Data Acquisition System using an FSMD Circuit

Daniel Josué Rodríguez Agraz
Communications and Electronics Enginnering, Unniversidad de Guadalajara

***Abstract*—This document outlines the design and verification process of a data acquisition system (DAQ) using SystemVerilog. The DAQ system incorporates an Analog-to-Digital Converter (ADC), a 32x256 memory unit for data storage, an 8-bit address counter to manage memory write operations, and a state machine to orchestrate the interaction between these modules. The design and verification were conducted using the SystemVerilog hardware description language, with simulation and compilation performed in Questasim software. The design implementation follows a state machine approach. Additionally, a test bench was developed to verify each instance and the overall design, demonstrating the successful implementation of the circuit.**

This document focuses on the design and application of state machines and their significance in the digital design industry, where they are a critical tool for implementing control logic. State machines provide a structured way to describe the behavior of a system with a finite number of states and the transitions between them, ensuring efficient control of operational flow.

I. Objectives

- Use Verilog to design and simulate a data acquisition system.
- Implement a finite state machine for circuit design.
- Design a test bench for the circuit verification.

II. Introduction

A state machine is a logical system that progresses through a sequence of states defined by internal logic and external inputs.[1] Sequential systems typically consist of inputs, outputs, and internal states. Two primary models are used to classify sequential circuits: the Mealy model and the Moore model.[2]

Mealy Model: In a Mealy state machine, the outputs depend on both the current state and the current inputs. This model often results in faster response times, as changes in inputs can immediately affect the outputs without waiting for a state transition.[2]

Moore Model: In a Moore state machine, the outputs depend solely on the current state and not directly on the inputs. As a result, the outputs only change on state transitions, which makes the design simpler and more predictable, though potentially slower to respond to input changes.[2]

The Figure 1 shows a diagram of the Data Acquisition System (SAD) module that will be implemented. The design of the Sample/Hold module and the ADC will be omitted, as they are outside the scope of this project. The state machine will control the storage of the data coming from the ADC module. Every time the ADC completes a conversion, the finite state machine (FSM) will enable the memory for the result to be stored, then increment the counter to set the next address for the subsequent data.

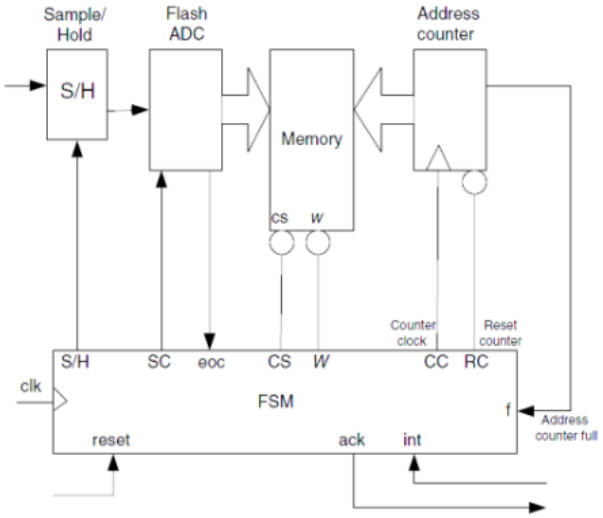

Fig. 1

In this project, a Mealy state machine is implemented to ensure proper data flow from the ADC to the memory. A crucial aspect of state machines is the state diagram, which illustrates state transitions based on inputs. As shown in Figure 2, the next state is determined by the current state and the inputs, which act as conditionals to control transitions between states. For instance, in the transition from the sixth state to the seventh state, the value of the FAC input plays a critical role.

III. Methodology

For the design we will implement a Questasim project and will use systemVerilog languaje to achieve the design and verification.

The first step involves implementing a memory module and a counter. In Verilog, a memory module can be created by defining a table of registers with the desired specifications. In this case, we will design a $32256 - bit$ memory.

The memory operation is controlled by an enable input. When this input is set high, the system can perform either a write or read operation based on the state of the WR input. When the WR input is low, reading is enabled, and the output will reflect the data stored at the address specified by the input address. Conversely, when the WR input is high, the data present at the $data_in$ input will be stored in the specified memory address.

To manage the memory and prevent overflow, as well as to generate a signal indicating to the state machine when the memory is full, a counter will be implemented. This counter will sequentially point to the corresponding address in the

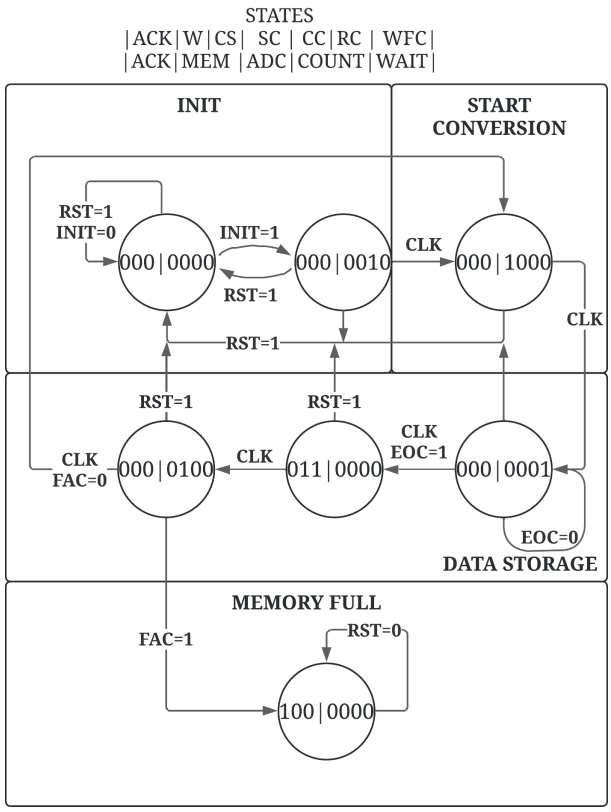

Fig. 2: State diagram for a FSM Data acquisition system.

memory and will increment after each write operation. Once the counter reaches its maximum value, an overflow flag will be activated to inform the state machine that the memory is full.

