HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

CIRCUIT DESIGN LAB WS 21/22

Ravensburg University of Applied Sciences

The Tea-Brewing-Controller

Project By:	Adil Azeez (34679) Gautam Sajeen (34636)
Guided By:	Dr.-Ing., Professor Walter Ludescher
Submitted on:	30.01.2022

Acknowledgement

We are glad to express our indebtedness to our guide Prof. Dr. Walter Ludescher, Department of Electrical Engineering and Information Technology, Hochschule Ravensburg - Weingarten, for his constant guidance and help at all the stages of this project.

The objective of this project is to understand how a Circuit has to been designed using VHDL programming and implement it in the real world. The ultimate target is detect which mode has been selected and defining the timer range accordingly. Also the proper functionality of the Acoustic alarm has to be assured after the brewing process.

Abstract

In this Project, we have designed a circuit using VHDL (Very high-speed integrated circuit hardware description language) and FPGA (Field programmable gate array). A Tea Brewing Controller has been made with a variety of logical components efficiently connected to each other for a proper functionality. Basically, a Teabag has to be submerged in boiling water for some defined time ranging from two to five minutes. The system consists of a sensor which detected the teabag being dumped. It also monitors the time range and subsequently activates a buzzer.

Our project consists or two main components namely "Teabag-Sensor-Controller" and "Acoustic-Tea-Alarm". The TSC manages the sensor and detectes the time of arrival of the teabag. It also propagates this information to the ATA. The ATA reads the TOA data passes them to the transmitting part of the serial interface, using the ASCII code. It keeps track of the brewing process and activates the acoustic alarm signal when brewing has ended.

The circuit has been designed using VHDL. A block diagram of the Top Level Architecture and the Sound Component has been made to explain the entire functionality of the system.

Contents

Section with acronyms

List of Figures

1 Introduction

In prder to design this circuitry, a few specific setups are necessary such as the Software setup for Simulation and Synthesis, Hardware Setup and uploading the code to the FPGA Board. This chapter contains the aforementioned details.

1.1 Introduction to FPGA

A Field Programmable Gate Array (FPGA) is a semiconductor device with programmable logic and interconnects. The programmable logic components can be used to replicate the functionality of fundamental logic gates like AND, OR, NOT, and XOR, as well as more advanced combinational operations like decoders and arithmetic functions. A hardware description language is used to specify the FPGA configuration (HDL). The most typical FPGA architecture comprises of a Configurable Logic Block (CLB) or Logic Array Block (LAB) array of logic blocks, I/O pads for off-chip connections, and programmable routing channels for logical function implementation. In addition to digital functions, FPGAs contain analogue features. Differential comparators on input pins designed to be coupled to differential signaling channels is a frequent analogue feature.

