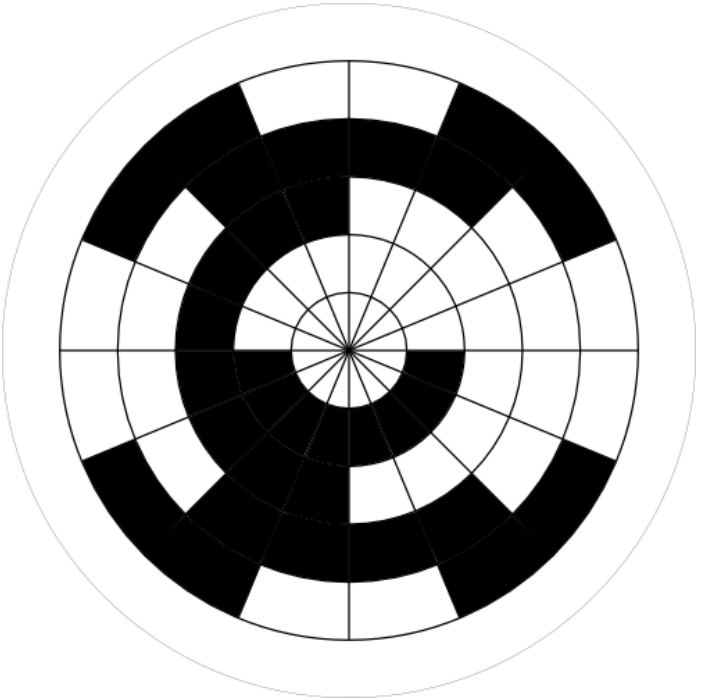
Design digitaler Schaltkreise
Vorlesung 8

Themen

- Grey Code
- Karnough-Tafeln
- Glitch
- (Kodierer)

Gray Code



0 1

0 1  1 0

0 0 1 1
0 1 1 0

0 0 1 1  1 1 0 0
0 1 1 0 0 1 1 0

0 0 0 0 1 1 1 1
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0

- Grey Code
- Grey Code hat die Eigenschaft, dass sich immer nur ein Bit ändert wenn man hochzählt

	B2	B1	B0	G2	G1	G0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

- Es gilt: (binary to Grey)
- G0 = B1 exor B0
- G1 = B2 exor B1
- …
- Gn-1 = Bn-1

	B2	B1	B0	G2	G1	G0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

- Es gilt auch: (Grey to binary)
- Bn-1 = Gn-1
- Bn-2 = Bn-1 exor Gn-2
- …
- B0 = B1 exor G0

	B2	B1	B0	G2	G1	G0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

Minimierung von Schaltfunktionen
Karnaugh Tabellen

- Karnaugh-Tabellen
- Kombinatorische Tabelle kann man als disjunktive Normalform darstellen
- Jeder Zeile mit 1 entspricht eine UND Funktion -> die Gesamttabelle ist ODER Funktion von einzelnen Zeilen
- Normalform kann oft vereinfacht werden
- Bsp. AB || !AB = B
- AB bedeutet hier A&B

Logische Funktion als Wahrheitstabelle

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Karnough Tabellen Grey Code

	B	0	0	1	1
D	CA	0	1	1	0
0	0				
0	1				
1	1			1	1
1	0			1	1

Karnough Tabelle - kompakter

DC	BA	00	01	11	10
00					
01					
11				1	1
10				1	1

Logische Funktion als Wahrheitstabelle

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	1	0	0	0
1	0	0	0	0

Karnough-Tabelle ist eine Graphische Darstellung der Wahrheitstabelle. Es ist aus einer Kanaugh-Tabelle leicht zu erkennen ob eine Normalform vereinfacht werden kann und wie.

Eine Karnaugh-Tabelle für n Eingangsvariablen hat 2^n Felder. (4 -> 16)

Am Rand der Tabelle werden die Variablen beschriftet – jede Zeile gehört einer Variable (negiert oder nicht-negiert) oder einem Produkt von zwei/drei (negierten oder nicht-negierten) Variablen – die negierte Variable wird mit Null beschriftet.

Wichtig ist, dass sich horizontal und vertikal benachbarte Felder nur in genau einer Variablen unterscheiden dürfen. *Gray Code* wird verwendet.

Mithilfe von Wahrheitstabelle wird in einzelnen Feldern Eins eigetragen wenn für die gegebene Variablen-Kombination die entsprechende Zeile eins ist.

Karnaugh-Diagramme eignen sich für die Vereinfachung von Funktionen mit maximal ca. 4–6 Eingangsvariablen; bis 4 Variablen sind sie übersichtlich. Ab dann Spiegelung

Karnough Tabellen Grey Code

	B	0	0	1	1
D	CA	0	1	1	0
0	0				
0	1				
1	1			1	1
1	0			1	1

Karnough Tabelle - kompakter

DC BA	00	01	11	10
00				
01				
11			1	1
10			1	1

In einem 2^k x 2^l Block nehmen k+l Variablen alle mögliche Kombinationen und die restlichen Variablen feste werte
Ein Block kann über den rechten bzw. unteren Rand des Diagramms fortgesetzt werden

