Gray Code Generator and Decoder

by
Carsten Kristiansen
Napier University

November 2004

Title page

• **Author:**	Carsten Kristiansen.
• **Napier No:**	04007712.
• **Assignment title:**	Design of a Gray Code Generator and Decoder.
• **Education:**	Electronic and Computer Engineering.
• **Module:**	Electronic Systems SE32102.
• **Place of education:**	Napier University Edinburgh 10 Colinton Road Edinburgh EH10 5DT
• **Lecturer:**	Jay Hoy.
• **Assignment period:**	27. October 2004 - 3. December 2004.

1. Abstract

This report describes how an Gray code generator and decoder are designed and simulated. The report takes a beginning in an introduction to what Gray code is, and can be used for in a practical matter of speaking. There is described the design and simulation process separately for the Gray code generator and the Gray code to binary decoder. This reaches a final and complete schematic as well as a simulation.

Contents

2. Introduction

In today's digital electronics, there are many different types of codes. The most used code combination are known as the binary code, which is a code where every bit represents an integer value. There are however also codes where the combination and shifts between a “0” and a “1” that is important like the Gray code. The Gray code are the type that's explained further in this report. It was used for the first time in a telegraph by the French engineer Émile Baudot in 1878, but was not patented before 1953, by the Bell Labs researcher Frank Gray, which also gave the name to the code. The Gray code can best be described as a combination of binary numbers, with the advantage that it only changes one bit at the time in the right sequence. It is possible to make the Gray code up to 4 bits of a maximum, cause more is not a unique sequence. Below in figure 1 it is shown visual how a 4 bit Gray code looks like.

	Position															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
D		●	●			●	●			●	●			●	●	
C			●	●	●	●					●	●	●	●		
B					●	●	●	●	●	●	●	●				
A									●	●	●	●	●	●	●	●

Fig. 1: 4 bit Gray code

The letters A to D represents the 4 bits, where D is the Least Significant Bit (LSB), and A is the Most Significant Bit (MSB). The 4 bit Gray code starts from the position 0 and ends at position 15. All of the dark circles indicates a logical high (“1”), where all the empty boxes indicates a logical zero (“0”). Note that with each shifts in position there are only increased or decreased with one (high) logical level.

2.1. Practical use of the Gray code

Gray codes are mostly used where a mechanical position needs to be converted to a digital signal. This can be very convenient to use with cogwheels, to determine whether it is at the right position or not. For instance, a production machine are running, and there would be cogwheels inside, where one of them has a Gray code scale (shown below in figure 2) mounted, to be read by photo-sensors and some electronics to decode the signals. Now if the cogwheel had to be very precise adjusted in its position, and at some time it would break a cog, so that the machine wouldn't run as desired (the precision of the wheel would be wrong), it would then be possible for some electronic to read from the Gray code scale , that the sequence of the code would be wrong, set of an alarm and stop the machine immediately.

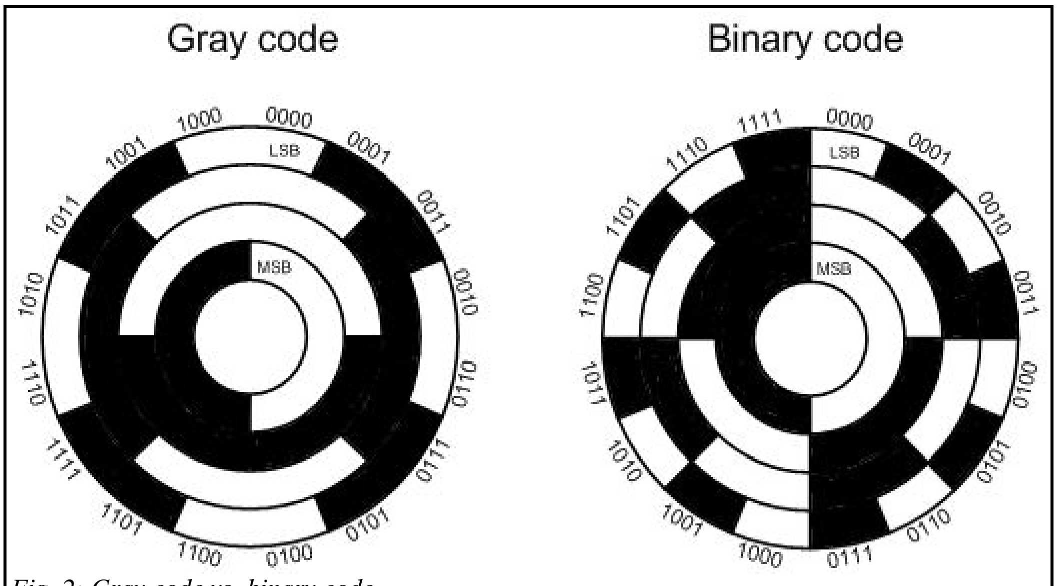


Fig. 2: Gray code vs. binary code

The figure 2 also shows the difference between the Gray code and the binary code, if they where both to be used on a cogwheel. The dark part in the circle are the logical high, and the light are the logical low levels. Again it would not be preferable not to use the binary code, because of the many shifts there would be in the logical levels compared to the Gray code.

3. Assignment specifications

- Use JK flip-flops and suitable logic gates to design a 4-bit binary Gray code generator.
- Use the output of the Gray code generator as inputs to a combinational logic circuit to decode the Gray code to produce the normal binary counting sequence.
- Use TINA or any other simulation package to produce results to verify the design.

3.1. Block diagram

From the specifications a general block diagram can be presented. The block diagram below in figure 3 indicates what the desired options for the assignment are.