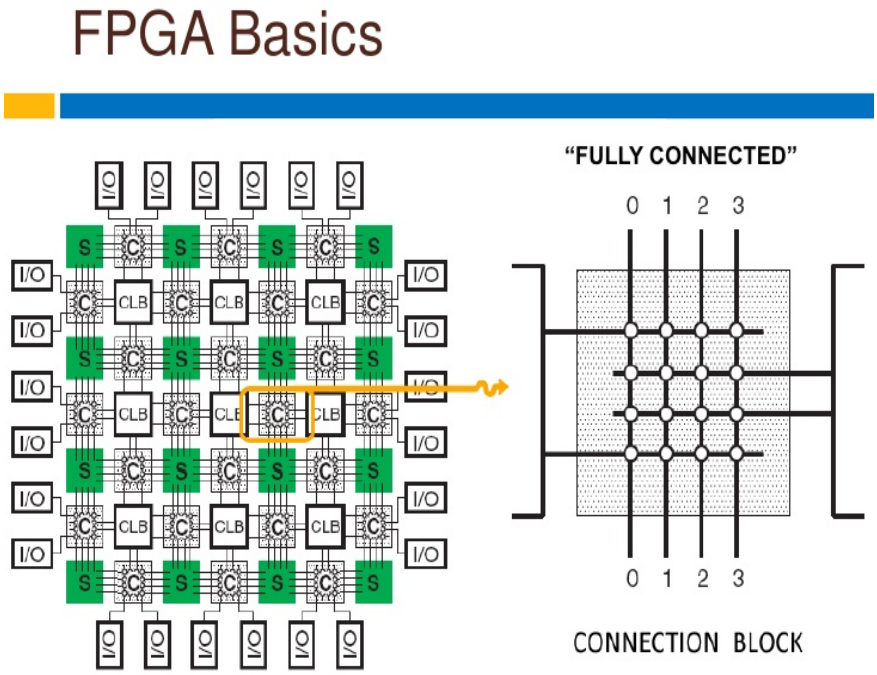

Figure 1: General Architecture of FPGA

1.2 Introduction to the hardware used for the Project

The following components are required to implement this project in reality:

Figure 2: Three wire Interface

Figure 3: Arduino Microcontroller

Figure 4: Earphone

1.3 Information about VHDL

VHDL stands for Very High-Speed Integration Circuit HDL (Hardware Description Language). It is an IEEE (Institute of Electrical and Electronics Engineers) standard hardware description language that is used to describe and simulate the behavior of complex digital circuits.
VHDL supports the following features:

- Design methodologies and their features
- Sequential and concurrent activities
- Design exchange
- Standardization
- Documentation
- Readability
- Large Scale Design
- A wide range of descriptive capability

Why did we choose VHDL ?

- It supports various design methodologies like Top-down approach and Bottom-up approach
- It provides a flexible design language
- It allows better design management
- It allows detailed implementations
- It supports a multi-level abstraction
- It provides tight coupling to lower levels of design
- It supports all CAD tools
- It strongly supports code re usability and code sharing

VHDL has constructs to handle the parallelism inherent in hardware designs, but these constructs (processes) differ in syntax from the parallel constructs in Ada (tasks). It is relatively easy for an inexperienced developer to produce code that simulates successfully but that cannot be synthesized into a real device, or is too large to be practical. One particular pitfall is the accidental production of transparent latches rather than D-type flip-flops as storage elements.

One can design hardware in a VHDL IDE (for FPGA implementation such as Xilinx ISE, Altera Quartus, Synopsys Synplify or Mentor Graphics HDL Designer) to produce the RTL schematic of the desired circuit. After that, the generated schematic can be verified using simulation software which shows the waveforms of inputs and outputs of the circuit

after generating the appropriate testbench. To generate an appropriate testbench for a particular circuit or VHDL code, the inputs have to be defined correctly. For example, for clock input, a loop process or an iterative statement is required.

A final point is that when a VHDL model is translated into the "gates and wires" that are mapped onto a programmable logic device such as a CPLD or FPGA, and then it is the actual hardware being configured, rather than the VHDL code being "executed" as if on some form of a processor chip.

2 CIRCUIT DESIGNING WITH VHDL

A brief overview of the top level of the Circuit blocks will be described in this chapter. Moreover, the top level consists of some small blocks such as counters, multiplexers,etc is going to described in this chapter.

2.1 Top Level Block Diagram Entity

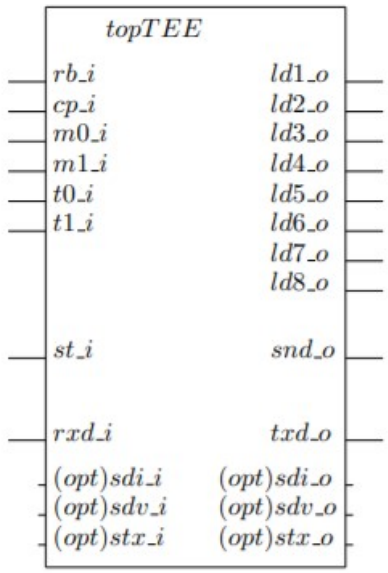

Figure 5: Top Level Entity

1. **cp_i:** cp_i denotes the system clock. It is a continuous sequence of low and High on the line providing to it. The system clock is needed to synchronize all the components that is running on the FPGA board. That means, they all do their work only if the clock is high;never when it's low. And because of the clock speed is set above the longest time any signal needs to propagate through any circuit on the board,this signals is preventing signals from arriving before other signals and thus makes everything safe and synchronized.