The data to be stored in the memory will come from an ADC module, the design of which is outside the scope of this project and will be omitted. Its corresponding signals will be emulated in the test bench.

With these modules tested and operational, the next step is to design the state machine. For this, we will use the state diagram shown in Figure 2. This diagram contains seven states divided into three stages.

The first stage is initialization. After a reset, all outputs will be set to zero. Once the circuit receives a pulse on the $init$ signal, it will initialize the entire system.

First, the counter will be set to zero to avoid starting from a different address. Then, the start conversion signal will send a pulse to the ADC to begin sampling. While the ADC is busy, the state machine will be in standby mode. Once the ADC finishes the conversion, it will set the end-of-conversion (EOC) flag high, which will initiate the storage stage.

In this stage, the memory will be enabled and set to write using the en and WR outputs. After saving the data, the counter will increment and point to the next memory address. This process will continue until the memory is full, at which point the full address flag will be set. Finally, the system will set the ack high and will be waiting for the init signal to start again.

IV. Results

The FIgure 3 shows the states after a reset. It can be noticed that after a reset we send an initialization pulse. When the rising edge of the init enters, the counter reset output is set to high to restart the counter, which can be observed because just after that it is set to zero. After resetting the counter, we send the signal to the ADC to start the conversion, which lasts 11 cycles; during that time, we can se the waiting flag is set high, this signal was added to have a stand by state, which makes the design of the state machine easier. Once we receive the end of conversion pulse, the memory is enabled and the data is written, which can be seen in the data in signal, then the counter is incremented and the start conversion signal is set high for a new conversio, which can be seen in the Figure 4.

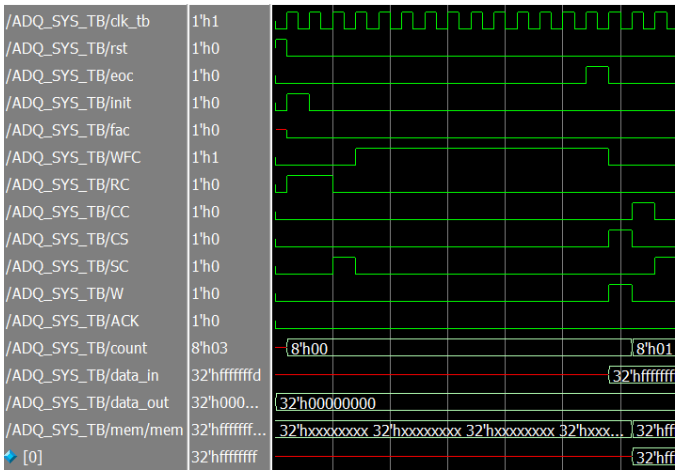

Fig. 3

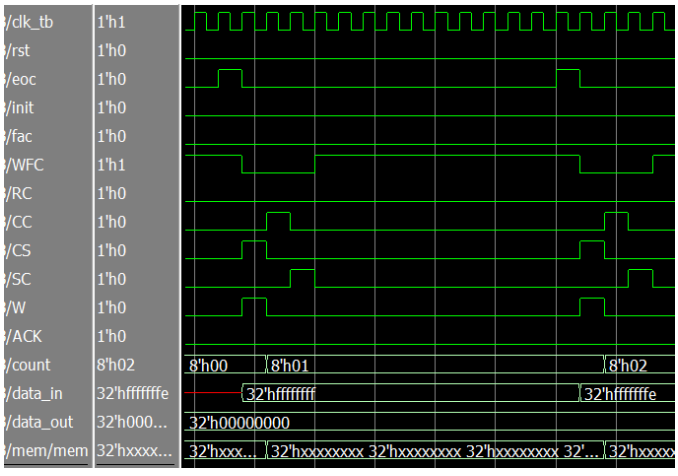

Fig. 4: Second and third readings.

In the Figure 5 is shown how the memory starts gradually filling after the writing is enabled.