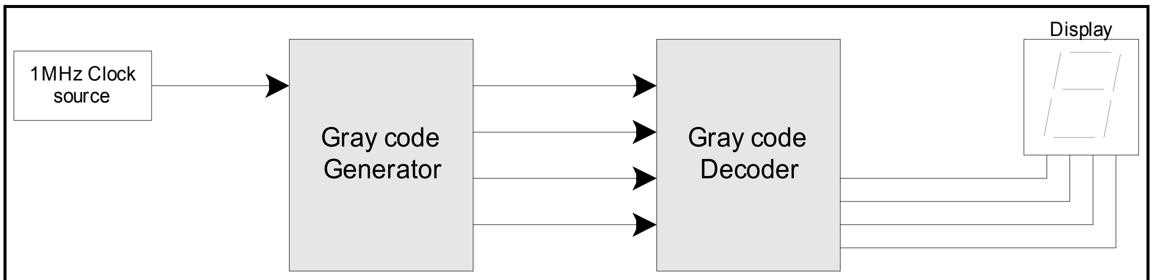


Fig. 3: Gray code generator and decoder block diagram

For the Gray code generator, a clock source is needed. This clock source should generate square pulses at 1 Mhz. The Gray code generator should then be able to send a 4 bit code parallel to the Gray code decoder, where the decoder will convert the Gray code to a 4 bit binary code, which can be used as an output to a display.

The table below indicates the conversion between the Gray code to the binary output. The time sequence for each of the steps are also shown in micro-seconds.

Label	Time [µS]	Gray code				Binary out			
		A	B	C	D	Q_A	Q_B	Q_C	Q_D
0	0-1	0	0	0	0	0	0	0	0
1	1-2	0	0	0	1	0	0	0	1
2	2-3	0	0	1	1	0	0	1	0
3	3-4	0	0	1	0	0	0	1	1
4	4-5	0	1	1	0	0	1	0	0
5	5-6	0	1	1	1	0	1	0	1
6	6-7	0	1	0	1	0	1	1	0
7	7-8	0	1	0	0	0	1	1	1
8	8-9	1	1	0	0	1	0	0	0
9	9-10	1	1	0	1	1	0	0	1
10	10-11	1	1	1	1	1	0	1	0
11	11-12	1	1	1	0	1	0	1	1
12	12-13	1	0	1	0	1	1	0	0
13	13-14	1	0	1	1	1	1	0	1
14	14-15	1	0	0	1	1	1	1	0
15	15-16	1	0	0	0	1	1	1	1

4. Gray code generator

To design the Gray code generator, it is analysed what is needed from the input to the output. By looking at the truth table for the Gray code, it is possible to follow the design procedure, and start with a state transition diagram that shows which steps the generator needs to fulfil. To get the generator running there is a synchronous clock pulse between each step.

4.1. State transition diagram

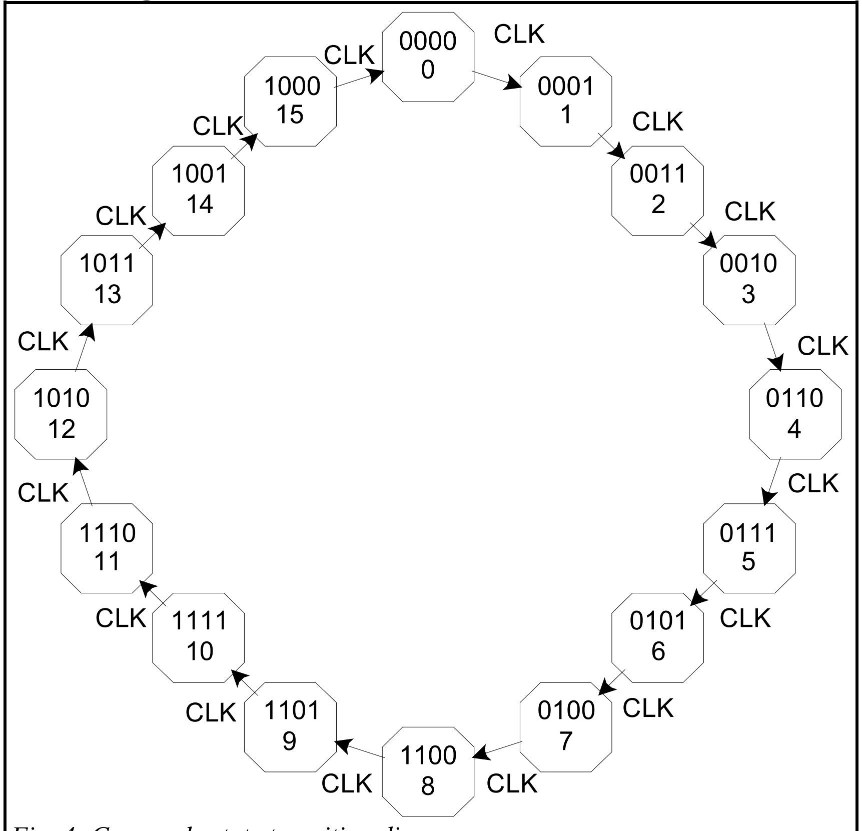


Fig. 4: Gray code state transition diagram

The state transition diagram in figure 4 indicates that there are 4 bits, and 16 steps that needs to be mapped. For this, 4 JK flip-flops are used, one for each bit. A table for the states of this type of flip-flop are shown below. The x's indicates a "don't care" state.

JK Flip-Flop		
Q $\quad Q^+$	J	K
0 → 0	0	X
0 → 1	1	X
1 → 0	X	1
1 → 1	X	0

4.2. State table

A state table can then be made, where it is possible to derive the change state for each of the JK flip-flops. This is done by filling out the next state logic for the Gray code.

