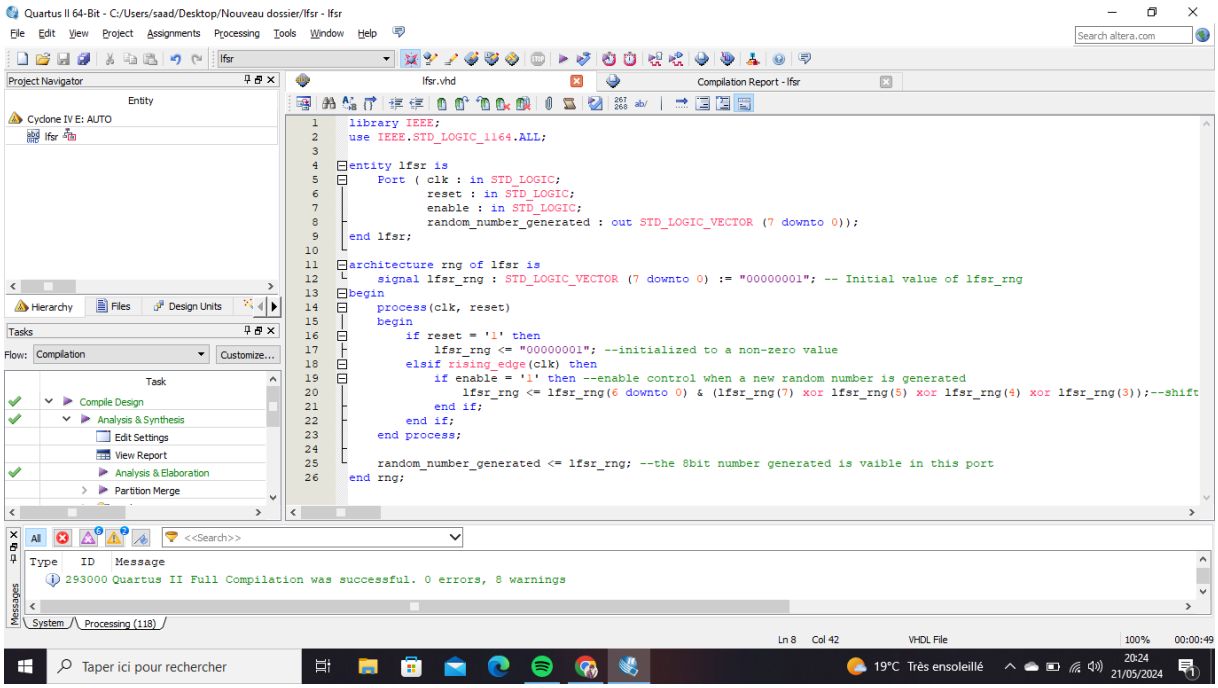
Conclusion :

- Slot machines are an integral part of the gaming and casino industry. Their simplicity, combined with the potential for significant jackpots, makes them a popular choice for many players. Whether online or in physical casinos, slot machines continue to attract and entertain millions of players worldwide.

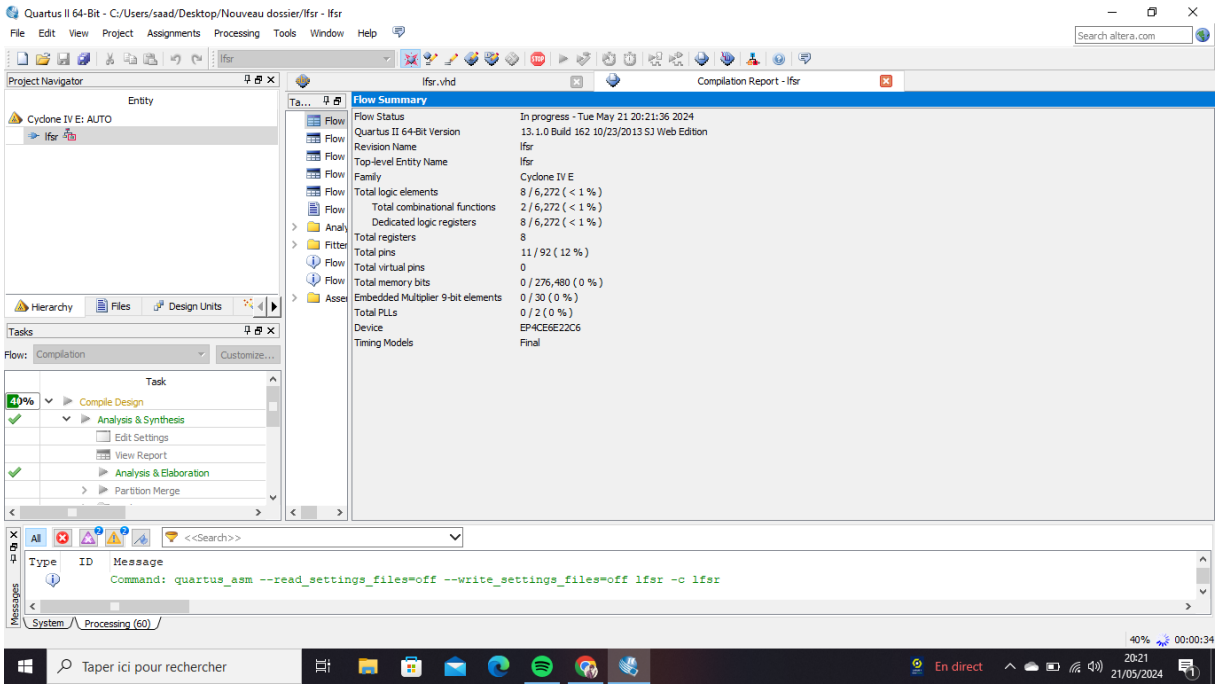
Code & execution:

