**Nome: Artur Vítor**

**1-**

Mintermos:

X = A'B'C'D + A'B'CD' + A'BC'D' + A'BCD + AB'C'D' + AB'CD + ABC'D + ABCD'

Y = A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD + A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D

Z = A'BC'D + A'BCD' + AB'C'D + AB'CD' + A'B'CD + A'B'CD' + A'BCD'

W = A'B'C'D' + A'B'CD + A'BC'D + A'BCD + AB'C'D + AB'CD' + AB'CD + A'B'CD' + A'BCD'

Maxtermos:

X = (A+B+C+D)(A+B+C'+D')(A+B'+C+D')(A+B'+C'+D')(A'+B+C+D')(A'+B+C'+D')(A'+B'+C+D')(A'+B'+C'+D')

Y = (A+B+C+D')(A+B+C'+D')(A+B'+C+D')(A'+B+C+D')(A'+B+C'+D')(A'+B'+C+D')

Z = (A+B+C+D)(A+B+C+D')(A+B+C'+D)(A+B+C'+D')(A+B'+C+D)(A+B'+C+D')(A'+B+C+D)(A'+B+C+D')(A'+B'+C+D)

W = (A+B+C'+D')(A+B+C'+D)(A+B'+C+D')(A+B'+C'+D)(A'+B+C'+D')(A'+B'+C+D')(A'+B'+C'+D')

**2-** Mapa K para simplificar X:  
  
 CD

00 01 11 10

AB

00 0 1 0 1

01 1 0 1 0

11 0 1 0 1

10 1 0 1 0

X=A′B′C′D+A′B′CD′+A′BC′D′+A′BCD+AB′C′D′+AB′CD+ABC′D+ABCD’

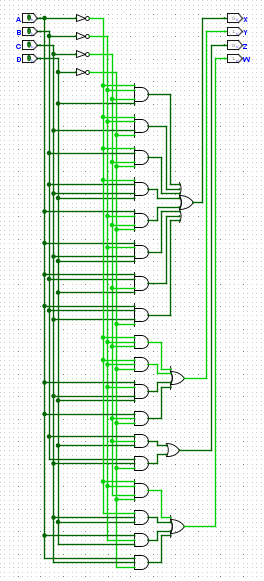
O mesmo processo foi repetido para Y, Z e W...

Y=A′B′C′D′+A′B′C′D+A′B′CD′+AB′C′D′+AB′CD+AB′C′D

Z=A′BC′D+A′BCD′+ABC′D+ABCD

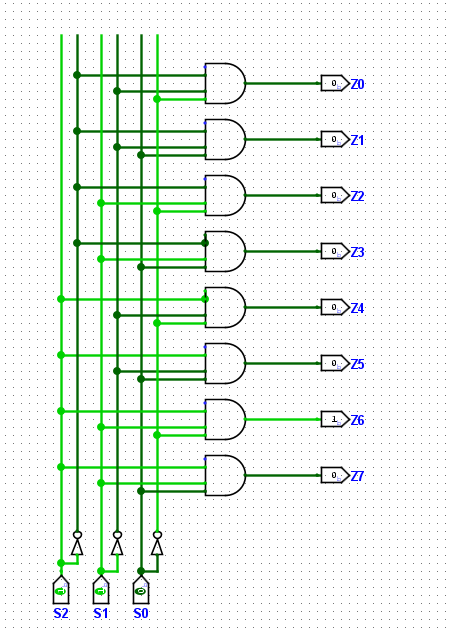
W=A′B′C′D′+A′B′CD+A′BCD+AB′C′D+AB′CD′+AB′CD+ABC′D′