Label	Present state				Next state				Change state							
	Q_A	Q_B	Q_C	Q_D	Q_A+	Q_B+	Q_C+	Q_D+	J_A	K_A	J_B	K_B	J_C	K_C	J_D	K_D
0	0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
1	0	0	0	1	0	0	1	1	0	X	0	X	1	X	X	0
2	0	0	1	1	0	0	1	0	0	X	0	X	X	0	X	1
3	0	0	1	0	0	1	1	0	0	X	1	X	X	0	0	X
4	0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
5	0	1	1	1	0	1	0	1	0	X	X	0	X	1	X	0
6	0	1	0	1	0	1	0	0	0	X	X	0	0	X	X	1
7	0	1	0	0	1	1	0	0	1	X	X	0	0	X	0	X
8	1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
9	1	1	0	1	1	1	1	1	X	0	X	0	1	X	X	0
10	1	1	1	1	1	1	1	0	X	0	X	0	X	0	X	1
11	1	1	1	0	1	0	1	0	X	0	X	1	X	0	0	X
12	1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
13	1	0	1	1	1	0	0	1	X	0	0	X	X	1	X	0
14	1	0	0	1	1	0	0	0	X	0	0	X	0	X	X	1
15	1	0	0	0	0	0	0	0	X	1	0	X	0	X	0	X

4.3. Next state logic

From the change state of the state table, the next state logic can be derived and written into Karnaugh maps. The method used for this, is to draw a map for each output. To illustrate better what is needed on the input to get the required output, the logic gates and connections for each of the reduced expressions are also shown to the right of each Karnaugh maps.

$Q_C Q_D$ \ $Q_A Q_B$	0 0	0 1	1 1	1 0
0 0	0	1	X	X
0 1	0	0	X	X
1 1	0	0	X	X
1 0	0	0	X	X

J_A change state

$$J_A = \underline{\underline{Q_B \cdot \overline{Q_C} \cdot \overline{Q_D}}}$$

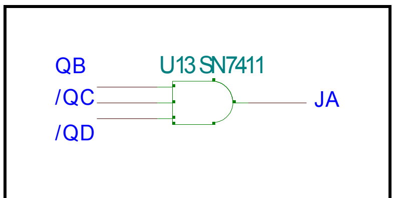


Fig. 5: J_A logic gate

$Q_C Q_D$ \ $Q_A Q_B$	0 0	0 1	1 1	1 0
0 0	X	X	0	1
0 1	X	X	0	0
1 1	X	X	0	0
1 0	X	X	0	0

K_A change state

$$K_A = \underline{\underline{\overline{Q_B} \cdot \overline{Q_C} \cdot \overline{Q_D}}}$$

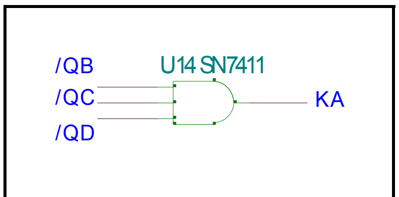


Fig. 6: K_A logic gate

$Q_C Q_D$ \ $Q_A Q_B$	0 0	0 1	1 1	1 0
0 0	0	X	X	0
0 1	0	X	X	0
1 1	0	X	X	0
1 0	1	X	X	0

J_B change state

$$J_B = \overline{Q_A} \cdot Q_C \cdot \overline{Q_D}$$

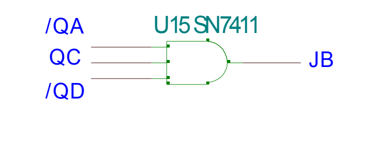


Fig. 7: J_B logic gate

$Q_C Q_D$ \ $Q_A Q_B$	0 0	0 1	1 1	1 0
0 0	X	0	0	X
0 1	X	0	0	X
1 1	X	0	0	X
1 0	X	0	1	X

K_B change state

$$K_B = Q_A \cdot Q_C \cdot \overline{Q_D}$$

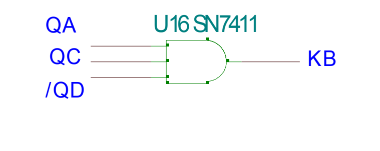


Fig. 8: K_B logic gate

$Q_C Q_D$ \ $Q_A Q_B$	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	1	0	1	0
1 1	X	X	X	X
1 0	X	X	X	X

J_C change state

$$\begin{aligned} J_C &= \overline{Q_A} \cdot \overline{Q_B} \cdot Q_D + Q_A \cdot Q_B \cdot Q_D \\ &= Q_D \cdot (\overline{Q_A} \cdot \overline{Q_B} + Q_A \cdot Q_B) \\ &= Q_D \cdot \overline{(Q_A \oplus Q_B)} \end{aligned}$$

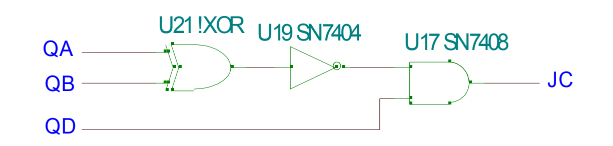


Fig. 9: J_C logic gates

$Q_C Q_D$ \ $Q_A Q_B$	0 0	0 1	1 1	1 0
0 0	X	X	X	X
0 1	X	X	X	X
1 1	0	1	0	1
1 0	0	0	0	0

K_C change state

$$\begin{aligned} K_C &= \overline{Q_A} \cdot Q_B \cdot Q_D + Q_A \cdot \overline{Q_B} \cdot Q_D \\ &= Q_D \cdot (\overline{Q_A} \cdot Q_B + Q_A \cdot \overline{Q_B}) \\ &= Q_D \cdot (Q_A \oplus Q_B) \end{aligned}$$

