-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

ECSE 222- DIGITAL LOGIC

LABORATORY REPORT #2:

g12_7_Segment_Decoder Circuit Summary and Functionality

-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

TABLE OF CONTENTS

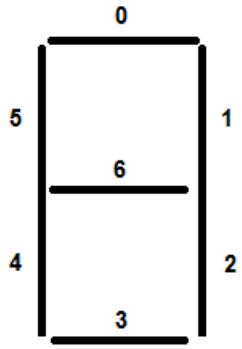
-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

INTRODUCTION

The goal of the second lab was to design a decoder circuit which converts 4 inputs signals (and 1 mode signal) into 7 signals which control a 7-segment display. Such a circuit has active-low outputs; when the mode is 0, the 7-segment display will represent hexadecimal digits (0-9 and capital letters A-F). While the mode is 1, it will show various symbolic characters. This report will describe in detail how such a circuit was implemented, showing all inputs, outputs and a logic-gate analysis of the circuit. It will also describe how the design of this circuit tested and proved to be functional, using Quartus II's waveform simulation tool for VHDL code.

PART 1 : g12_7_Segment_Decoder

DESIGN (I/O)



The circuit has 2 inputs, a 4-bit vector *code*, and 1-bit *mode.* As described in the introduction, the purpose of the *mode* signal is to dictate which type of values will be displayed by a 7-segment display (hexadecimal or symbolic). The output of the circuit is a 7-bit vector *segment_out.* Each bit represents one segment of the 7-segment display. The MSB is thus bit 6, and the LSB is bit 0.

FIGURE 1: 7-Segment Display and *segment_out(6:0)* Correspondence (Clark)

An internal signal *xcode*, combines code and mode into a 5-bit vector, where mode is the LSB.

-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

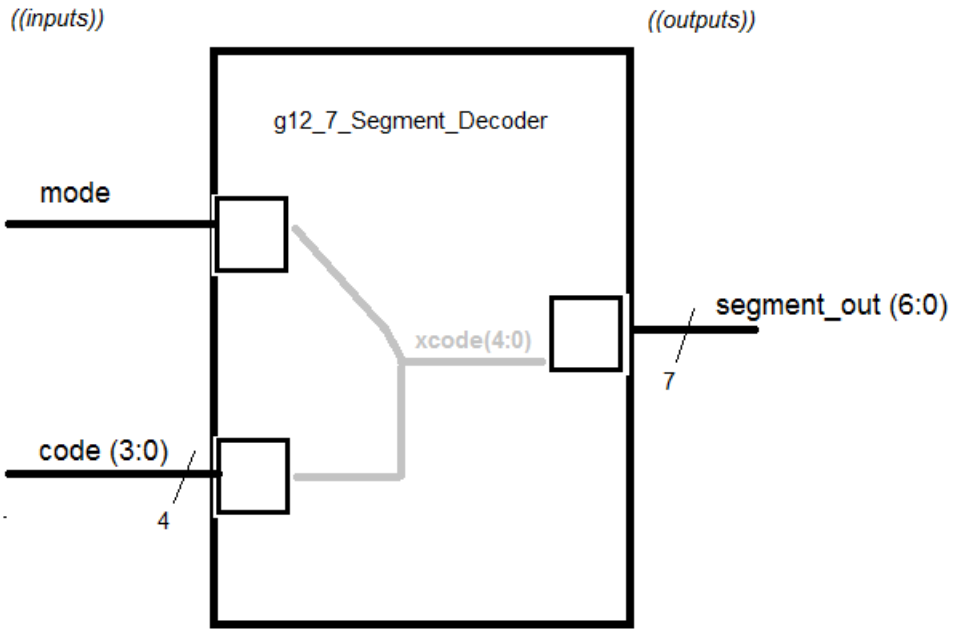


FIGURE 2: Pinout Diagram of g12_7_Segment_Decoder

FUNCTIONALITY

An output segment is active (lit up, if it was a physical LED display) when the output *segment_out(n)*'s value is 0 and is dormant when the *segment_out*(n) is 1. This is called an active-low output (Clark). Below is a list of characters represented by the 7-segment display, depending on the value of the *mode* and of the *code* (shown in decimal instead of binary).