2. **rb_i:** rb_i denotes the reset button signal. It is an active low signal. It is a signal that initialize the complete system. Reset can be switch button or a push button and is generally used by the user. It restart all the interfaces and all the State Machines which are working inside the FPGA chip, are forced to go to its initial state.

3. **st_i:** st_i is the signal that is coming from the sensor, that is activated when the Teabag is dipped in the boiling water. As soon as the teabag is dipped, the sensor is set to active high untill the Teabag stays in the water and the Alarm starts ringing. Therefore this signal is collected by the TSC and propagated towards the ATA.

4. **txd_o:** txd_o is the signal that send a serial line of data that is processed inside the system. That means, the data that comes are the bits that are sent in UART sequence corresponding to the RS-232 Protocol.

Here, the signals from the top_level entity have been discussed. The Architecture is declared and explained in the next Section.

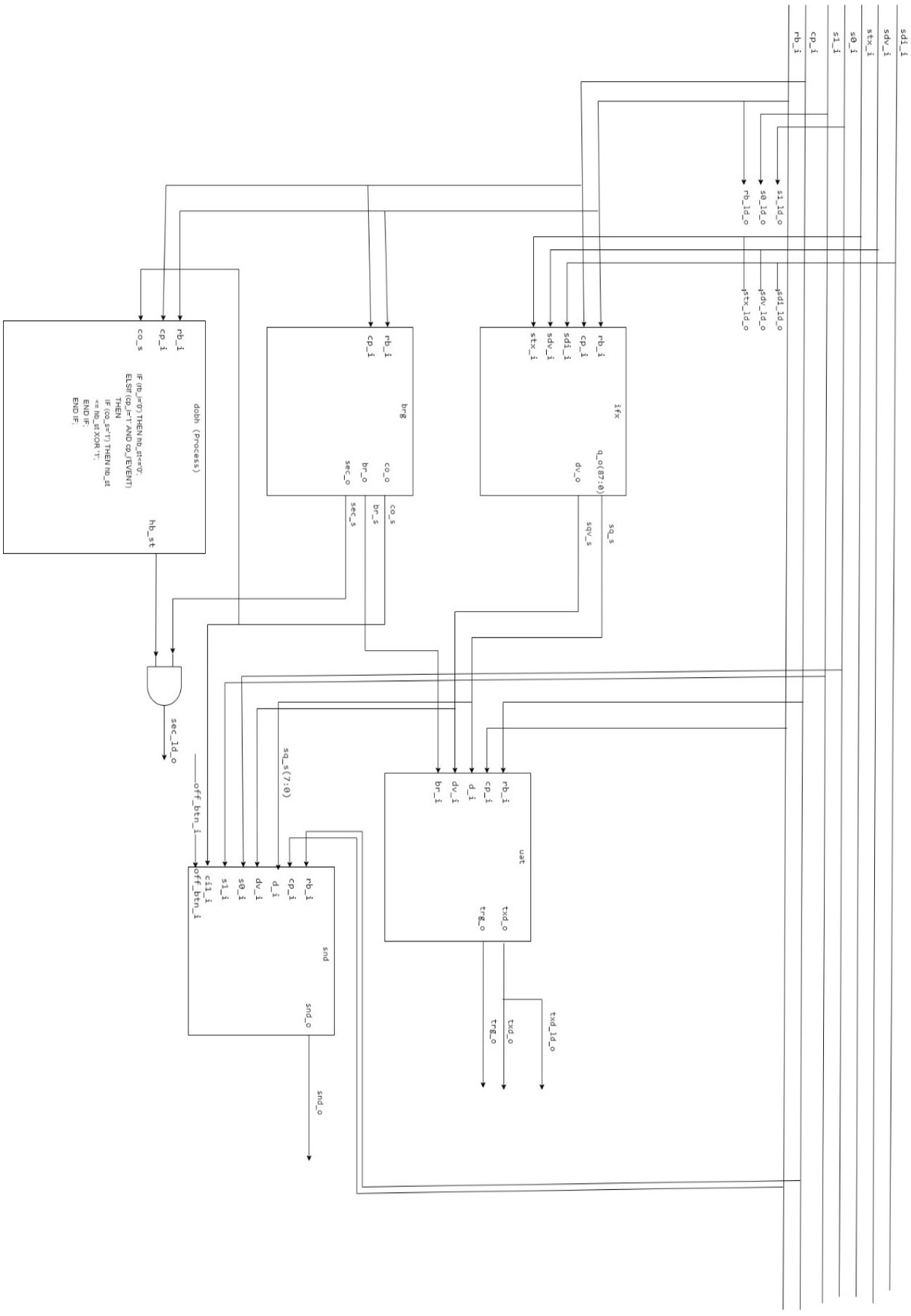

Figure 6: Top Level Architecture

2.2 ifx

The entity of the ifx looks like :