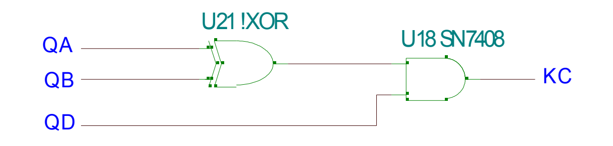


Fig. 10: K_C logic gates

$Q_C\,Q_D$ \ $Q_A\,Q_B$	0 0	0 1	1 1	1 0
0 0	1	0	1	0
0 1	X	X	X	X
1 1	X	X	X	X
1 0	0	1	0	1

J_D change state

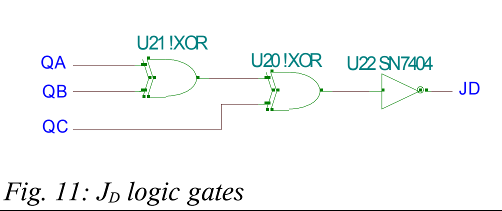


Fig. 11: J_D logic gates

$$\begin{aligned}J_D &= \overline{Q_A}\cdot\overline{Q_B}\cdot\overline{Q_C}+Q_A\cdot Q_B\cdot\overline{Q_C}+\overline{Q_A}\cdot Q_B\cdot Q_C+Q_A\cdot\overline{Q_B}\cdot Q_C\\ &= \overline{Q_C}\cdot(\overline{Q_A}\cdot\overline{Q_B}+Q_A\cdot Q_B)+Q_C\cdot(\overline{Q_A}\cdot Q_B+Q_A\cdot\overline{Q_B})\\ &= \overline{Q_C}\cdot(\overline{Q_A\oplus Q_B})+Q_C(Q_A\oplus Q_B)\\ &= \underline{\underline{\overline{Q_A\oplus Q_B\oplus Q_C}}}\end{aligned}$$

$Q_C\,Q_D$ \ $Q_A\,Q_B$	0 0	0 1	1 1	1 0
0 0	X	X	X	X
0 1	0	1	0	1
1 1	1	0	1	0
1 0	X	X	X	X

K_D change state

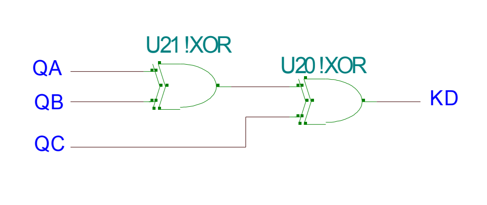


Fig. 12: K_D logic gates

$$\begin{aligned}K_D &= \overline{Q_A}\cdot Q_B\cdot\overline{Q_C}+Q_A\cdot\overline{Q_B}\cdot\overline{Q_C}+\overline{Q_A}\cdot\overline{Q_B}\cdot Q_C+Q_A\cdot Q_B\cdot Q_C\\ &= \overline{Q_C}\cdot(\overline{Q_A}\cdot Q_B+Q_A\cdot\overline{Q_B})+Q_C\cdot(\overline{Q_A}\cdot\overline{Q_B}+Q_A\cdot Q_B)\\ &= \overline{Q_C}\cdot(Q_A\oplus Q_B)+Q_C\cdot(\overline{Q_A\oplus Q_B})\\ &= \underline{\underline{Q_A\oplus Q_B\oplus Q_C}}\end{aligned}$$

Note that the J_C, K_C, J_D and K_D can “share” some of the gates. Meaning that some of the gates can be spared cause they use the same outputs. This is illustrated better in the schematics.

4.4. Schematic

The schematic below in figure 13 shows the diagram from TINA, used for the simulation of the Gray code generator. At each of the JK flip-flop's output, there is a logic indicator connected, which have been used for a step-by-step simulation to indicate if the circuit are working as desired. The clock-output (CLK), sends a 1 MHz clock source synchronously to each of the flip-flop's. This is corresponding to 1 µS for each pulse.