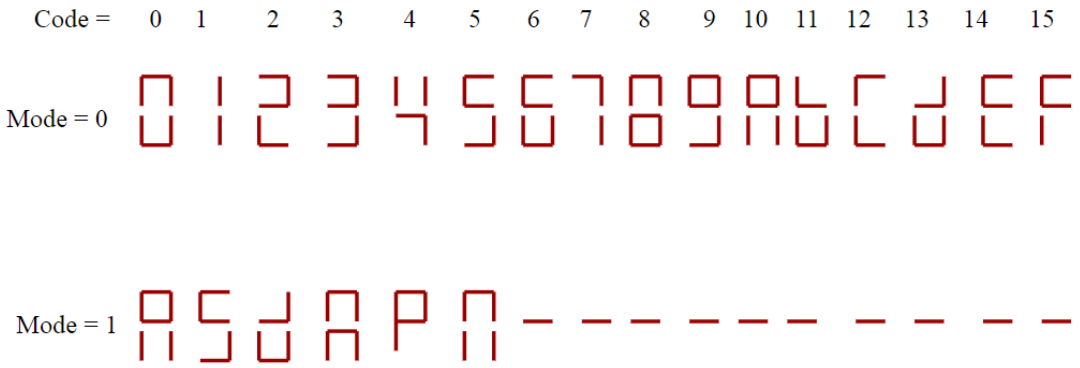


FIGURE 3: 7-Segment Display Values (Clark)

-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

The output segments' state corresponding to every possible combination of code and mode (32 in total) can be seen in Table 1 and Table 2.

code				mode	segments (6 - 0)						
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	1	1	1	1	0	0	1
0	0	1	0	0	0	1	0	0	1	0	0
0	0	1	1	0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1	1	0	0	1
0	1	0	1	0	0	0	1	0	0	1	0
0	1	1	0	0	0	0	0	0	0	1	0
0	1	1	1	0	1	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	1	0	0	0
1	0	1	1	0	0	0	0	0	0	1	1
1	1	0	0	0	1	0	0	0	1	1	0
1	1	0	1	0	0	1	0	0	0	0	1
1	1	1	0	0	0	0	0	0	1	1	0
1	1	1	1	0	0	0	0	1	1	1	0

TABLE 1: 7-Segment Display Truth Table for mode = 0 (by inspection of Figure 1 and Figure 3)

code				mode	segments (6 - 0)						
0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	1	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	0	0	0	0	1
0	0	1	1	1	0	0	0	1	0	0	0
0	1	0	0	1	0	0	0	1	1	0	0
0	1	0	1	1	1	0	0	1	0	0	0
0	1	1	0	1	0	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
1	0	1	0	1	0	1	1	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1	1
1	1	0	0	1	0	1	1	1	1	1	1
1	1	0	1	1	0	1	1	1	1	1	1
1	1	1	0	1	0	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1	1

TABLE 2: 7-Segment Display Truth Table for mode = 1 (by inspection of Figure 1 and Figure 3)

Our circuit was implemented by making use of the *with*, *select* and *when* keywords in our VHDL file. The *with* and *when* keywords act as an iterator for every

-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

possible combination of the xcode vector which contains the code and mode values. For every iteration, the *select* keyword maps it to a 7-bit vector.

The RTL viewer allowed us to generate a gate-level schematic of our circuit, shown in ANNEXE A.

SIMULATION PROCEDURE EXPLANATION

To test if the circuit was adequate, we ran a waveform simulation of the circuit using Altera Quartus II software. The input vector code was set to increase by 1 every 60ns, thus providing a simulation for every single combination from 0000 to 1111. To match every possible combination of the code vector, which increased by 1 every 60 ns, to the output, the mode bit was set to a period of 1920 ns (= 60 * 32).

SIMULATION OF CIRCUIT

For the sake of clarity, the waveform simulation is shown in two parts: mode = 0 and mode = 1. As can be observed in Figure 4 (mode = 0), there are 16 possible code combinations, which are each mapped to a 7-bit vector representing the 7 segments' state. These represent the characters from 0 to F.