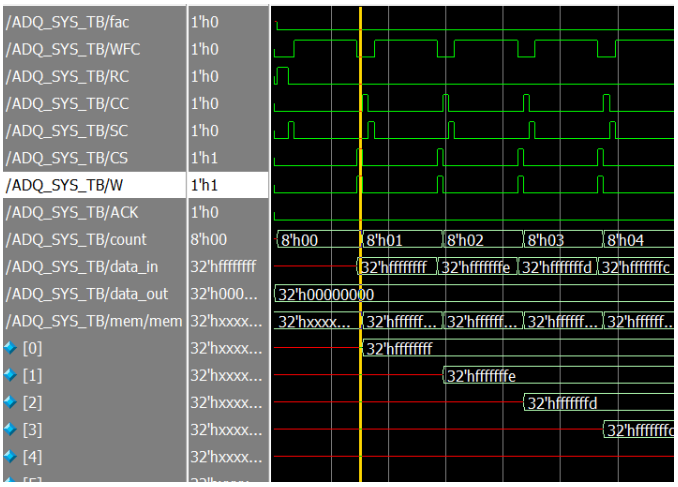

Fig. 5

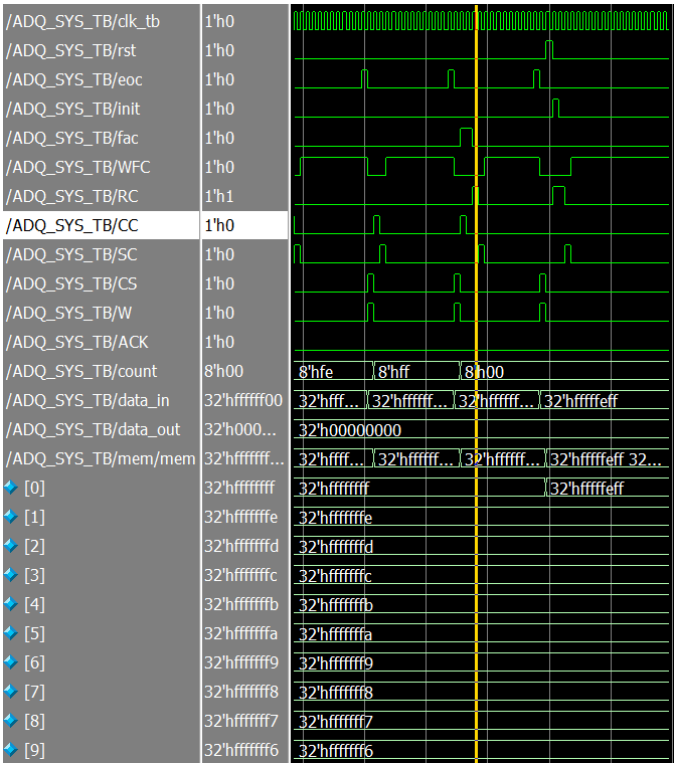

Fig. 6

In the Figure 7 once the counter is full, the ack signal is set to high, and doesnt change until a reset signal or an init sinal activate it.

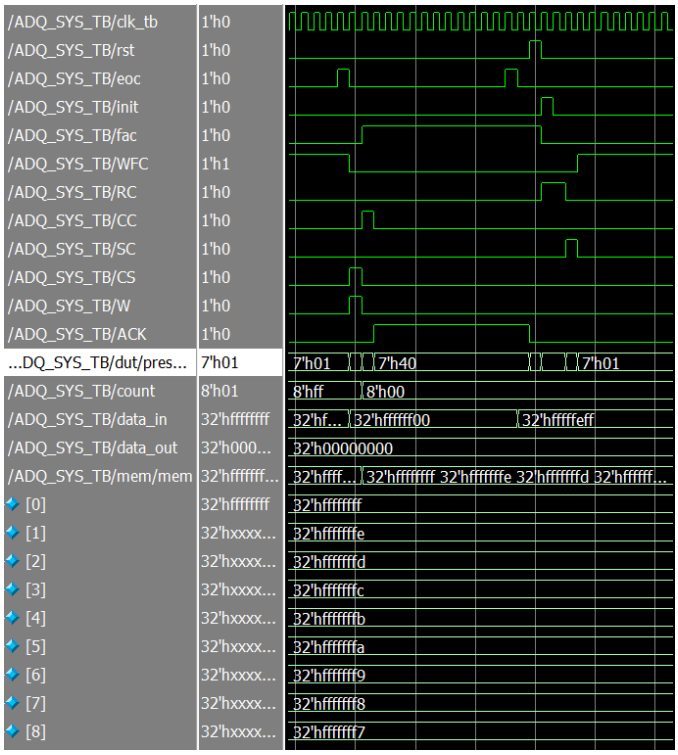

Fig. 7

For the test bench a loop was used to go over every memory address, so by the end of the verification we can make sure the memory is filling up adequately. The data being saved is the subtraction of $'hFFFFFFFF$ and the variable being used in the loop. The Figure **??** shows the state of the memory at the end of the verification.

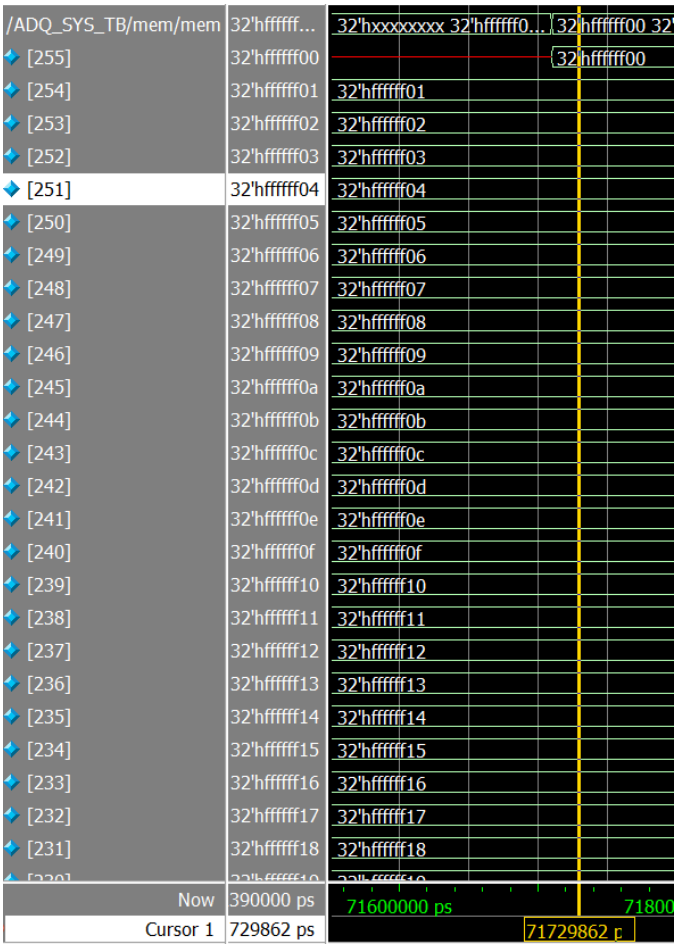
/ADQ_SYS_TB/mem/mem
32'hffffff...
32'hxxxxxxxx 32'hffffff0...
32'hffffff00 32'
[255]
32'hffffff00
[254]
32'hffffff01
[253]
32'hffffff02
[252]
32'hffffff03
[251]
32'hffffff04
[250]
32'hffffff05
[249]
32'hffffff06
[248]
32'hffffff07
[247]
32'hffffff08
[246]
32'hffffff09
[245]
32'hffffff0a
[244]
32'hffffff0b
[243]
32'hffffff0c
[242]
32'hffffff0d
[241]
32'hffffff0e
[240]
32'hffffff0f
[239]
32'hffffff10
[238]
32'hffffff11
[237]
32'hffffff12
[236]
32'hffffff13
[235]
32'hffffff14
[234]
32'hffffff15
[233]
32'hffffff16
[232]
32'hffffff17
[231]
32'hffffff18
Now
390000 ps
71600000 ps
71800
Cursor 1
729862 ps
71729862 p