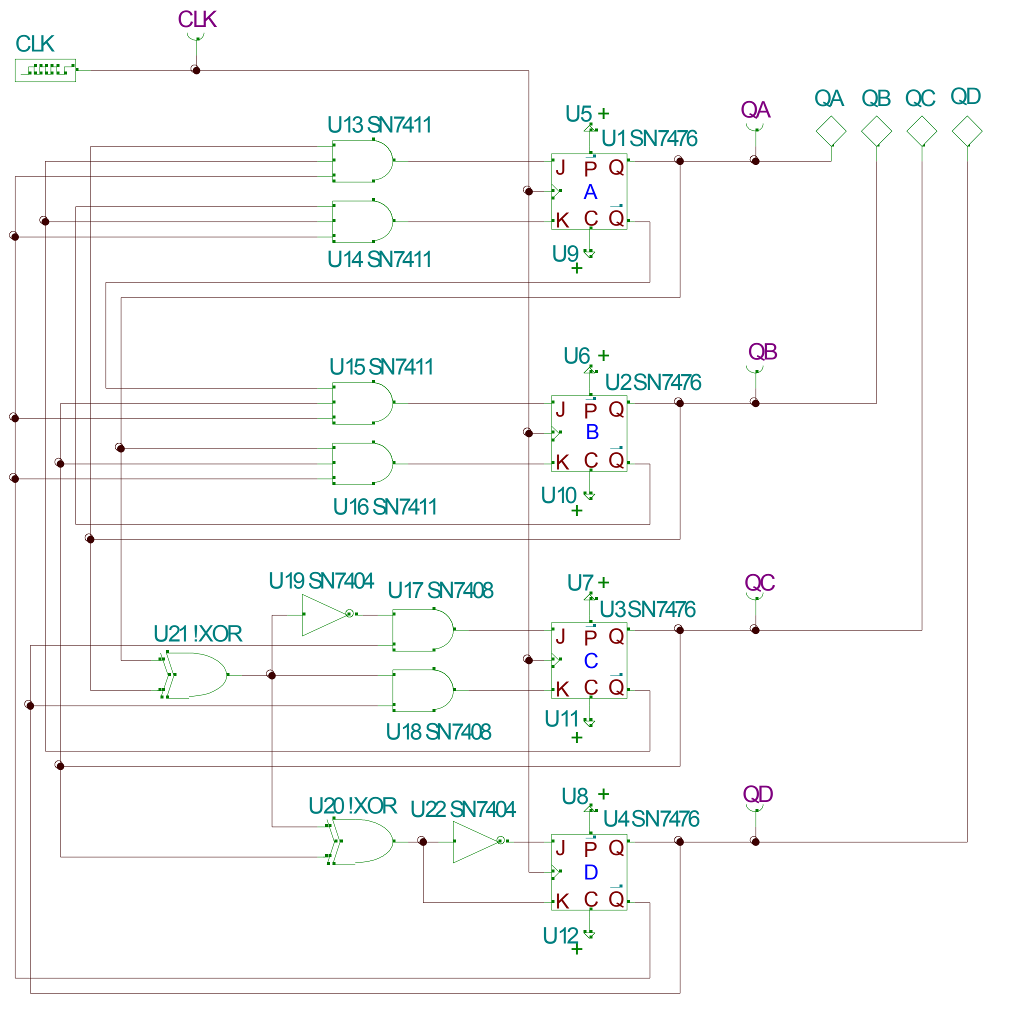


Fig. 13: Gray code generator simulation schematic

4.5. Simulation result

The figure 14 below shows the digital timing analysis for the Gray code generator derived from TINA. When looking at the timing for each clock pulse, the output and then compare it with the state table, it is possible to see that the output of the generator matches the specifications.

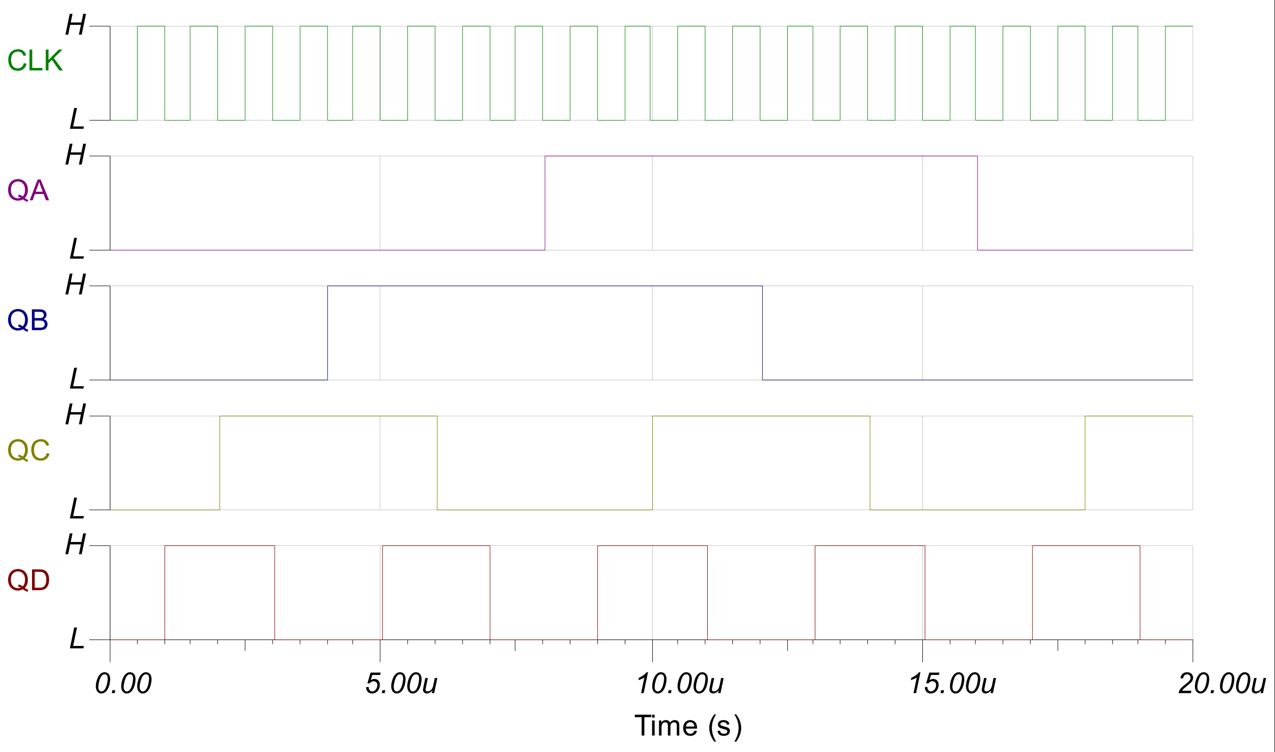


Fig. 14: Gray code generator digital timing analysis

5. Gray code to BCD decoder

This type of code converter are used to convert the Gray code to a useful and known code like the binary code. The binary output could then be set to a display where the counting sequence for the Gray code could be visual accomplished. The design procedure is the same as for the Gray code generator, except there will for the BCD decoder only be used logic gates, and no flip-flop's.