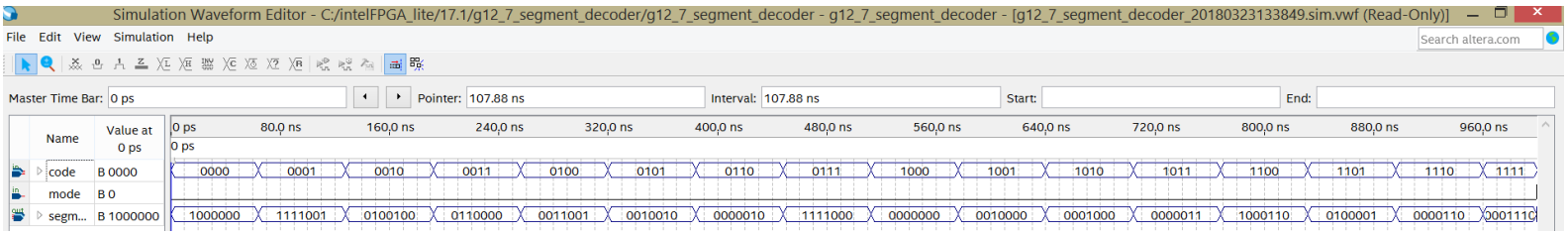


FIGURE 4: Waveform Simulation of g12_7_Segment_Decoder when mode=0

Similarly, the waveform simulation shown in Figure 5 (mode = 1) represents the mapping of the code vector to the symbolic characters specified in Figure 3.

-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

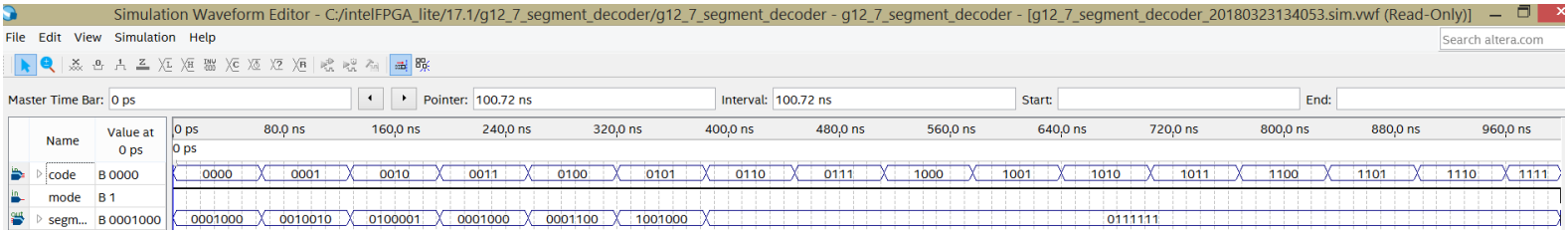


FIGURE 5: Waveform Simulation of g12_7_Segment_Decoder when mode=1

ANALYSIS

To analyze the adequacy of our circuit implementation, we compared the values of the output vectors in our waveform simulation to the values in the truth table in Table 1 and Table 2. All 32 combinations of the values in the code vector and the mode bit in the waveform simulation provided us with outputs that matched those in the truth tables, thus proving that our circuit implementation was adequate.

The implementation of our circuit using seven 32-to-1 multiplexers (one for each segment) using the 5 bits in the xcode vector as selector lines, as shown in ANNEXE A, proved to be accurate.

DISCUSSION

Although our circuit produced the expected outputs and was shown to be accurate, its implementation requires the use of multiplexers with more data lines than necessary. This is a limitation in terms of the speed efficiency of our circuit, since it will add complexity. Other implementations using MUX's with less data lines such as 4-to-1 MUX's are possible (such as the one in Assignment 4). For each segment, a 4-to-1 MUX where the two selector lines can be connected directly to two bits in the xcode vector, and the data lines can represent the simplification of the minterms or maxterms in terms of the unassigned bits in xcode for each segment, perform more efficiently.

-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

References

Clark, J. *“Lab#2 – Combinational Circuit Design with VHDL”,* 2018.

-- Alexandra Zajda (260807517)
-- Nicolas Barreyro (260730549)
-- Gorkem Yalcinoz (260710053)
-- GROUP 12

ANNEXE A (MUX g12_7_Segment_Decoder CIRCUIT)
*obtained used RTL Viewer