Fig. 8

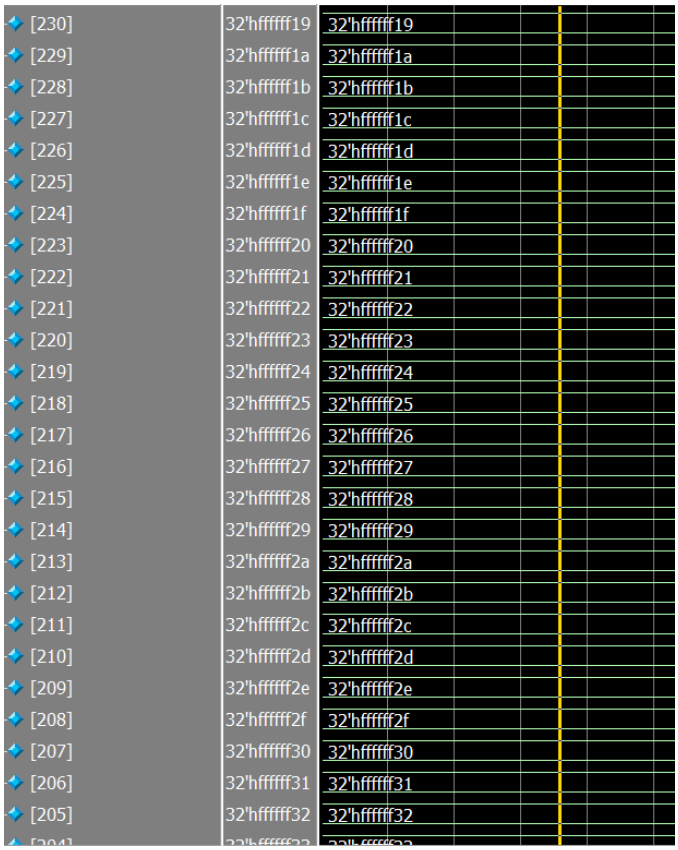
[230]
32'hffffff19
[229]
32'hffffff1a
[228]
32'hffffff1b
[227]
32'hffffff1c
[226]
32'hffffff1d
[225]
32'hffffff1e
[224]
32'hffffff1f
[223]
32'hffffff20
[222]
32'hffffff21
[221]
32'hffffff22
[220]
32'hffffff23
[219]
32'hffffff24
[218]
32'hffffff25
[217]
32'hffffff26
[216]
32'hffffff27
[215]
32'hffffff28
[214]
32'hffffff29
[213]
32'hffffff2a
[212]
32'hffffff2b
[211]
32'hffffff2c
[210]
32'hffffff2d
[209]
32'hffffff2e
[208]
32'hffffff2f
[207]
32'hffffff30
[206]
32'hffffff31
[205]
32'hffffff32

Fig. 9

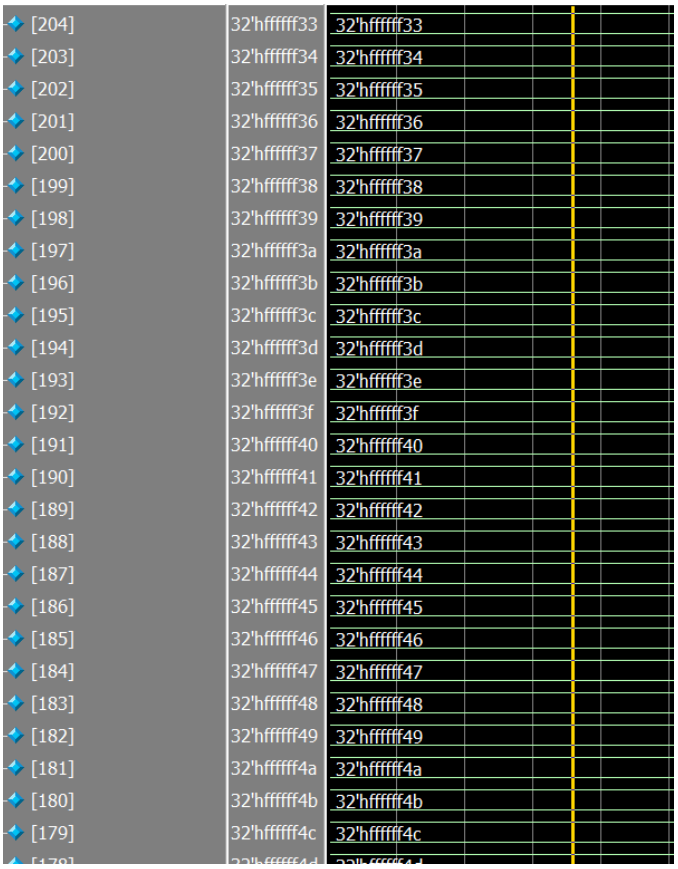
[204] 32'hffffff33 32'hffffff33
[203] 32'hffffff34 32'hffffff34
[202] 32'hffffff35 32'hffffff35
[201] 32'hffffff36 32'hffffff36
[200] 32'hffffff37 32'hffffff37
[199] 32'hffffff38 32'hffffff38
[198] 32'hffffff39 32'hffffff39
[197] 32'hffffff3a 32'hffffff3a
[196] 32'hffffff3b 32'hffffff3b
[195] 32'hffffff3c 32'hffffff3c
[194] 32'hffffff3d 32'hffffff3d
[193] 32'hffffff3e 32'hffffff3e
[192] 32'hffffff3f 32'hffffff3f
[191] 32'hffffff40 32'hffffff40
[190] 32'hffffff41 32'hffffff41
[189] 32'hffffff42 32'hffffff42
[188] 32'hffffff43 32'hffffff43
[187] 32'hffffff44 32'hffffff44
[186] 32'hffffff45 32'hffffff45
[185] 32'hffffff46 32'hffffff46
[184] 32'hffffff47 32'hffffff47
[183] 32'hffffff48 32'hffffff48
[182] 32'hffffff49 32'hffffff49
[181] 32'hffffff4a 32'hffffff4a
[180] 32'hffffff4b 32'hffffff4b
[179] 32'hffffff4c 32'hffffff4c

