

# UNIVERSIDADE FEDERAL DO CARIRI CENTRO DE CIEˆNCIAS E TECNOLOGIA

**BACHARELADO EM CIEˆNCIA DA COMPUTAC¸ A˜O**

# Projeto 01 - Guess The Number Professor: Ramon Santos Nepomuceno

## Wanderson Faustino Patricio - MAT: 2022005052 Francisco Anderson Maciel Cruz - MAT: 2022005876

Juazeiro do Norte, 29 de agosto de 2023

# Apresenta¸c˜ao do projeto

Este projeto introduz o conceito de um jogo de adivinhação de 4 bits. Duas pessoas participam da atividade, competindo para acertar um número aleatório de 4 bits. Cada jogador faz palpites alternados e recebe feedback sobre se o número que eles sugeriram é maior, menor ou igual ao número real. O objetivo principal é ser o primeiro a adivinhar corretamente o número em questão e, assim, conquistar a vitória no jogo.

Empregamos o software Logism para criar os circuitos lógicos que implementam as funções essenciais de forma que o layout inicial do jogo é apresentado da seguinte forma.

Diagrama, Esquemático

Descrição gerada automaticamente

Figura 0: Layout final do jogo.

1. **Circuito Decodificador**

## Motiva¸c˜ao para o uso do circuito decodificador

A entrada de dados para o circuito principal do game ´e feita atrav´es de 4 pinos de entrada, que podem ter seus valores alternados entre 0 e 1. Cada combina¸c˜ao distinta de valores nos pinos de entrada corresponde a um nu´mero bin´ario diferente, sendo ao total 16 combina¸c˜oes (0 ao 15).

Para tornar o game mais acess´ıvel e de f´acil visualiza¸c˜ao, torna-se necess´aria a utiliza¸c˜ao de um display (no nosso caso um display de 7 segmentos) para informar de maneira mais direta ao usu´ario qual nu´mero est´a sendo tomado como palpite na rodada. Com o display ´e poss´ıvel imprimir qualquer nu´mero de 0 a 9, al´em de algumas letras do alfabeto. Tendo isto em mente, a utiliza¸c˜ao do sistema de numera¸c˜ao hexadecimal torna-se o mais indicado para a interface com o usu´ario, pois ser´a necess´ario apenas um display para cada jogador.

Fazendo a correspondˆencia entre os sistemas de numera¸c˜ao podemos montar a seguinte tabela de convers˜ao.

|  |  |  |
| --- | --- | --- |
| Bin´ario | Decimal | Hexadecimal |
| 0000 | 0 | 0 |
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | 10 | A |
| 1011 | 11 | B |
| 1100 | 12 | C |
| 1101 | 13 | D |
| 1110 | 14 | E |
| 1111 | 15 | F |

Tabela de convers˜ao Bin-Dec-Hex

## Montagem da tabela verdade do decodificador

Para formar algum nu´mero no display ´e necess´ario permitir a passagem de energia em entradas espec´ıficas, enquanto bloqueia a passagem nas outras. Nomeando cada pino do display por uma letra temos a seguinte disposi¸c˜ao:

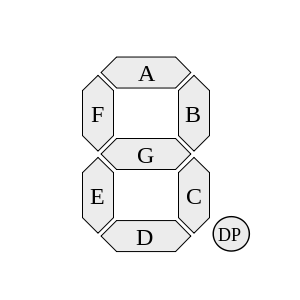


Figura 1: Pinos do display de 7 segmentos

Formando cada nu´mero temos:

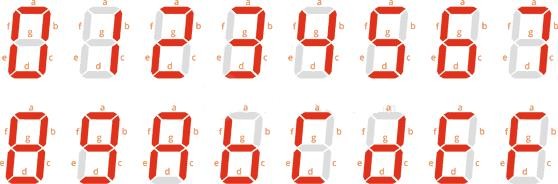


Figura 2: Nu´meros no display de 7 segmentos

A partir da forma¸c˜ao dos nu´meros podemos montar a tabela verdade para todas as combina¸c˜oes de bits.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *I*3 | *I*2 | *I*1 | *I*0 |  | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 |  | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 |  | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |  | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |  | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |  | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |  | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 |  | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |  | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |  | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |  | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 |  | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 |  | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |  | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |  | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

Tabela verdade do display

Com a tabela verdade ´e poss´ıvel montar o circuito combinacional de cada pino do display. Para exemplificar, mostraremos a express˜ao l´ogica do pino a.