5.1. State transition diagram

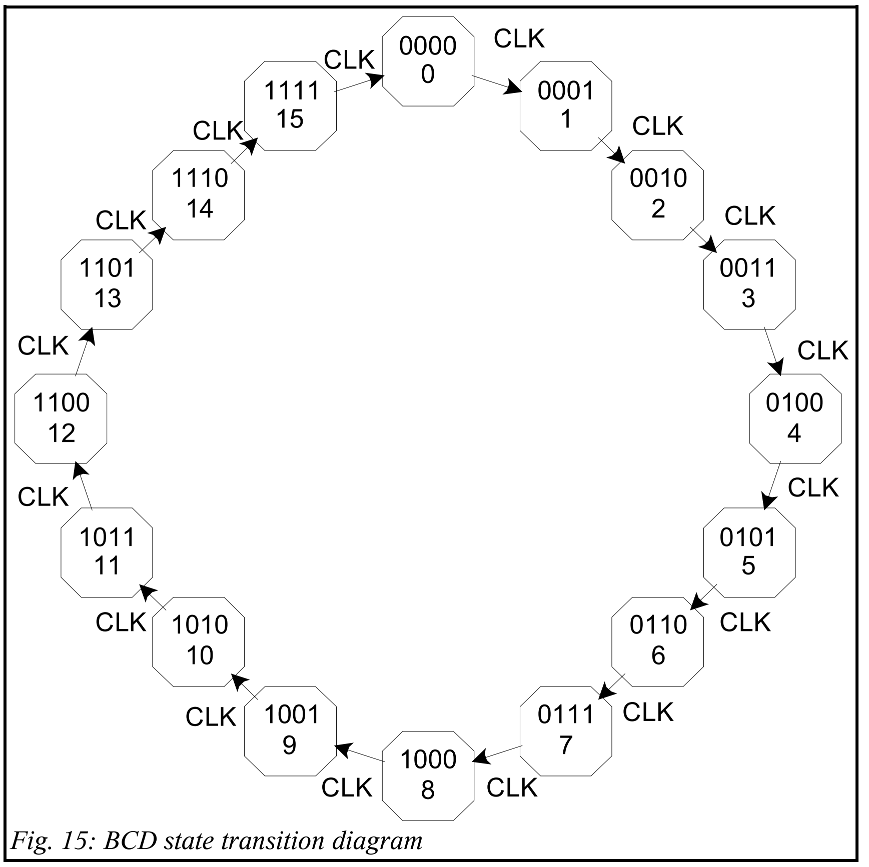


Fig. 15: BCD state transition diagram

The input to output conversion to be dealt with, consists of 4 bit for both the input and the output. This can be described best as a parallel input to a parallel output, which each of the clock-pulses in the figure 15 above indicates.

5.2. State table

With the known Gray code and binary sequence, the state table can made as shown in the table below.

Label	Gray code				Binary out			
	A	B	C	D	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

Like with the Gray code generator, there are used Karnaugh mapping, to find the boolean expressions for each of the outputs (next state logic). From this the needed logical gates can then be found.

5.3. Next state logic

CD \ AB	0 0	0 1	1 1	1 0
0 0	0	0	1	1
0 1	0	0	1	1
1 1	0	0	1	1
1 0	0	0	1	1

Q_A Output

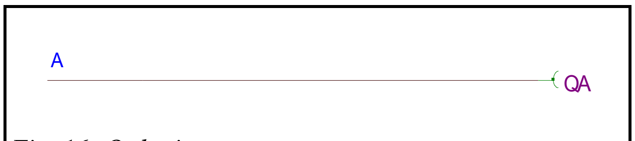


Fig. 16: Q_A logic output

$$Q_A = \underline{\underline{A}}$$

CD \ AB	0 0	0 1	1 1	1 0
0 0	0	1	0	1
0 1	0	1	0	1
1 1	0	1	0	1
1 0	0	1	0	1

Q_B Output

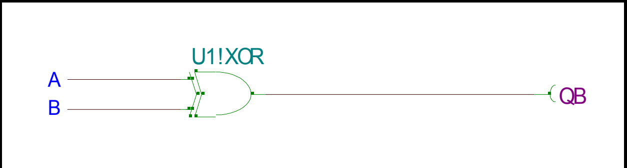
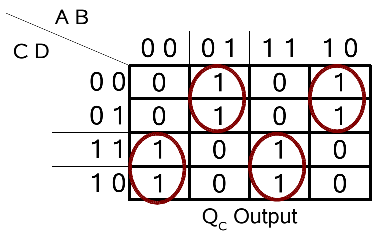


Fig. 17: Q_B logic output

$$Q_B = \overline{A} \cdot B + A \cdot \overline{B}$$
$$= \underline{\underline{A \oplus B}}$$



CD \ AB	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	0	1	0
10	1	0	1	0

Q_C Output

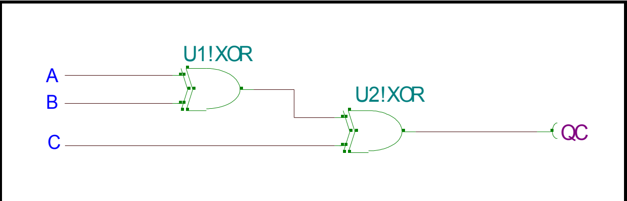
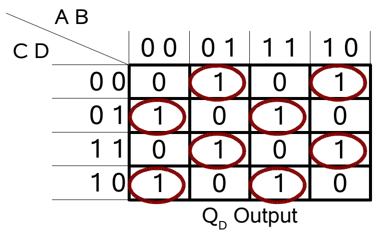


Fig. 18: Q_C logic output

$$
\begin{aligned}
Q_C &= \overline{A}\cdot\overline{B}\cdot C+\overline{A}\cdot B\cdot\overline{C}+A\cdot B\cdot C+A\cdot\overline{B}\cdot\overline{C}\\
&= C\cdot(\overline{A}\cdot\overline{B}+A\cdot B)+\overline{C}\cdot(\overline{A}\cdot B+A\cdot\overline{B})\\
&= C\cdot(\overline{A\oplus B})+\overline{C}\cdot(A\oplus B)\\
&= \underline{\underline{A\oplus B\oplus C}}
\end{aligned}
$$



CD \ AB	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

Q_D Output

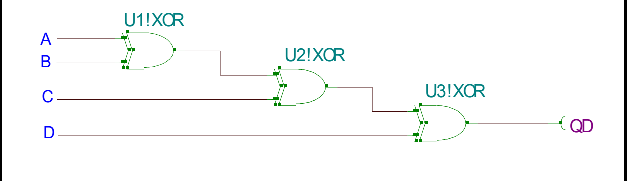


Fig. 19: Q_D logic output

$$
\begin{aligned}
Q_D &= \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D+\overline{A}\cdot\overline{B}\cdot C\cdot\overline{D}+\overline{A}\cdot B\cdot\overline{C}\cdot\overline{D}+\overline{A}\cdot B\cdot C\cdot D+A\cdot B\cdot\overline{C}\cdot D+A\cdot B\cdot C\cdot\overline{D}+A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}+A\cdot\overline{B}\cdot C\cdot D\\
&= \overline{A}\cdot\overline{B}\cdot(\overline{C}\cdot D+C\cdot\overline{D})+\overline{A}\cdot B\cdot(\overline{C}\cdot\overline{D}+C\cdot D)+A\cdot B\cdot(\overline{C}\cdot D+C\cdot\overline{D})+A\cdot\overline{B}\cdot(\overline{C}\cdot\overline{D}+C\cdot D)\\
&= \overline{A}\cdot\overline{B}\cdot(C\oplus D)+\overline{A}\cdot B\cdot(\overline{C\oplus D})+A\cdot B\cdot(C\oplus D)+A\cdot\overline{B}\cdot(\overline{C\oplus D})\\
&= \overline{A\oplus B}\cdot(C\oplus D)+A\oplus B\cdot(\overline{C\oplus D})\\
&= \underline{\underline{A\oplus B\oplus C\oplus D}}
\end{aligned}
$$