Code (clearer on GitHub):

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lfsr is
   Port ( clk : in STD_LOGIC;
          reset : in STD_LOGIC;
          enable : in STD_LOGIC;
          random_number_generated : out STD_LOGIC_VECTOR (7 downto 0));
end lfsr;

architecture rng of lfsr is
   signal lfsr_rng : STD_LOGIC_VECTOR (7 downto 0) := "00000001"; -- Initial
value of lfsr_rng
begin
   process(clk, reset)
   begin
      if reset = '1' then
         lfsr_rng <= "00000001"; --initialized to a non-zero value
      elsif rising_edge(clk) then
         if enable = '1' then --enable control when a new random number is
generated
            lfsr_rng <= lfsr_rng(6 downto 0) & (lfsr_rng(7) xor lfsr_rng(5) xor
lfsr_rng(4) xor lfsr_rng(3));--shift right, take the 7 last significant bit and a new
bit is calculated (XOR)
         end if;
      end if;
   end process;

   random_number_generated <= lfsr_rng; --the 8bit number generated is
vaible in this port
end rng;
```


Quartus II 64-Bit - C:/Users/saad/Desktop/Nouveau dossier/lfsr - lfsr
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
Project Navigator
Entity
Cyclone IV E: AUTO
lfsr
lfsr.vhd
Compilation Report - lfsr
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity lfsr is
Port (clk : in STD_LOGIC;
reset : in STD_LOGIC;
enable : in STD_LOGIC;
random_number_generated : out STD_LOGIC_VECTOR (7 downto 0));
end lfsr;
architecture rng of lfsr is
signal lfsr_rng : STD_LOGIC_VECTOR (7 downto 0) := "00000001"; -- Initial value of lfsr_rng
begin
process(clk, reset)
begin
if reset = '1' then
lfsr_rng <= "00000001"; --initialized to a non-zero value
elsif rising_edge(clk) then
if enable = '1' then --enable control when a new random number is generated
lfsr_rng <= lfsr_rng(6 downto 0) & (lfsr_rng(7) xor lfsr_rng(5) xor lfsr_rng(4) xor lfsr_rng(3));--shift
end if;
end if;
end process;
random_number_generated <= lfsr_rng; --the 8bit number generated is vaible in this port
end rng;
Hierarchy
Files
Design Units
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Edit Settings
View Report
Analysis & Elaboration
Partition Merge
Type ID Message
293000 Quartus II Full Compilation was successful. 0 errors, 8 warnings
System
Processing (118)
Ln 8 Col 42 VHDL File 100% 00:00:49
Taper ici pour rechercher
19°C Très ensoleillé
20:24
21/05/2024

Execution :


Quartus II 64-Bit - C:/Users/saad/Desktop/Nouveau dossier/lfsr - lfsr
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
Project Navigator
Entity
Cyclone IV E: AUTO
lfsr
lfsr.vhd
Compilation Report - lfsr
Flow Summary
Flow Status In progress - Tue May 21 20:21:36 2024
Quartus II 64-Bit Version 13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name lfsr
Top-level Entity Name lfsr
Family Cyclone IV E
Total logic elements 8 / 6,272 (< 1 %)
Total combinational functions 2 / 6,272 (< 1 %)
Dedicated logic registers 8 / 6,272 (< 1 %)
Total registers 8
Total pins 11 / 92 (12 %)
Total virtual pins 0
Total memory bits 0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements 0 / 30 (0 %)
Total PLLs 0 / 2 (0 %)
Device EP4CE6E22C6
Timing Models Final
Hierarchy
Files
Design Units
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Edit Settings
View Report
Analysis & Elaboration
Partition Merge
Type ID Message
Command: quartus_asm --read_settings_files=off --write_settings_files=off lfsr -c lfsr
System
Processing (60)
40% 00:00:34
Taper ici pour rechercher
En direct
20:21
21/05/2024