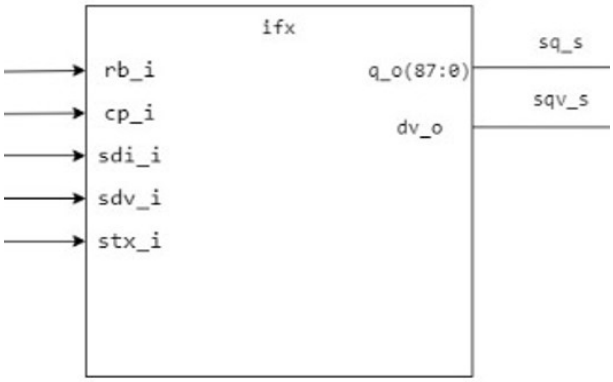

Figure 7: ifx Entity

This block is responsible for handling the serial input. It accepts the data serially from the outside world. rb_i is the reset bit. cp_i is the clock pulse. sd_i is serial data, sdv_i is serial data valid , stx_i basically starts the transmission. The output q_0 transmits the data as a standard logic vector of 87 down to 0. dv_o is associated with validity of the data.

2.3 brg

The entity of the Baudrate Generator looks like :

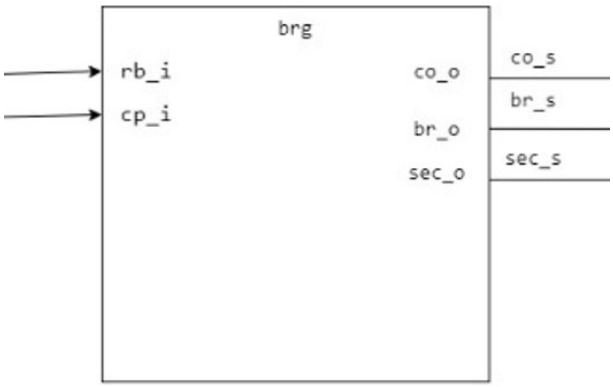

Figure 8: Baudrate Generator Entity

This block is responsible for generating the Baudrate. rb_i is the reset bit. cp_i is the clock pulse. It outputs the baudrate, the one second signal and the one second carry out.

2.4 uat

The entity of the Transmission looks like :

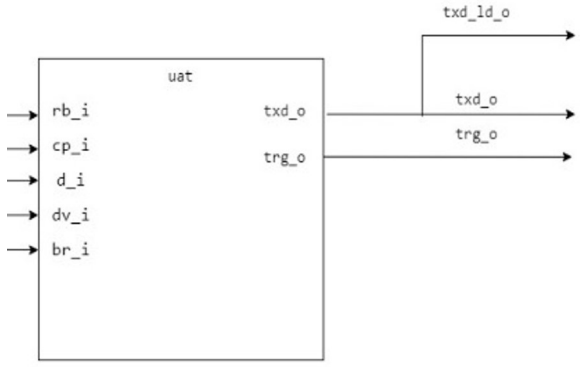

Figure 9: uat Entity

This block is responsible for transmitting the data to the outside world. rb_i is the reset bit. cp_i is the clock pulse. d_i is data which is needed to be transmitted. dv_i is the validity of the data. txd_o transmits the data serially. trg_o represents the slope at the start bit.