5.4. Schematic

By looking at each of the next state logic outputs of the Gray code to BCD decoder, there is one XOR gate added for each output. This means that the final schematic can be reduced quite a lot! When comparing the schematic in figure 20 with all the reduced logic outputs above, it is possible to derive, that the circuits can be reduced from six to three gates. This circuit has not been simulated individually but instead in the final and complete schematic.

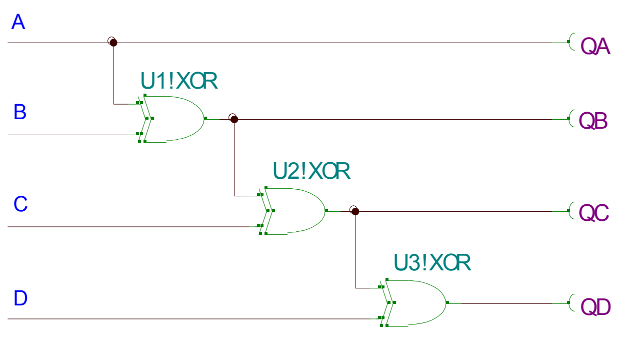


Fig. 20: Gray code to BCD schematic

6. Complete schematic

The complete schematic drawing from TINA are shown in figure 21 below. The Gray code generator are connected to the decoder, from where the output for the following timing analysis can be derived. Further more an hex display are connected to the output of the decoder, to better visualize the counting sequence in TINA. The schematic are simulated for this, in the digital step-by-step mode, where the display has counted from 0 to 9 then A to F, and over again.

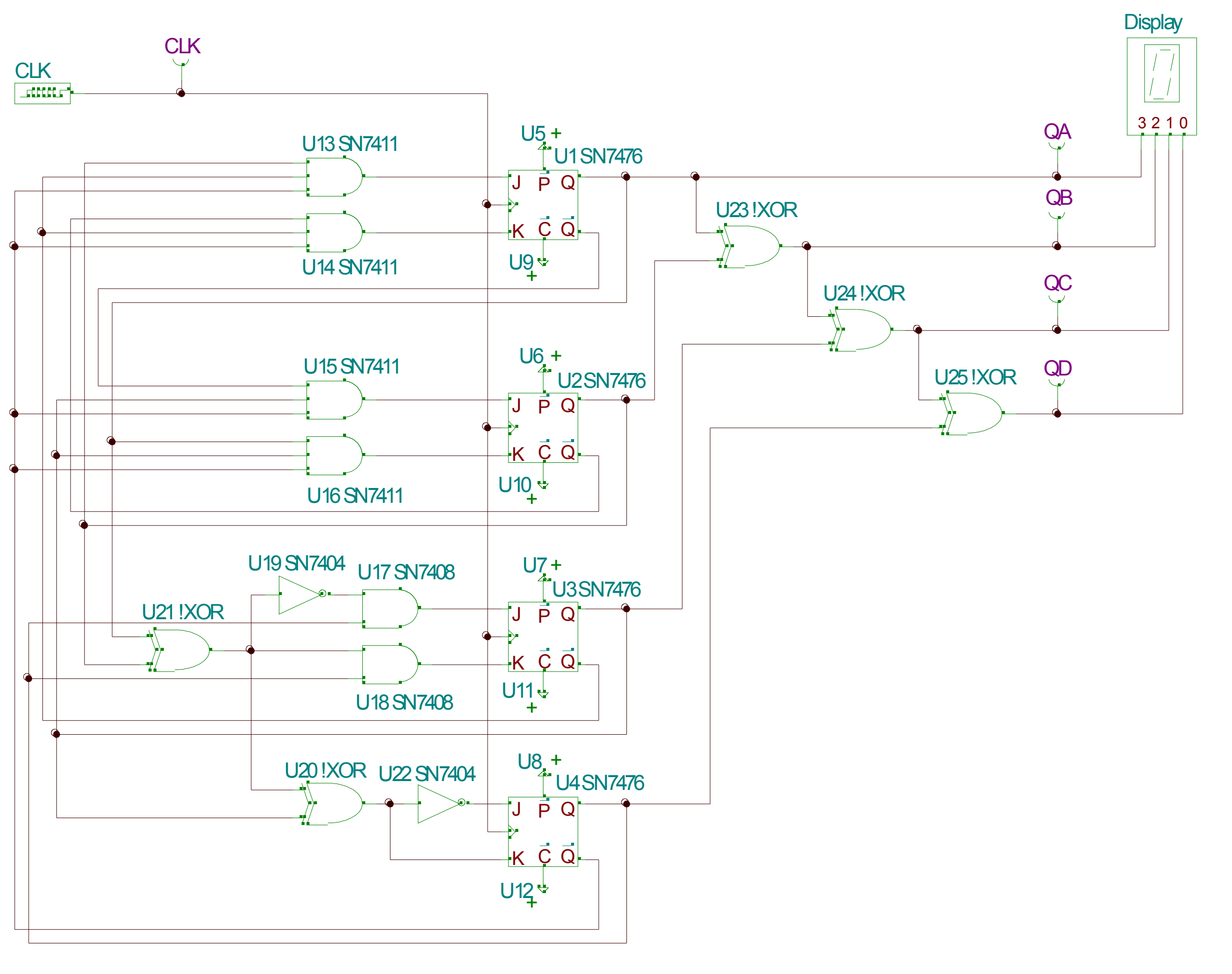


Fig. 21: Gray code generator and decoder – complete schematic

6.1. Simulation result

The figure 22 shows the final simulation with the complete schematic. The simulation has lasted over a time period of 20 µS, where after the period of 16 µS the final stage where all the outputs are logic high (1111). Each of the output sequences matches those set in the table of specifications, for the Gray code to binary output.