	B	0	0	1	1
D	CA	0	1	1	0
0	0				
0	1		A,D,C fest		
1	1				
1	0				

	B	0	0	1	1
D	CA	0	1	1	0
0	0				
0	1				
1	1	A,D,C fest			
1	0				

	B	0	0	1	1
D	CA	0	1	1	0
0	0				
0	1				
1	1		A,D fest		
1	0				

	B	0	0	1	1
D	CA	0	1	1	0
0	0				
0	1				
1	1			B fest	
1	0				

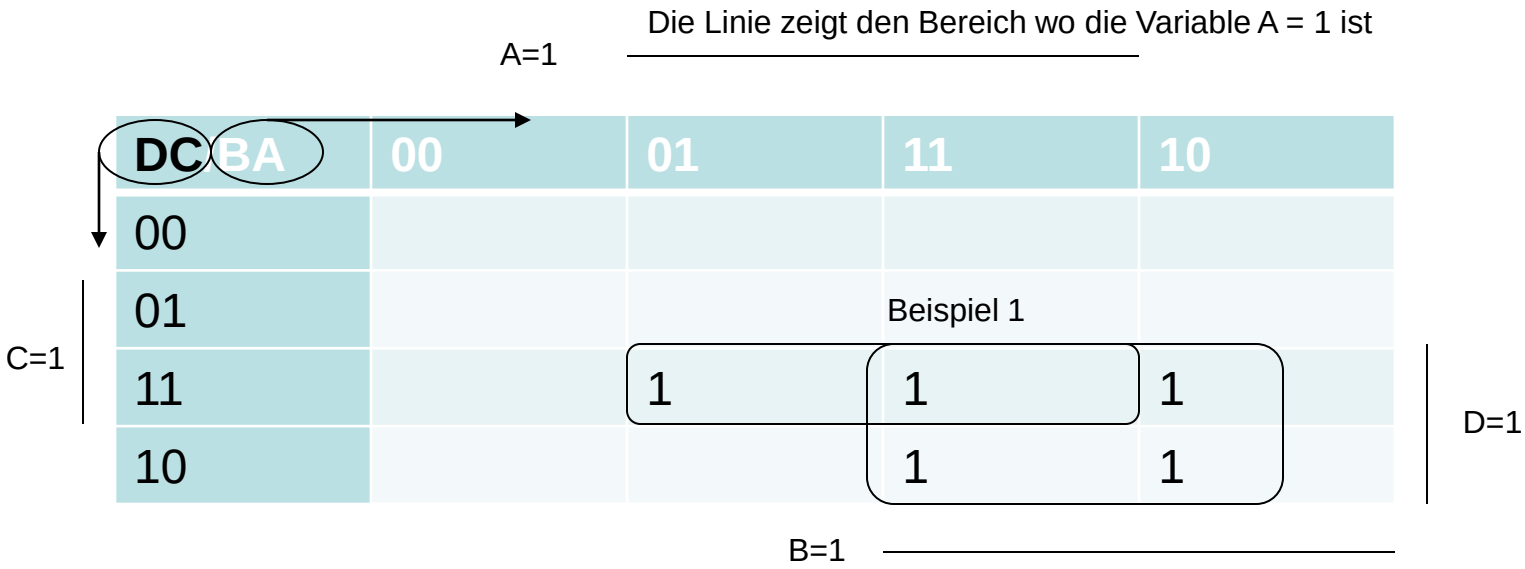
Wenn wir ein Block mit Einsen haben und wenn in diesem Block einige Variablen alle Kombinationen durchlaufen, können wir die entsprechende logische Funktion als UND von den festen Variable darstellen

	B	0	0	1	1
D	CA	0	1	1	0
0	0				
0	1				
1	1		1	1	
1	0		1	1	

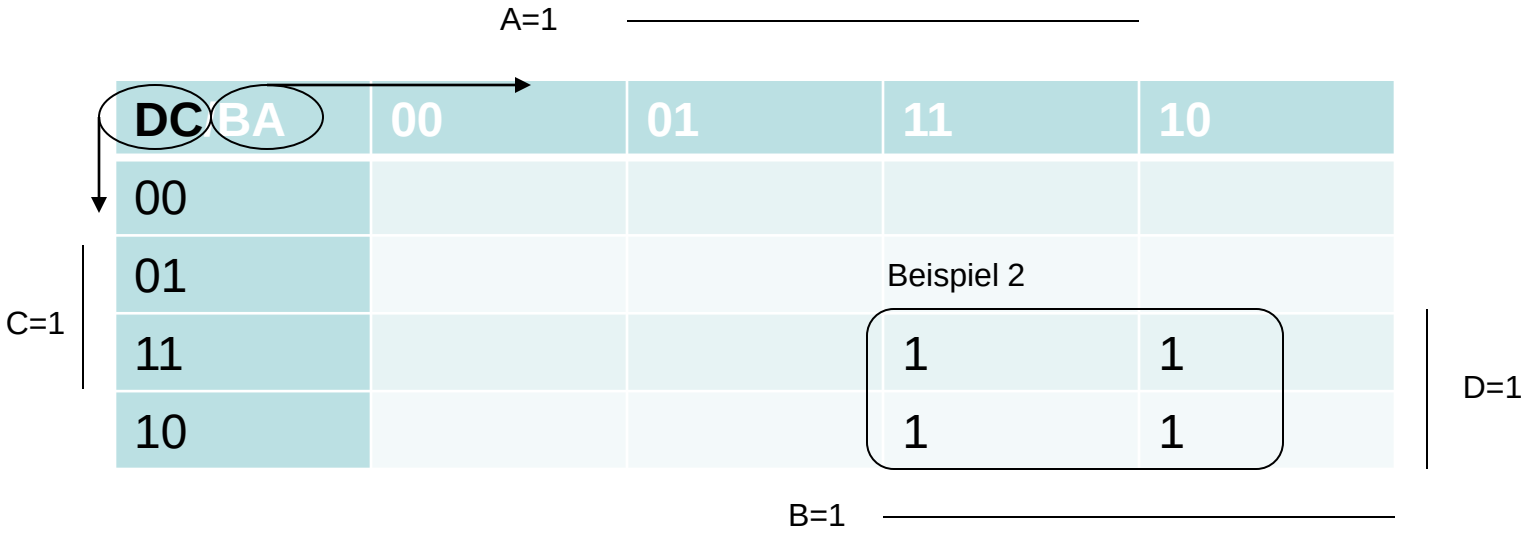
Y = A & D

A,D fest

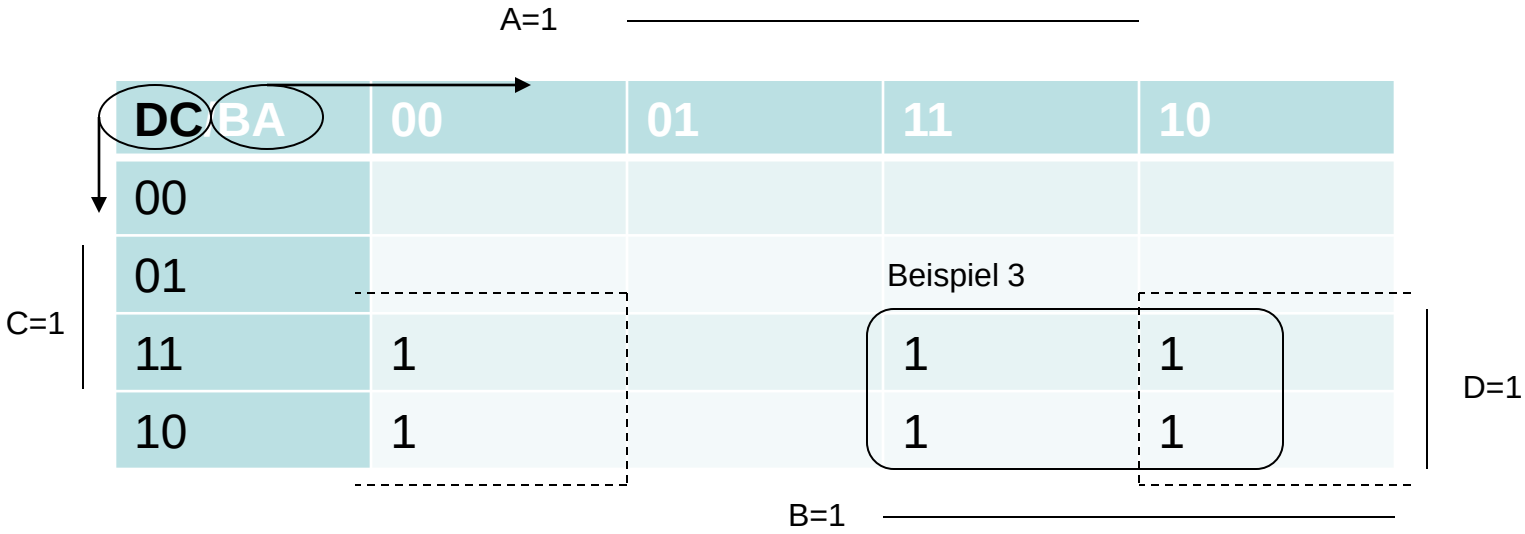
- Die Minimierung einer beliebiegen Funktion wird wie folgend gemacht
- Alle 1-Felder werden mit möglich größer Blöcken (2^k x 2^l) erfasst