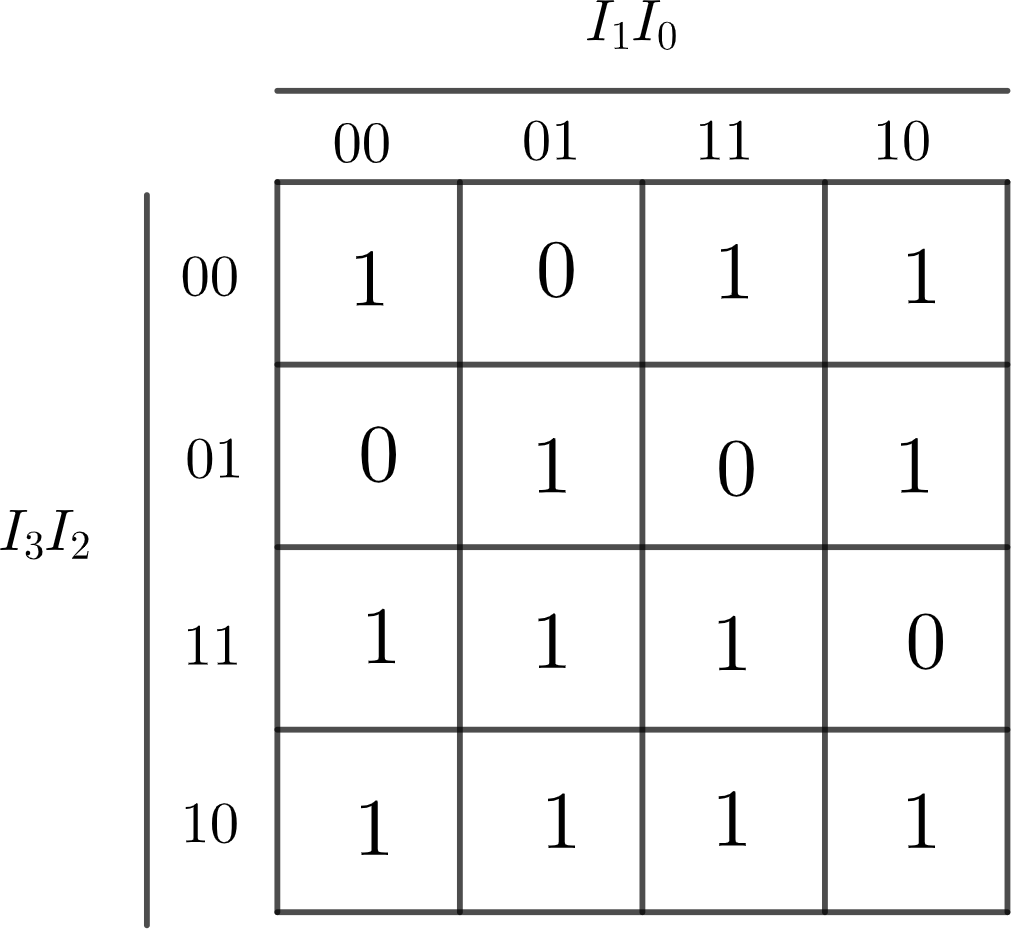


Figura 3: Mapa de Karnaugh para o pino a

Atrav´es do mapa de Karnaugh constru´ımos a express˜ao

*a* = *I*¯3*I*1 + *I*2*I*1 + *I*3*I*¯0 + *I*¯2*I*¯0 + *I*¯3*I*2*I*0 + *I*3*I*¯2*I*¯1

An´alogo ao procedimento descrito acima faremos `as demais entradas.