Fig. 10

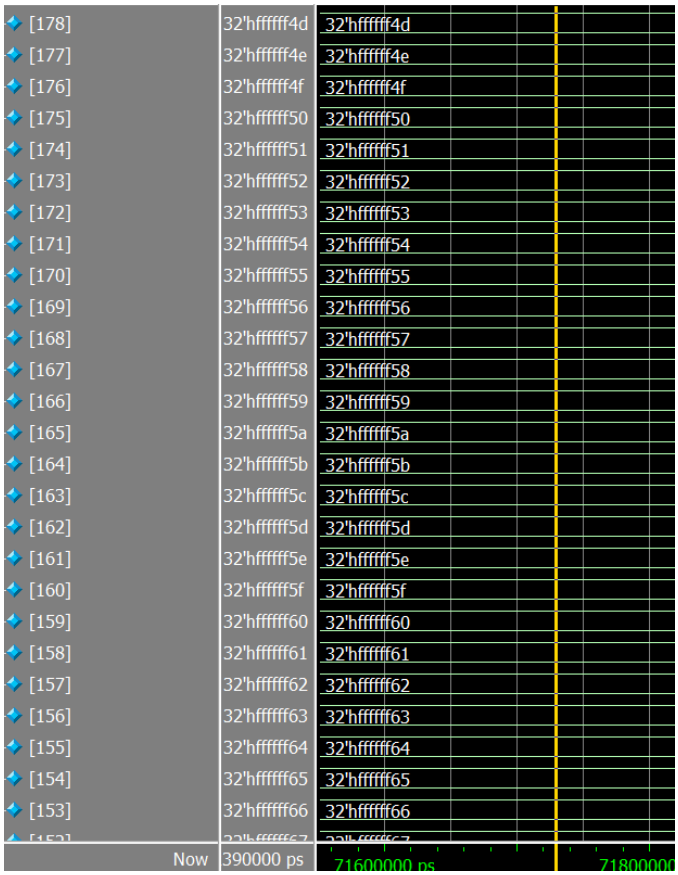
[178] 32'hffffff4d 32'hffffff4d
[177] 32'hffffff4e 32'hffffff4e
[176] 32'hffffff4f 32'hffffff4f
[175] 32'hffffff50 32'hffffff50
[174] 32'hffffff51 32'hffffff51
[173] 32'hffffff52 32'hffffff52
[172] 32'hffffff53 32'hffffff53
[171] 32'hffffff54 32'hffffff54
[170] 32'hffffff55 32'hffffff55
[169] 32'hffffff56 32'hffffff56
[168] 32'hffffff57 32'hffffff57
[167] 32'hffffff58 32'hffffff58
[166] 32'hffffff59 32'hffffff59
[165] 32'hffffff5a 32'hffffff5a
[164] 32'hffffff5b 32'hffffff5b
[163] 32'hffffff5c 32'hffffff5c
[162] 32'hffffff5d 32'hffffff5d
[161] 32'hffffff5e 32'hffffff5e
[160] 32'hffffff5f 32'hffffff5f
[159] 32'hffffff60 32'hffffff60
[158] 32'hffffff61 32'hffffff61
[157] 32'hffffff62 32'hffffff62
[156] 32'hffffff63 32'hffffff63
[155] 32'hffffff64 32'hffffff64
[154] 32'hffffff65 32'hffffff65
[153] 32'hffffff66 32'hffffff66
Now 390000 ps