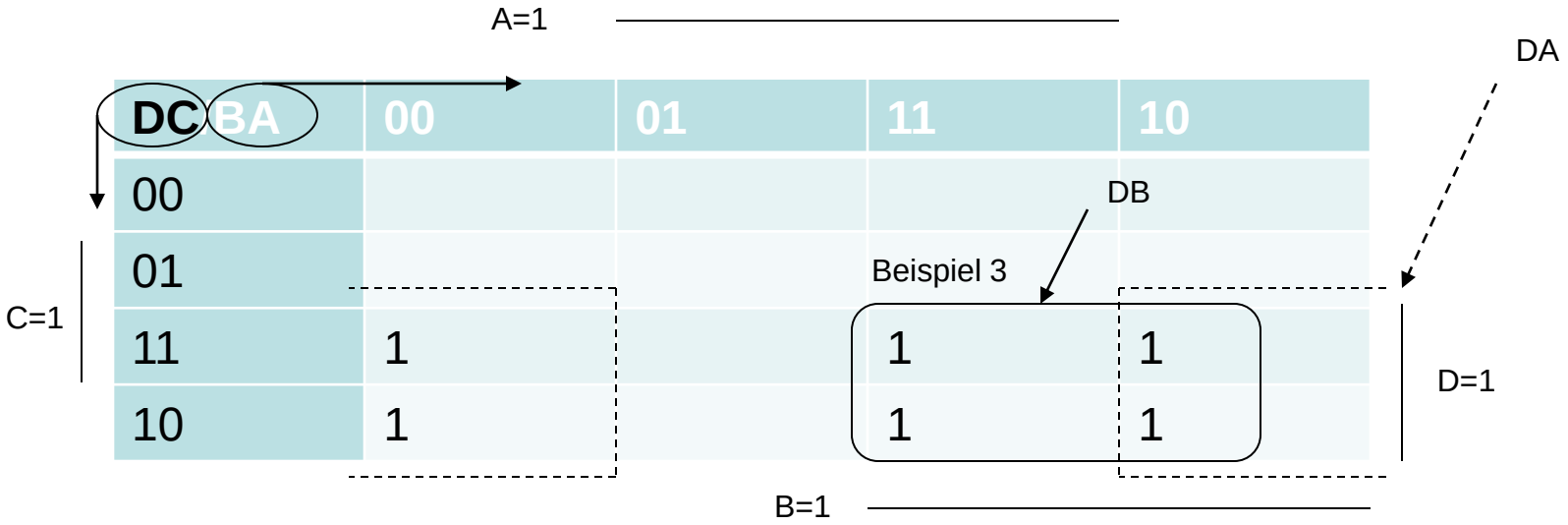
- Die Minimierung einer beliebiegen Funktion wird wie folgend gemacht
- Alle 1-Felder werden mit möglich größer Blöcken (2^k x 2^l) erfasst



- Die Minimierung einer beliebiegen Funktion wird wie folgend gemacht
- Alle 1-Felder werden mit möglich größer Blöcken (2^k x 2^l) erfasst



- Die gebildeten Blöcke bilden die Konjunktionsterme. Dabei werden Variablen innerhalb eines Blockes, die in allen Formenkombinationen auftreten, weggelassen.
- Diese UND-Verknüpfungen werden „ver-ODERt“ und ergeben eine disjunktive Minimalform.