*a* = *I*¯3*I*1 + *I*2*I*1 + *I*3*I*¯0 + *I*¯2*I*¯0 + *I*¯3*I*2*I*0 + *I*3*I*¯2*I*¯1

*b* = *I*¯3*I*¯2 + *I*¯3*I*¯1*I*¯0 + *I*¯3*I*1*I*0 + *I*3*I*¯1*I*0 + *I*3*I*¯2*I*¯1 + *I*¯2*I*1*I*¯0

*c* = *I*¯1*I*0 + *I*3 ⊕ *I*2 + *I*¯3(*I*1 ⊕ *I*0)

*d* = *I*3*I*¯1*I*¯0 + *I*2(*I*1 ⊕ *I*0) + *I*¯3*I*¯2(*I*1 + *I*¯0) + *I*3*I*¯2*I*0

*e* = *I*1*I*¯0 + *I*3*I*2 + *I*¯2*I*¯1*I*¯0 + *I*3*I*1*I*0

*f* = *I*¯1*I*¯0 + *I*3*I*¯2 + *I*2(*I*3 ⊕ *I*1) + *I*2*I*1*I*¯0

*g* = *I*3*I*¯2 + *I*1*I*¯0 + *I*3*I*0 + *I*¯2*I*1 + *I*¯3*I*2*I*¯1

Ap´os montar o circuito a partir das express˜oes teremos o nosso decodificador.

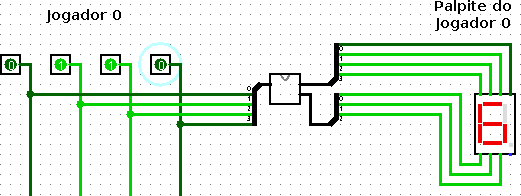


Figura 4: Decodificador em funcionamento

**3 Circuito Multiplexador**

**3.1 Motivação para o uso do circuito MULTIPLEXADOR.**  
 O multiplexador desempenha o papel de encaminhar o chute efetuado pelo jogador atual entre Jogador 0 e Jogador 1. Para realizar essa tarefa, empregamos uma estrutura composta por portas lógicas AND e OR.

**Diagrama

Descrição gerada automaticamente** Figura 5: Multiplexador .

**3.2 Agrupamento do MULTIPLEXADOR.**

A ideia é encapsular o multiplexador para que ele consiga atender a necessidade da entrada do jogo.

**Diagrama

Descrição gerada automaticamente**

Figura 6: Multiplexador 4bits.

**4 . Circuito Comparador**  
 Aceita dois números binários de 4 bits como entrada e produz três resultados distintos: um para indicar se o valor é superior, outro para quando é inferior e um terceiro quando coincide com um "número mágico" estabelecido dentro do comparador. Introduzimos uma constante que representa esse valor especial. Nosso dispositivo comparador é desenvolvido empregando uma combinação de portas lógicas XNOR, AND, NOT, OR e NOR.

Tabela

Descrição gerada automaticamente

Figura 7: Comparador.

**4.1 Comparador igual** Para representar a igualdade usamos **XNOR** e comparamos os dois números do bit mais significativo para o menos significativo.

**Diagrama, Esquemático

Descrição gerada automaticamente**

Figura 8: Comparador de igual.

**4.2 Comparador maior** Para comparar se o chute tem magnitude maior que o valor mágico, comparamos os dois números olhando do bit mais significativo para o menos significativo. **Diagrama, Esquemático

Descrição gerada automaticamente** Figura 9: Comparador maior.

**4.3 Comparador menor** utilizando das estruturas **4.1** e **4.2** podemos construir o comparador menor visto que um número sempre será maior ou igual, caso contrário ele será menor, dessa forma temos: **Diagrama

Descrição gerada automaticamente**

Figura 10: Comparador.