Fig. 11

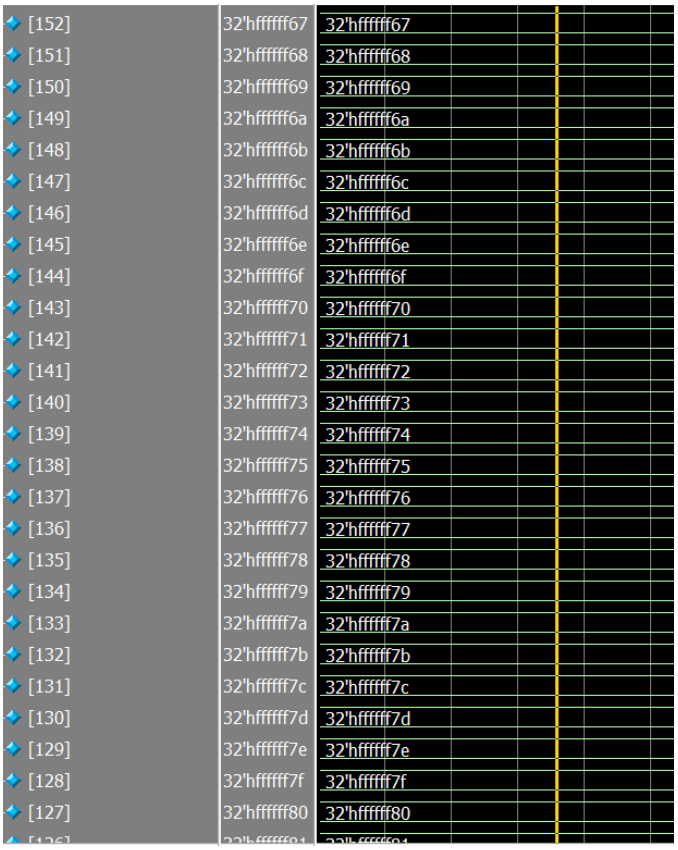
[152] 32'hffffff67 32'hffffff67
[151] 32'hffffff68 32'hffffff68
[150] 32'hffffff69 32'hffffff69
[149] 32'hffffff6a 32'hffffff6a
[148] 32'hffffff6b 32'hffffff6b
[147] 32'hffffff6c 32'hffffff6c
[146] 32'hffffff6d 32'hffffff6d
[145] 32'hffffff6e 32'hffffff6e
[144] 32'hffffff6f 32'hffffff6f
[143] 32'hffffff70 32'hffffff70
[142] 32'hffffff71 32'hffffff71
[141] 32'hffffff72 32'hffffff72
[140] 32'hffffff73 32'hffffff73
[139] 32'hffffff74 32'hffffff74
[138] 32'hffffff75 32'hffffff75
[137] 32'hffffff76 32'hffffff76
[136] 32'hffffff77 32'hffffff77
[135] 32'hffffff78 32'hffffff78
[134] 32'hffffff79 32'hffffff79
[133] 32'hffffff7a 32'hffffff7a
[132] 32'hffffff7b 32'hffffff7b
[131] 32'hffffff7c 32'hffffff7c
[130] 32'hffffff7d 32'hffffff7d
[129] 32'hffffff7e 32'hffffff7e
[128] 32'hffffff7f 32'hffffff7f
[127] 32'hffffff80 32'hffffff80

Fig. 12

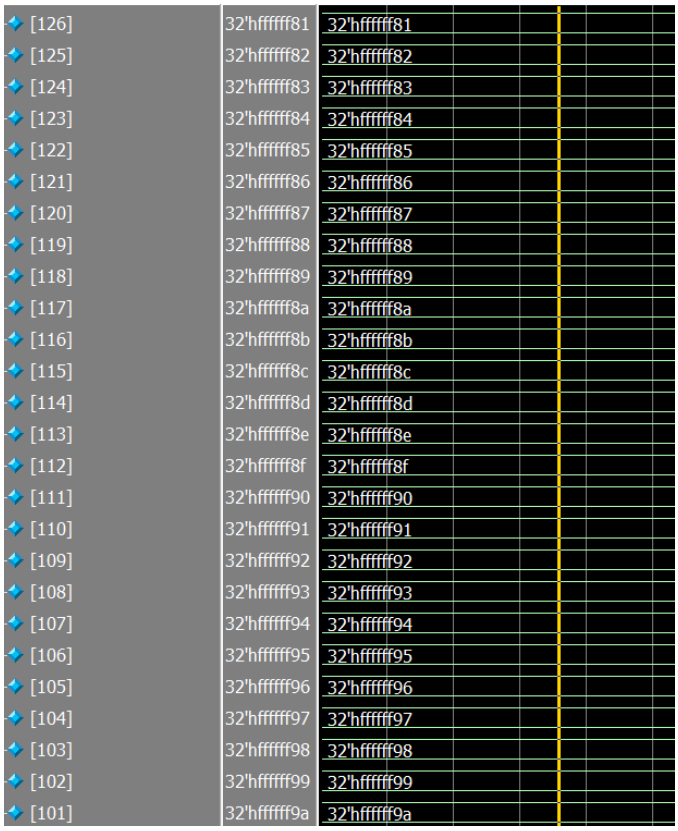
[126] 32'hffffff81 32'hffffff81
[125] 32'hffffff82 32'hffffff82
[124] 32'hffffff83 32'hffffff83
[123] 32'hffffff84 32'hffffff84
[122] 32'hffffff85 32'hffffff85
[121] 32'hffffff86 32'hffffff86
[120] 32'hffffff87 32'hffffff87
[119] 32'hffffff88 32'hffffff88
[118] 32'hffffff89 32'hffffff89
[117] 32'hffffff8a 32'hffffff8a
[116] 32'hffffff8b 32'hffffff8b
[115] 32'hffffff8c 32'hffffff8c
[114] 32'hffffff8d 32'hffffff8d
[113] 32'hffffff8e 32'hffffff8e
[112] 32'hffffff8f 32'hffffff8f
[111] 32'hffffff90 32'hffffff90
[110] 32'hffffff91 32'hffffff91
[109] 32'hffffff92 32'hffffff92
[108] 32'hffffff93 32'hffffff93
[107] 32'hffffff94 32'hffffff94
[106] 32'hffffff95 32'hffffff95
[105] 32'hffffff96 32'hffffff96
[104] 32'hffffff97 32'hffffff97
[103] 32'hffffff98 32'hffffff98
[102] 32'hffffff99 32'hffffff99
[101] 32'hffffff9a 32'hffffff9a