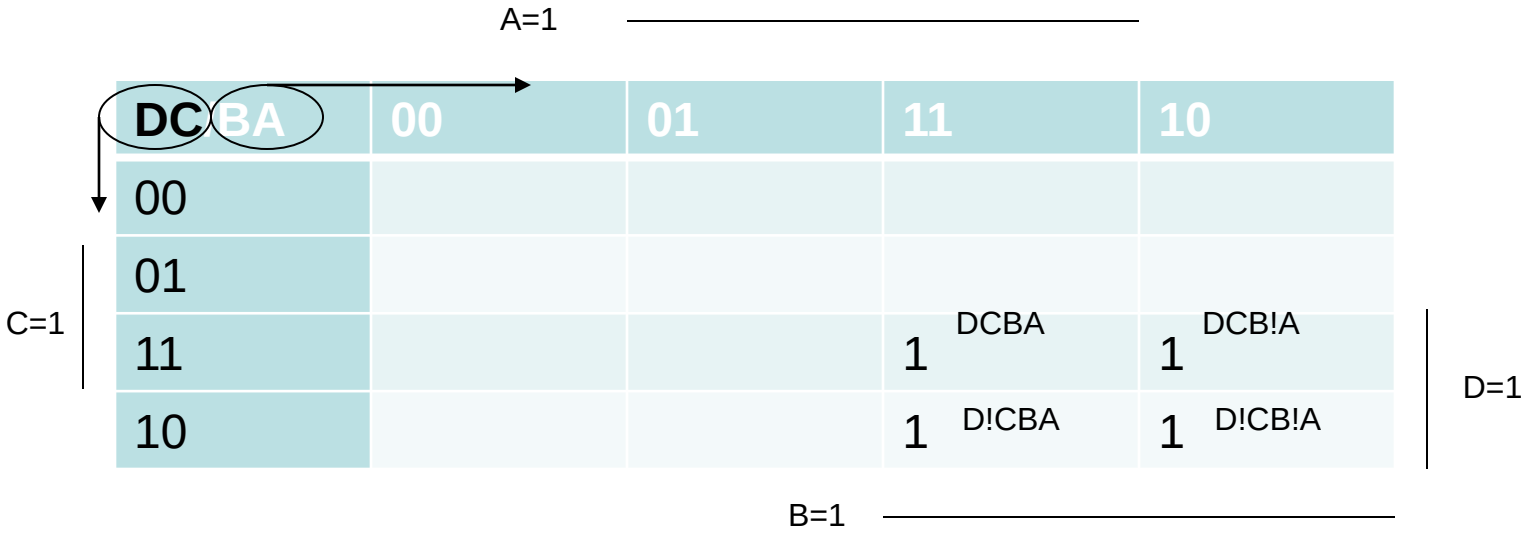


$$Y = DA + DB$$

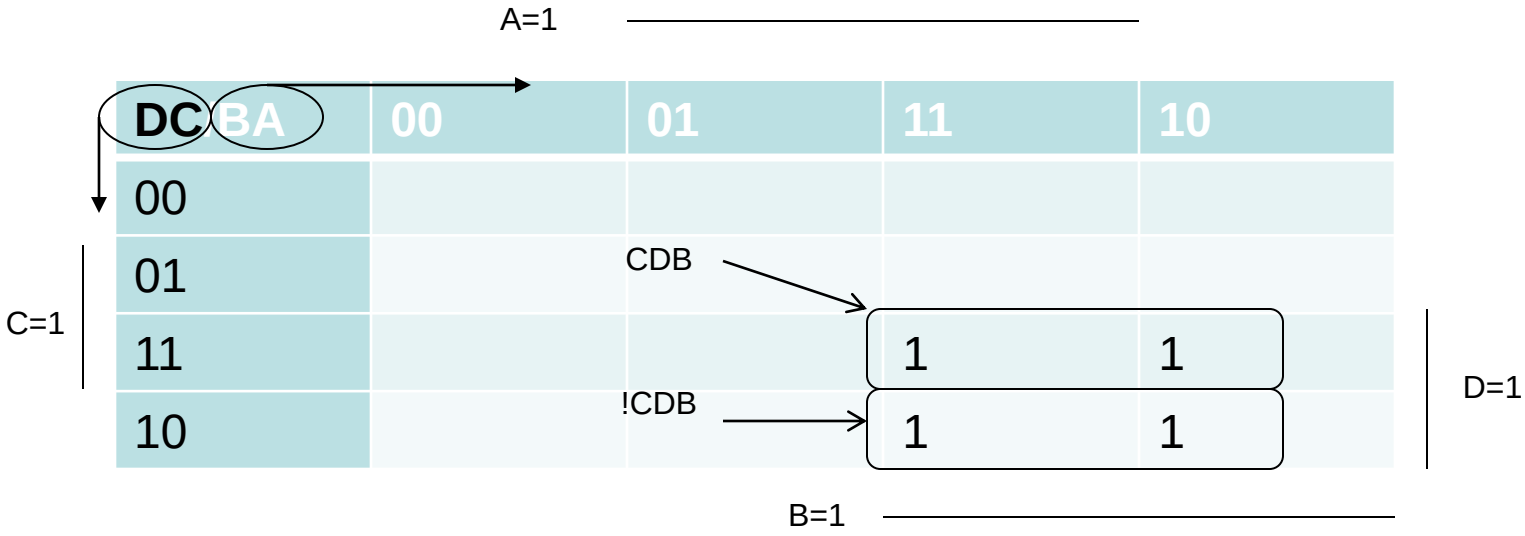
Beispiel

DNF (nicht minimal)

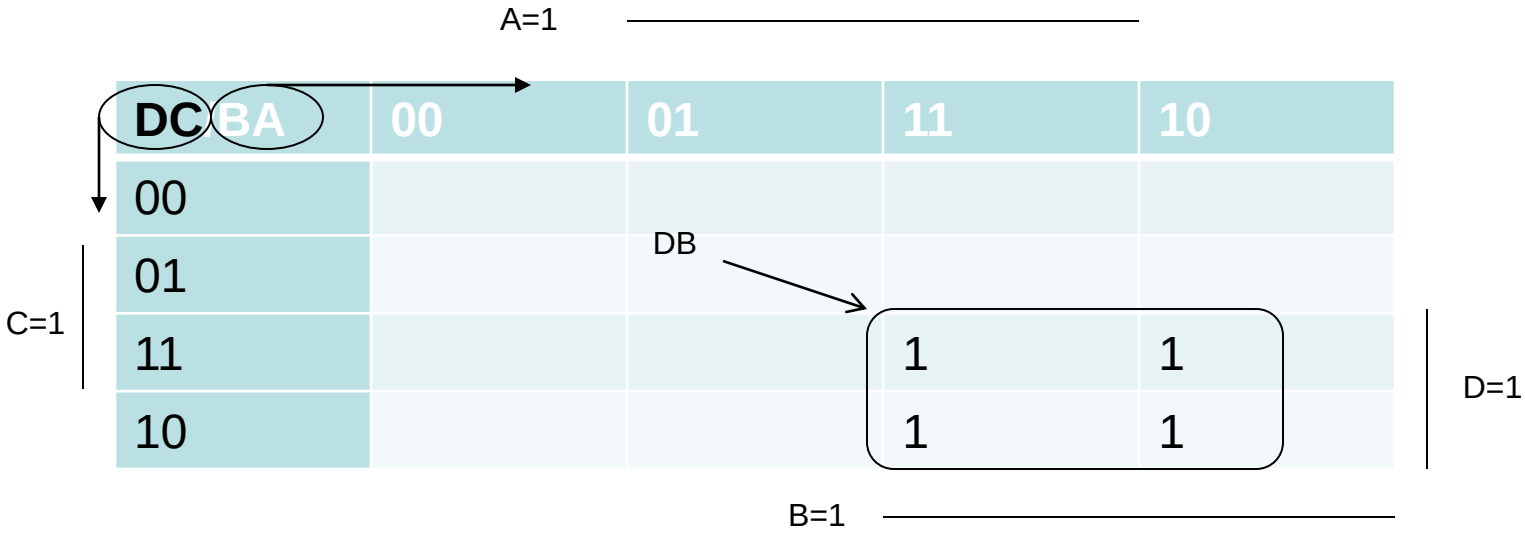
Y = DCBA + DCB!A + D!CBA + D!CB!A



Y=CDB + !CDB



Y=DB



Glitch

Glitch

Glitch steht für:

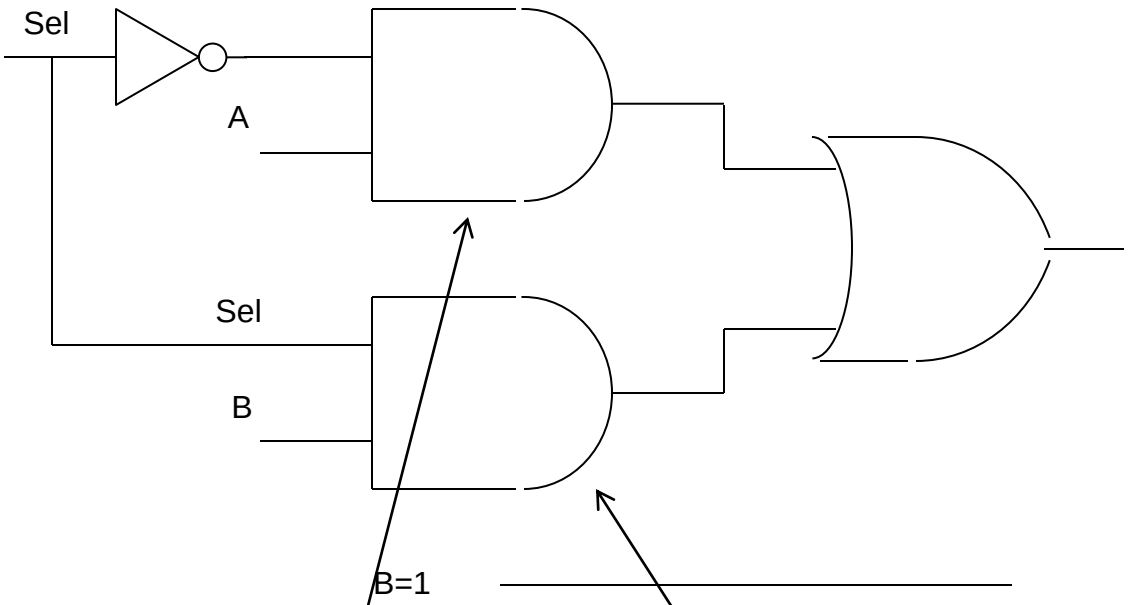
- Fehler in einem Computerspiel, siehe Gamersprache #Glitch
- Glitch (Elektronik), eine temporäre Falschaussage in logischen Schaltungen
- Glitch (Media), ein Filmfehler
- Glitch (Pulsar), Veränderung der Drehgeschwindigkeit bei Pulsaren
- Glitch (Fernsehserie), australische Fernsehserie
- eine Musikrichtung, siehe Clicks & Cuts #Glitch

Siehe auch:

- Glietsch
- Klietsch
- Klitsch
- Klitzsch

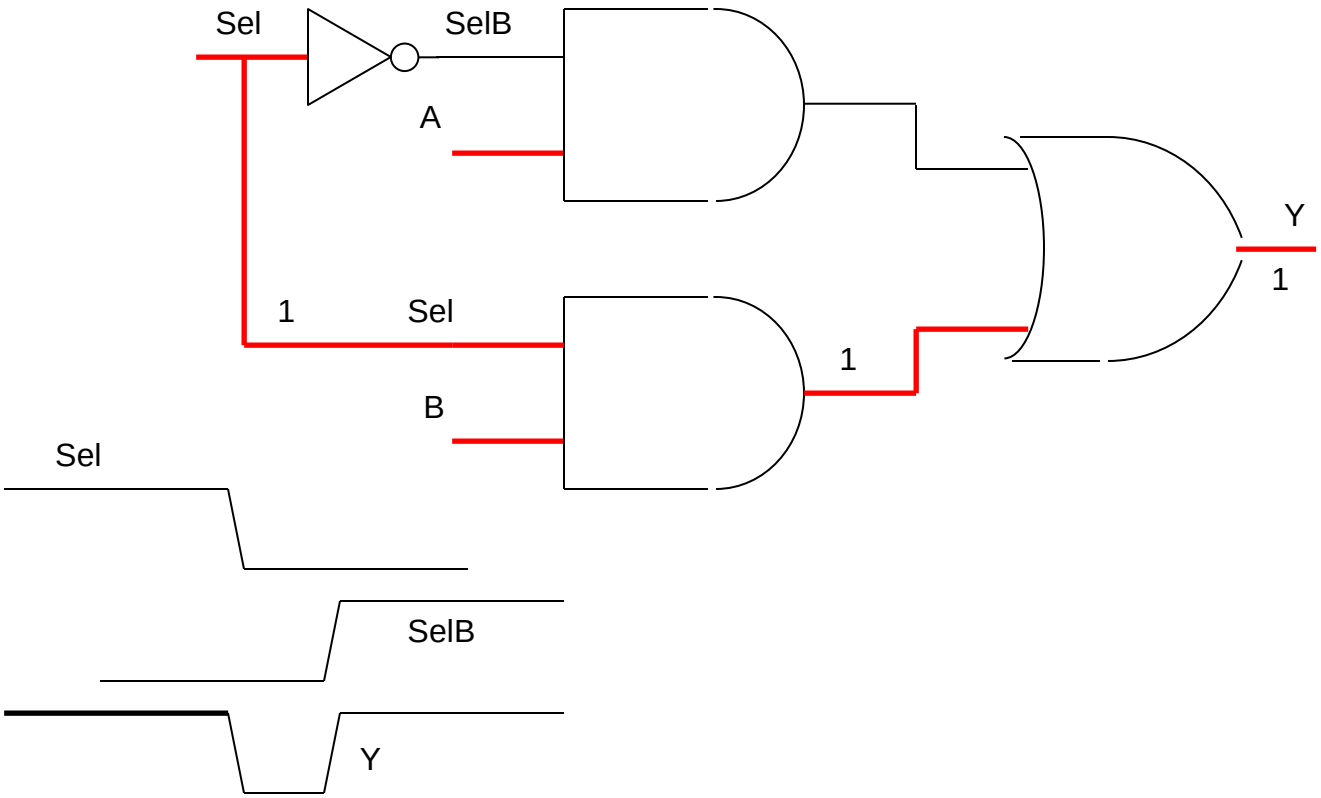
Wiktionary: glitch – Bedeutungserklärungen, Wortherkunft, Synonyme, Übersetzungen

- Die Verwendung minimaler Logik führt oft zu einem Problem genannt Glitch.