2.5 snd

The entity of the Sound Generator looks like :

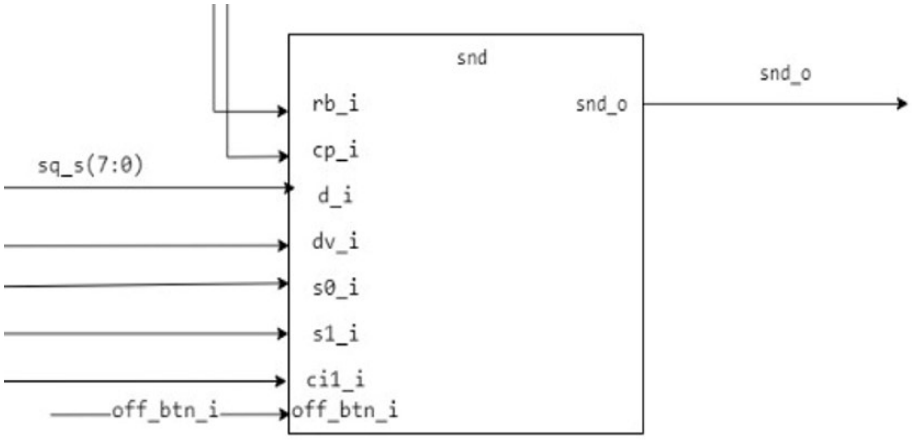

Figure 10: snd Entity

This block is responsible for the sound generation. rb_i is the reset bit. cp_i is the clock pulse. d_i is data which is needed to be transmitted. dv_i is the validity of the data. s0_i

and s1_i are responsible for the mode selection. ci1_i represents the one second pulse. snd_o is the final sound output.

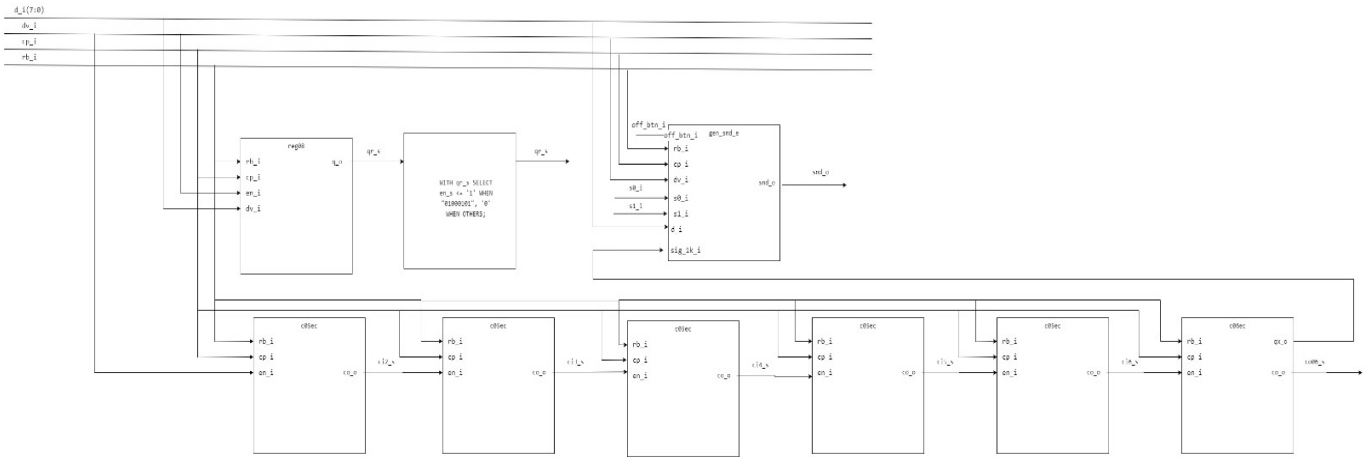

Figure 11: The architecture of the Sound Component

Sound Component: In this Block Diagram, c05ec and co6ec are used to generate the 1kHz signal. Thereafter, all the signals are driven to the gen_snd_e. This block consists of an FSM and generates the sound depending on the selected mode. The FSM has been given below.