Fig. 13

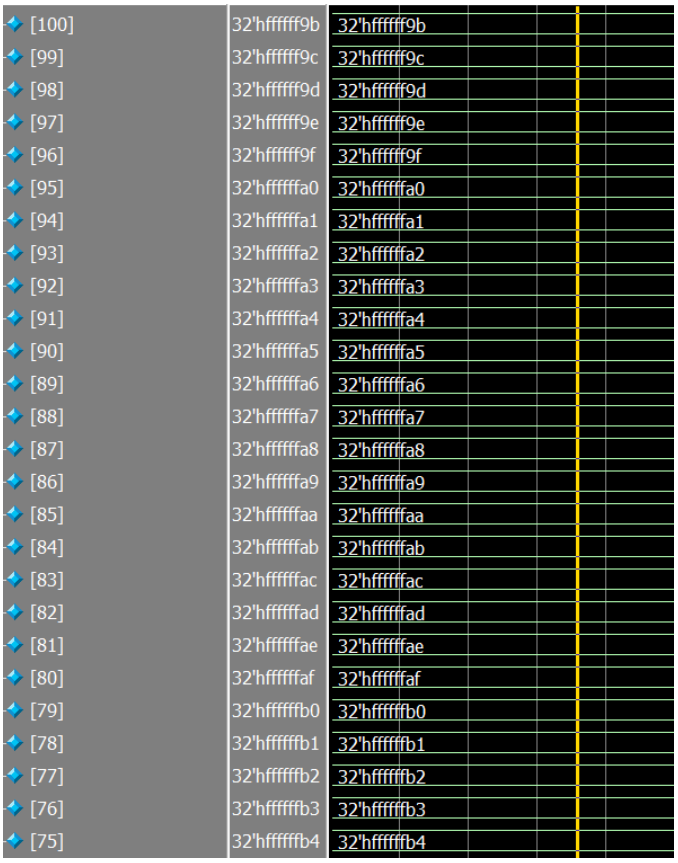
[100] 32'hffffff9b 32'hffffff9b
[99] 32'hffffff9c 32'hffffff9c
[98] 32'hffffff9d 32'hffffff9d
[97] 32'hffffff9e 32'hffffff9e
[96] 32'hffffff9f 32'hffffff9f
[95] 32'hffffffa0 32'hffffffa0
[94] 32'hffffffa1 32'hffffffa1
[93] 32'hffffffa2 32'hffffffa2
[92] 32'hffffffa3 32'hffffffa3
[91] 32'hffffffa4 32'hffffffa4
[90] 32'hffffffa5 32'hffffffa5
[89] 32'hffffffa6 32'hffffffa6
[88] 32'hffffffa7 32'hffffffa7
[87] 32'hffffffa8 32'hffffffa8
[86] 32'hffffffa9 32'hffffffa9
[85] 32'hffffffaa 32'hffffffaa
[84] 32'hffffffab 32'hffffffab
[83] 32'hffffffac 32'hffffffac
[82] 32'hffffffad 32'hffffffad
[81] 32'hffffffae 32'hffffffae
[80] 32'hffffffaf 32'hffffffaf
[79] 32'hffffffb0 32'hffffffb0
[78] 32'hffffffb1 32'hffffffb1
[77] 32'hffffffb2 32'hffffffb2
[76] 32'hffffffb3 32'hffffffb3
[75] 32'hffffffb4 32'hffffffb4

Fig. 14

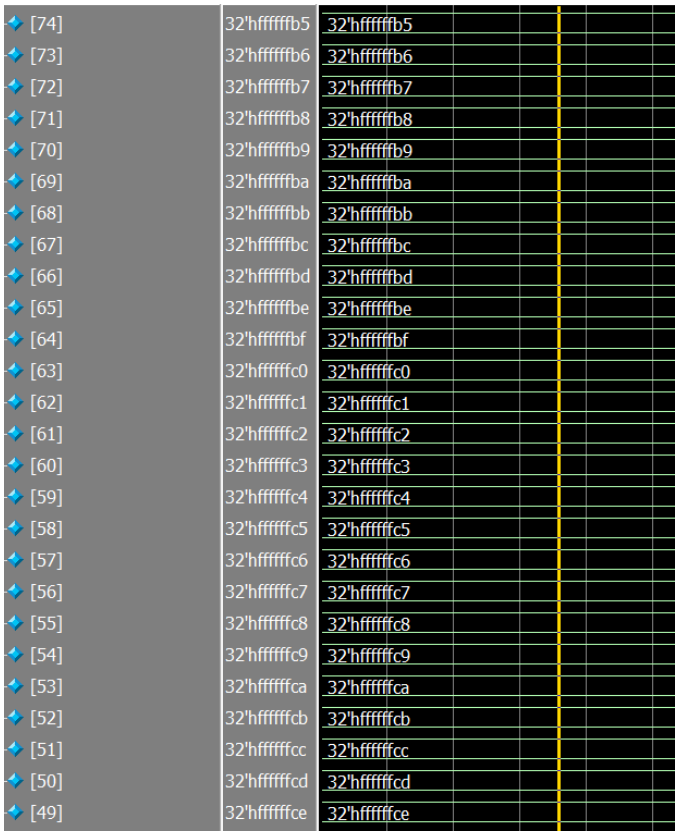
[74] 32'hffffffb5 32'hffffffb5
[73] 32'hffffffb6 32'hffffffb6
[72] 32'hffffffb7 32'hffffffb7
[71] 32'hffffffb8 32'hffffffb8
[70] 32'hffffffb9 32'hffffffb9
[69] 32'hffffffba 32'hffffffba
[68] 32'hffffffbb 32'hffffffbb
[67] 32'hffffffbc 32'hffffffbc
[66] 32'hffffffbd 32'hffffffbd
[65] 32'hffffffbe 32'hffffffbe
[64] 32'hffffffbf 32'hffffffbf
[63] 32'hffffffc0 32'hffffffc0
[62] 32'hffffffc1 32'hffffffc1
[61] 32'hffffffc2 32'hffffffc2
[60] 32'hffffffc3 32'hffffffc3
[59] 32'hffffffc4 32'hffffffc4
[58] 32'hffffffc5 32'hffffffc5
[57] 32'hffffffc6 32'hffffffc6
[56] 32'hffffffc7 32'hffffffc7
[55] 32'hffffffc8 32'hffffffc8
[54] 32'hffffffc9 32'hffffffc9
[53] 32'hffffffca 32'hffffffca
[52] 32'hffffffcb 32'hffffffcb
[51] 32'hffffffcc 32'hffffffcc
[50] 32'hffffffcd 32'hffffffcd
[49] 32'hffffffce 32'hffffffce