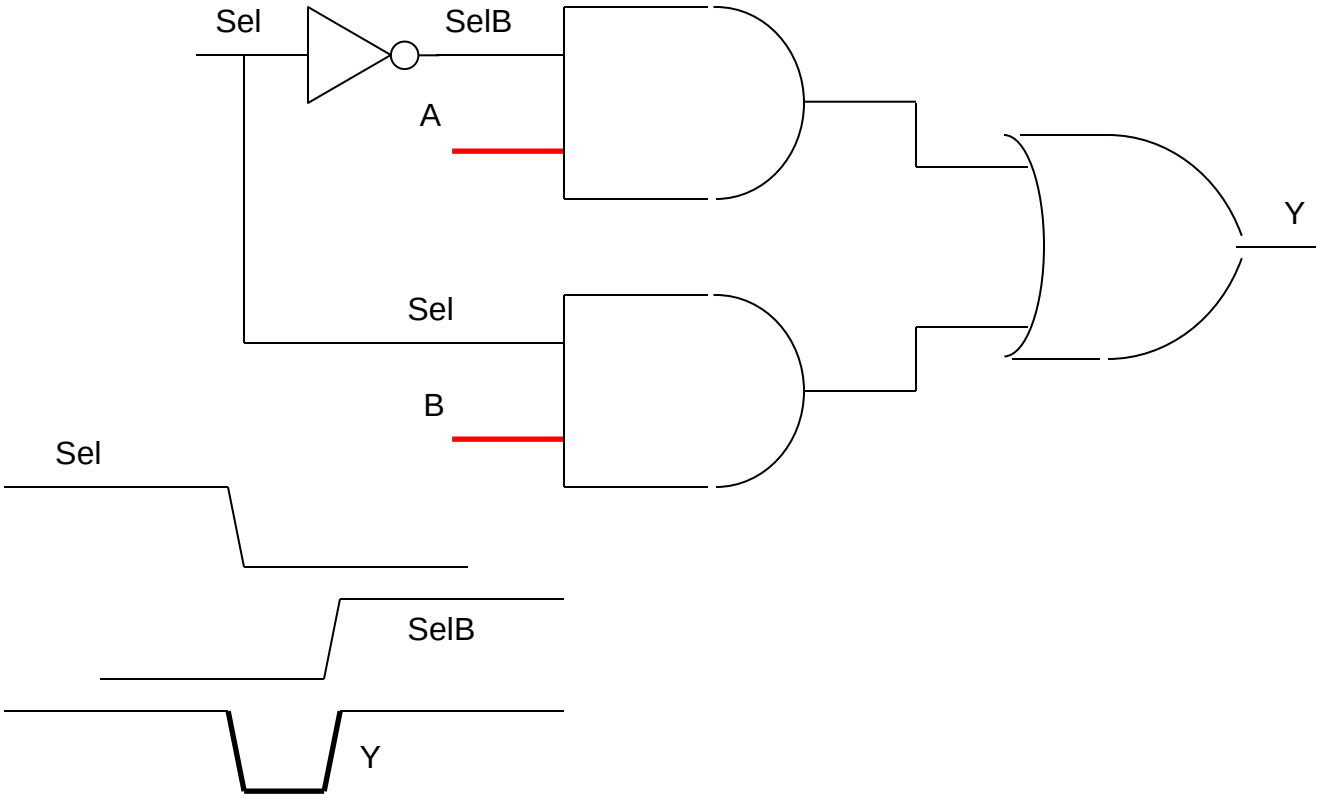


A SelB	00	01	11	10
0			1	
1	1	1	1	

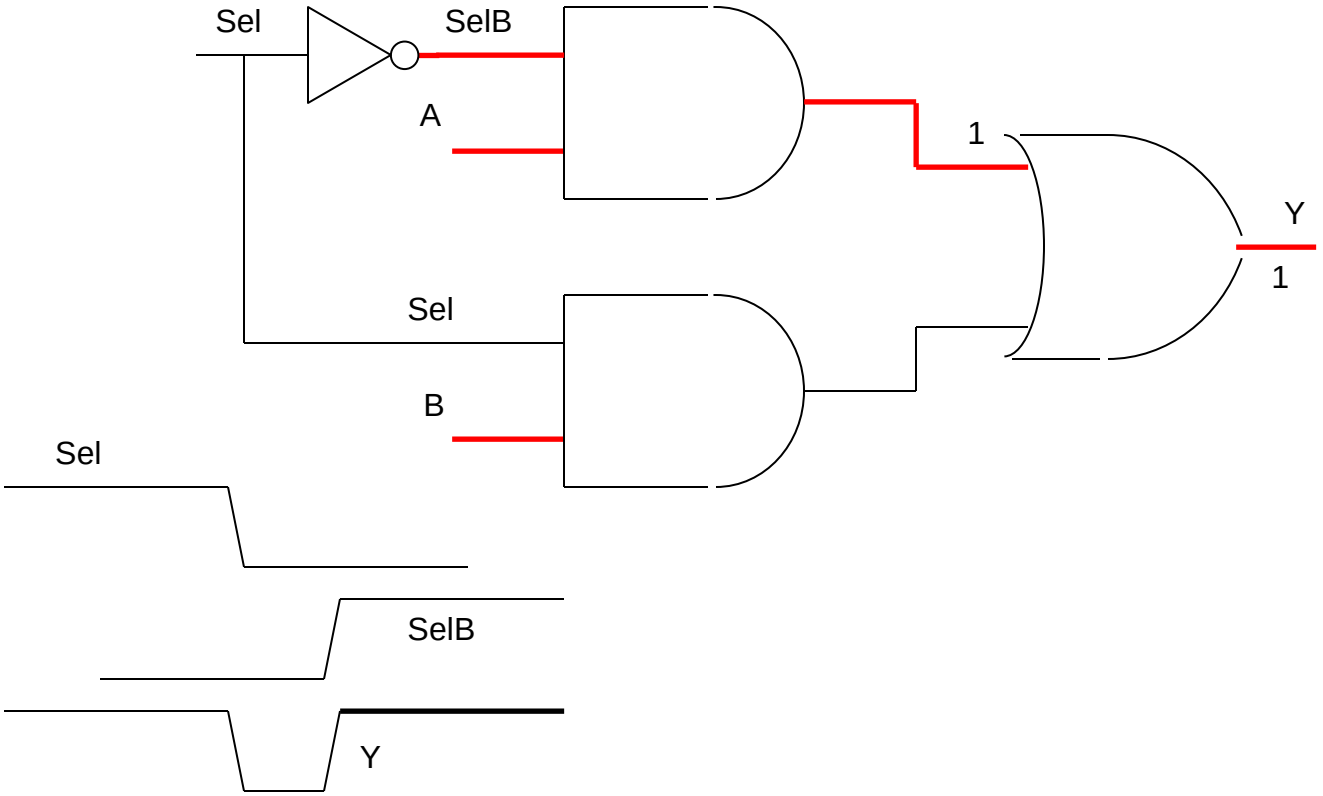
- Y = !Sel A + Sel B
- Nehmen wir an, dass beide Eingänge „1“ sind: A = B = 1
- Sel ist anfangs 1