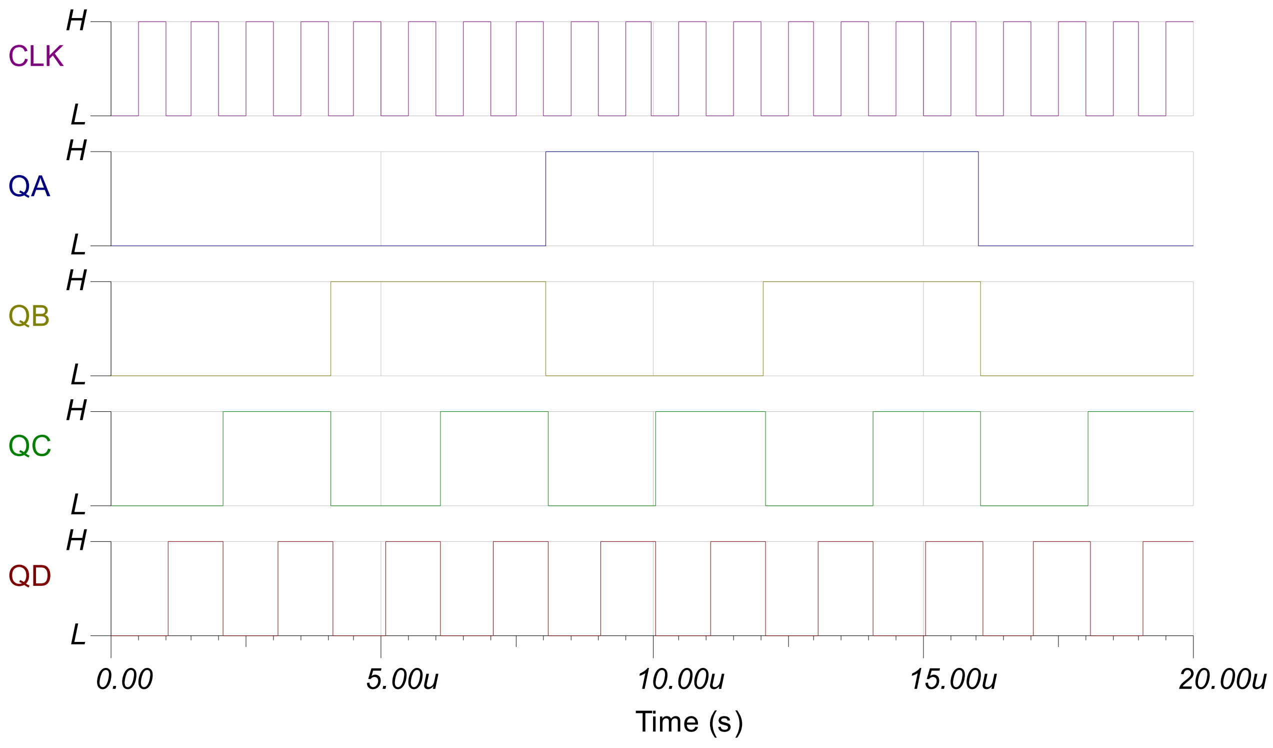


Fig. 22: Gray code generator and decoder digital timing analysis

7. Conclusion

The design and simulation assignment of a Gray code generator and decoder has been completed successfully. The assignment seemed difficult in the beginning, but once a little research about Gray code and its counting sequence was found, everything just went smoothly. Also the part where the logical gates where to be found by reducing the boolean expressions had to be refreshed, before the final design could be a successful possibility. There are though limits to TINA and designing the generator, where TINA don't have XNOR gates. This gate has been achieved by using an inverter before the final output of the logical gate, to make the direct opposite of a XOR.

When designing the generator and decoder on paper with the boolean expression and in TINA, it came in handy to give it a extra look to see that some of the gates could share their input to output conversion.

The assignment of designing the Gray code generator and decoder has been an interesting assignment, cause of the experiences it has given by the use of boolean algebra, logical gates and flip-flops. The use of this to make generators and decoders has given some second thoughts on how to build these and how the procedures would be to make a complete design. If I as the designer where to build the generator and decoder for practical use, it would be by using either a CPLD or a microprocessor. The CPLD would be the best device to use for this assignment, cause of the analysing, gate outputs and the use of flip-flops, which is very suitable to make a complete design of in a CPLD. The software for this like for instance Xilinx's has also got the neat feature that it is capable of simulating the schematics.

An important thing to do, when working with logical devices is also to think logical, and that has been a very stimulating experience in this assignment. A little knowledge to electronic is of course a must, but one can get far just by having a logical mind.

Carsten Kristiansen

8. References

8.1. Internet

- The meaning of Gray code, www.hyperdictionary.com/computing/gray+code
- Gray code – from Mathworld, http://mathworld.wolfram.com/GrayCode.html
- Gray code algorythms, http://yagni.com/graycode/#what
- Binary codes, www.shef.ac.uk/physics/teaching/phy107/codes.html

8.2. Literature

- Introduction to electrical engineering, Mulukutla S. Sarma.
- Handout notes for Sequential synchronous circuits, Jay Hoy.
- Digital Teknik, Leif Møller Andersen.

8.3. Software tools

- TINA Pro for Windows, www.designsoft.com