**3- a)**

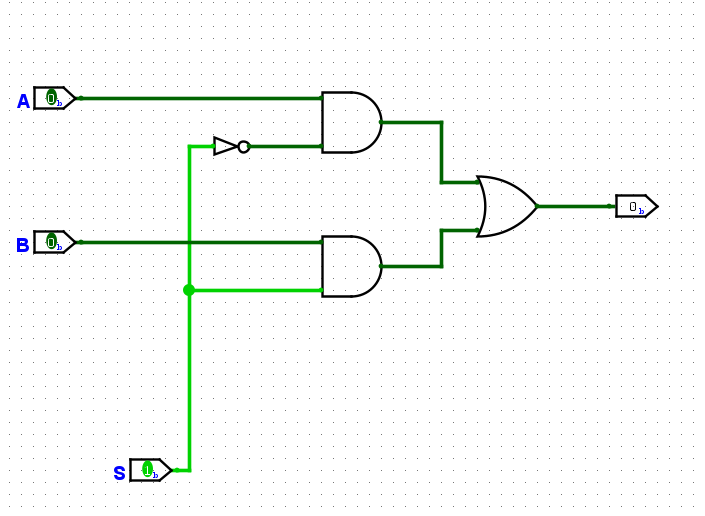
****

**4-** Para construir qualquer função lógica de n variáveis com um decodificador n para 2 elevado a n e uma porta OU, a função deve estar na forma canônica Soma de Produtos (SoP). Isso ocorre porque o decodificador gera todas as combinações possíveis das variáveis, e cada combinação ativa uma saída. A forma SoP permite que cada minitermo seja ligado diretamente às saídas do decodificador, e a porta OU soma esses minitermos para formar a função lógica.

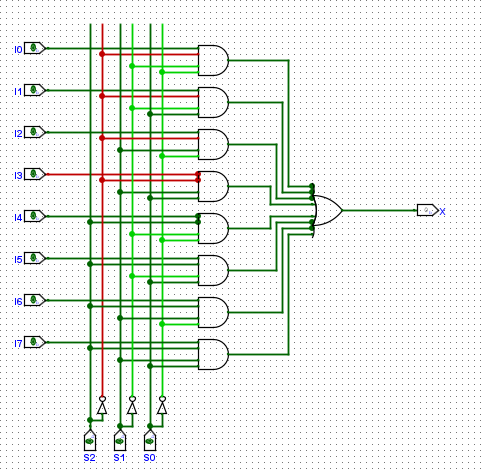
**5-**



**7-**

****

**8-**



**10-**

| **Estado Atual** | **Entrada X** | **Próximo Estado** |
| --- | --- | --- |
| S0 | 0 | S1 |
| S0 | 1 | S0 |
| S1 | 0 | S1 |
| S1 | 1 | S2 |
| S2 | 0 | S3 |
| S2 | 1 | S0 |
| S3 | 0 | S1 |
| S3 | 1 | S4 |
| S4 | 0 | S5 |
| S4 | 1 | S0 |
| S5 | \* | S0 |

| **Estado Atual (Q2 Q1 Q0)** | **Entrada X** | **Próximo Estado (Q2' Q1' Q0')** | **Saída (Z)** |
| --- | --- | --- | --- |
| 000 (S0) | 0 | 001 (S1) | 0 |
| 000 | 1 | 000 (S0) | 0 |
| 001 (S1) | 0 | 001 (S1) | 0 |
| 001 | 1 | 010 (S2) | 0 |
| 010 (S2) | 0 | 011 (S3) | 0 |
| 010 | 1 | 000 (S0) | 0 |
| 011 (S3) | 0 | 001 (S1) | 0 |
| 011 | 1 | 100 (S4) | 0 |
| 100 (S4) | 0 | 101 (S5) | 0 |
| 100 | 1 | 000 (S0) | 0 |
| 101 (S5) | \* | 000 (S0) | 1 |

Equações do circuito:

D2 = (Q1 · Q0 · X') + (Q2' · Q1' · Q0 · X)

D1 = (Q2' · Q1 · Q0') + (Q2' · Q1' · Q0 · X) + (Q2' · Q1 · Q0 · X')

D0 = (Q2' · Q1' · Q0' · X') + (Q2' · Q1 · Q0' · X) + (Q2 · Q1' · Q0')

Z = Q2 · Q0

**12- Overflow**: Ocorre em operações com **números com sinal** (representados geralmente em complemento de dois), quando o resultado ultrapassa o intervalo representável com o número de bits disponível.

Carry: Ocorre quando uma operação binaria gera um bit extra, que não cabe no espaço disponível para armazenar o resultado.