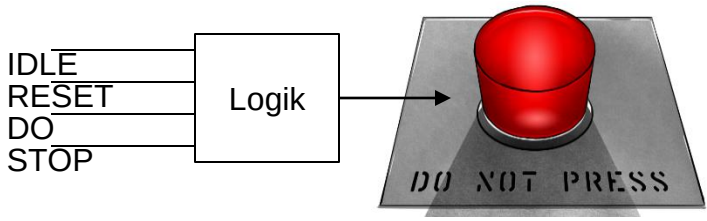
- Y = !Sel A + Sel B
- Nehmen wir an, dass beide Eingänge „1“ sind: A = B = 1
- Sel ist anfangs 1 und ändert sich auf 0 -> wir erwarten Y = 1
- Ein kurze Zeit sehen beide AND Gatter den Select Eingang 0, wir bekommen für eine kurze Zeit 0 am Ausgang

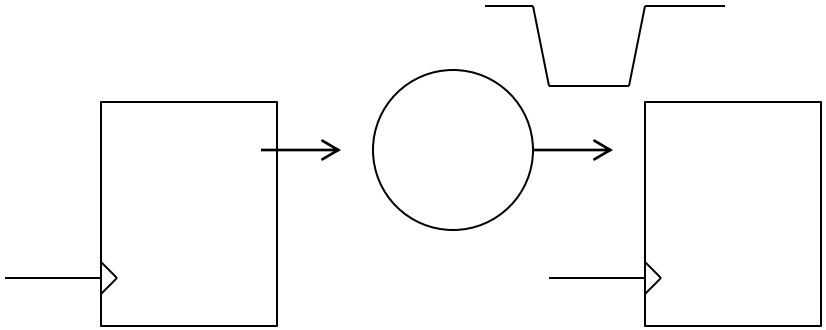


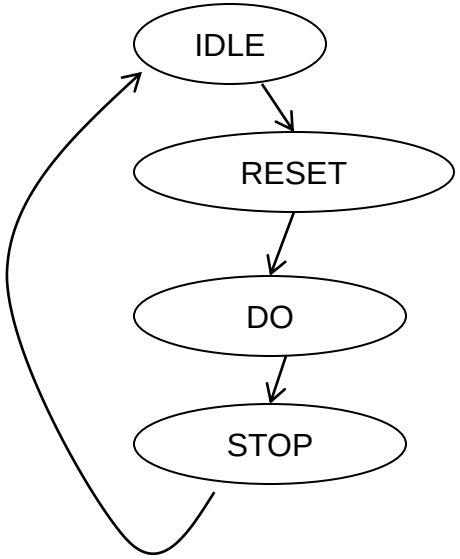
- Y = !Sel A + Sel B
- Nehmen wir an, dass beide Eingänge „1“ sind: A = B = 1
- Sel ist anfangs 1 und ändert sich auf 0 -> wir erwarten Y = 1
- Ein kurze Zeit sehen beide AND Gatter den Select Eingang 0, wir bekommen für eine kurze Zeit 0 am Ausgang



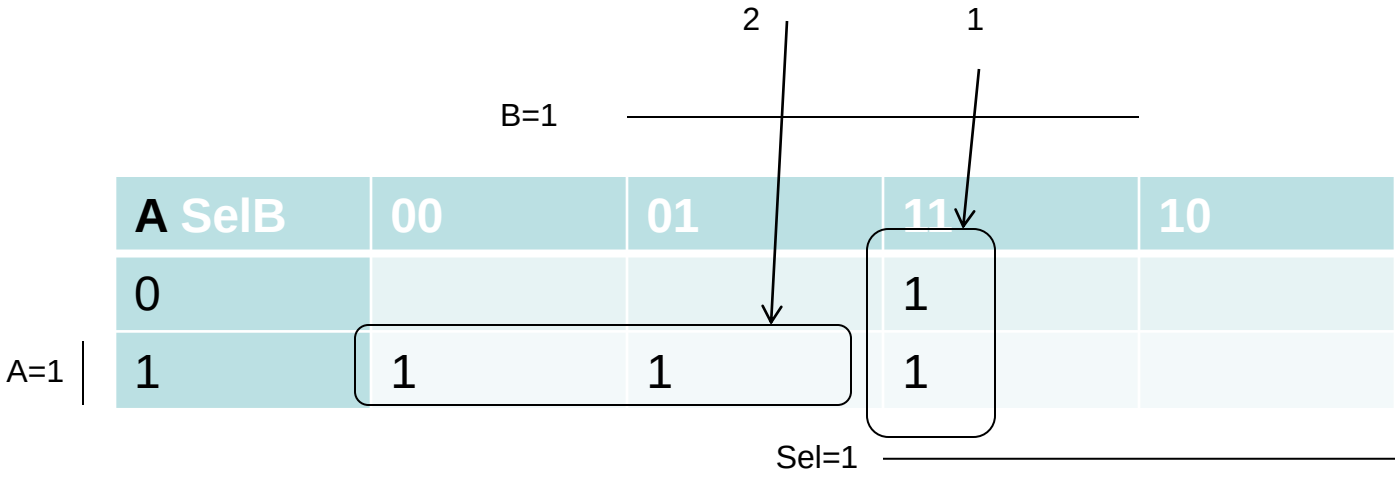
- Ist Glich ein Problem?
- Synchrone Schaltungen: Unproblematisch falls es kürzere Zeit als eine Taktperiode dauert
- Problem: Ausgänge der Statemaschine ohne Register