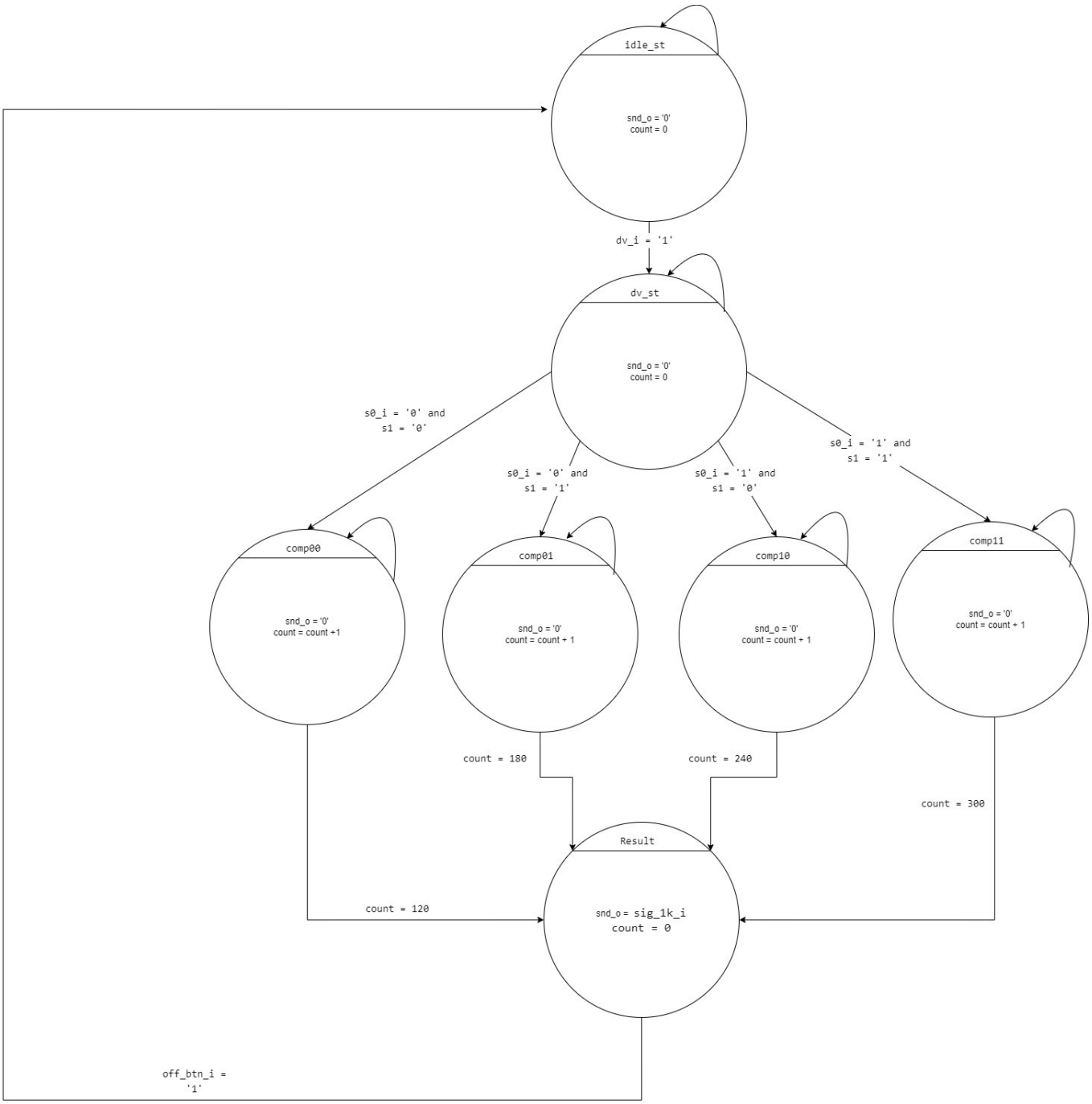

Figure 12: The Finite State Machine of the Sound Component

Sound FSM: When the FSM starts, it is in ideal state. When dv_i signal is detected, the transition occurs to state dv_st. Now depending on the mode selected, it correspondingly goes to comp00 or comp01 or comp10 or comp11. For example, the mode for 2 minutes is selected, then it is directed towards the comp00 state. It waits in that respective state until the count is equal to 120. Count is constantly getting incremented at every second pulse. So 120 second pulses mean 2 minutes. Thereafter, it is driven to the result state where the sound is generated. Once the OFF button is pressed, then only it goes to the ideal state, otherwise the sound is constantly generated.

References

[1] <https://en.wikipedia.org/wiki/Field-programmable_gate_array>

[2] <https://en.wikipedia.org/wiki/VHDL>