Fig. 15

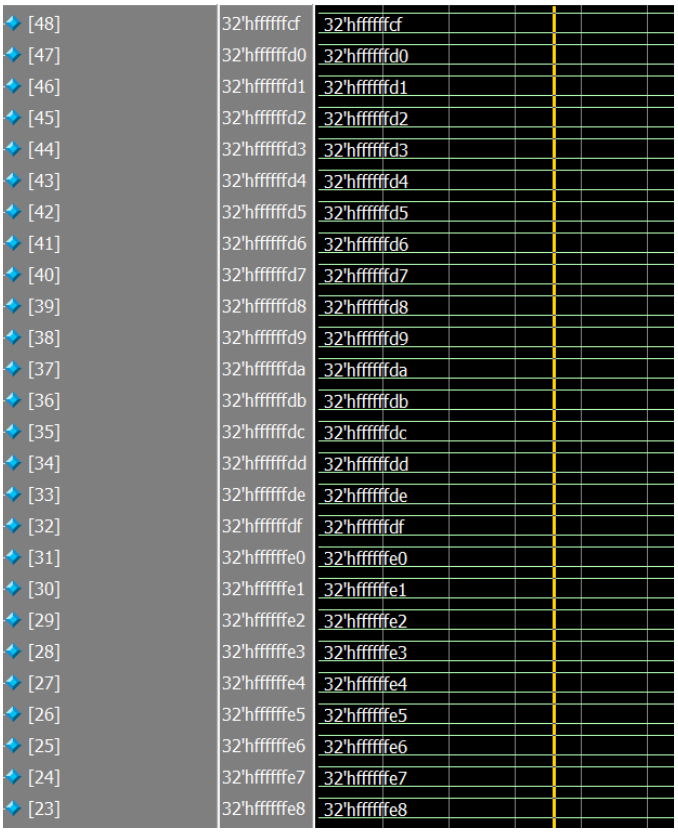

Fig. 16

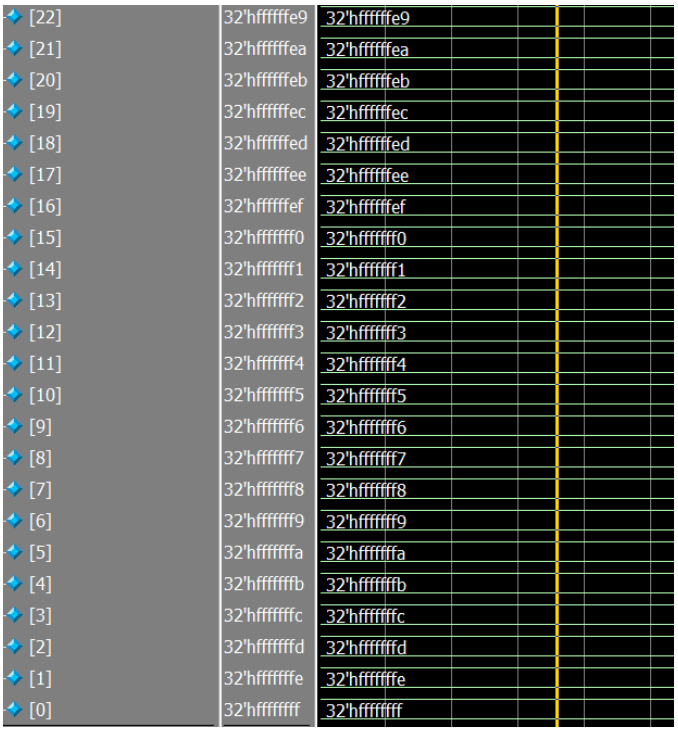

Fig. 17

V. Conclusions

Through this project, we successfully designed and implemented a data acquisition system using the SystemVerilog hardware description language. The design leveraged a state machine to control the data flow from an ADC circuit to a memory unit, where the ADC results were stored. Once the memory was full, an acknowledge signal was activated, prompting the system to wait for a reset before restarting. The project also included the successful design and verification of an 8-bit counter and a 32x256 memory.

The state machine, crucial to managing the overall system flow, was implemented and tested alongside the memory and counter modules. In the test bench, these three modules were interconnected and thoroughly verified through simulations using Questasim, ensuring correct operation under various conditions.

This approach, coupled with the implementation of a comprehensive testbench, demonstrated the effectiveness and reliability of the design.

References

[1] T. L. Floyd, *Fundamentos de Sistemas Digitales*, 9th ed.
[2] M. M. Mano, *Digital design*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2002.