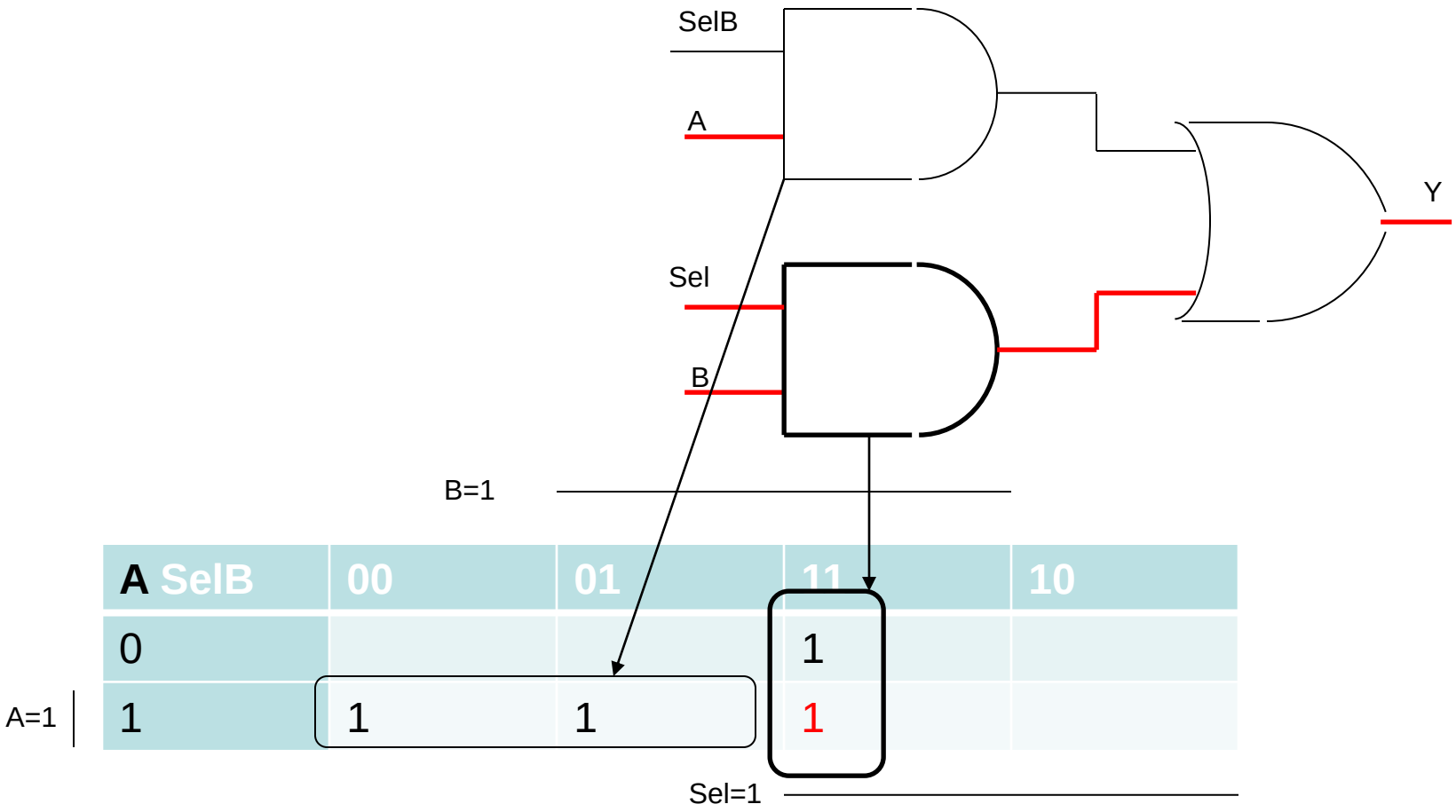


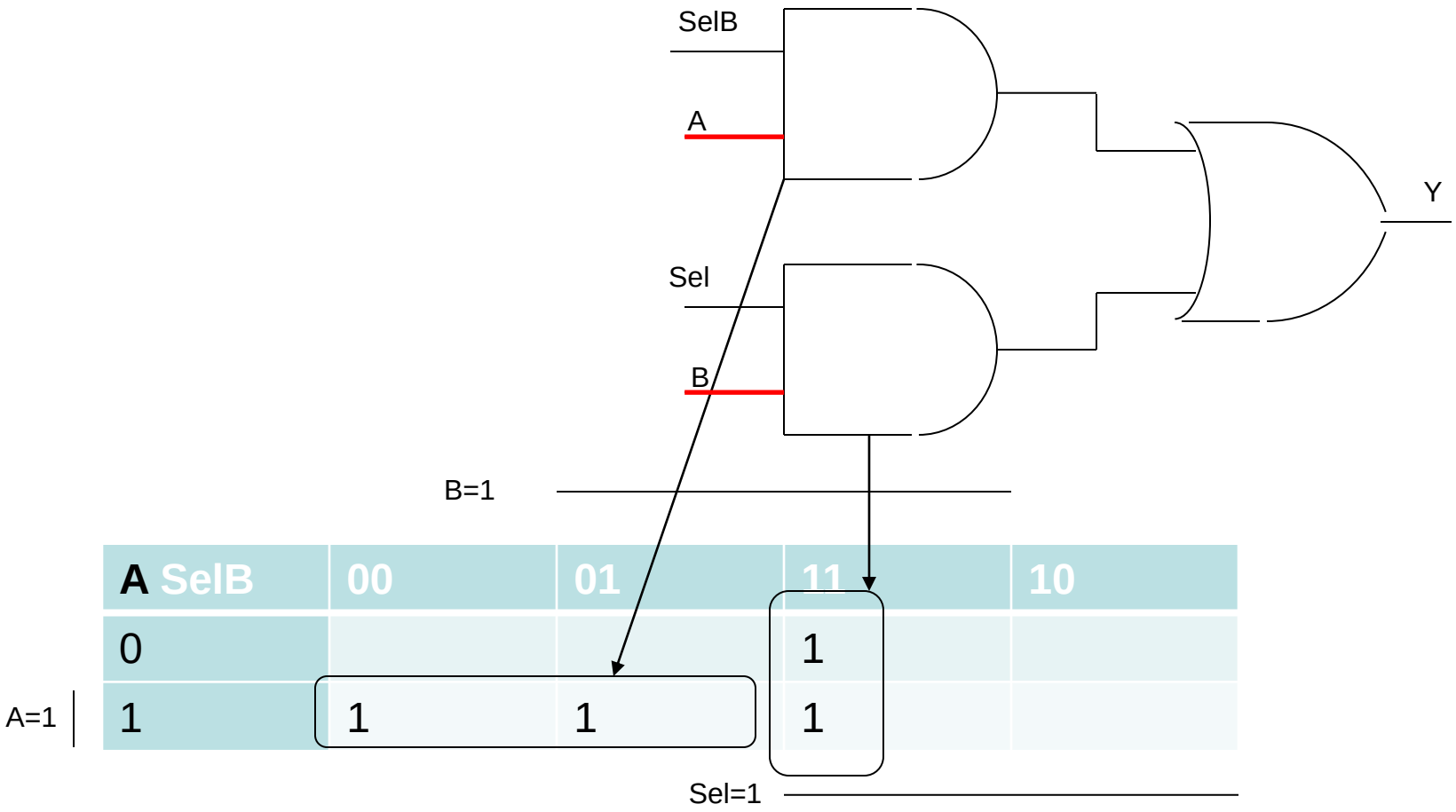




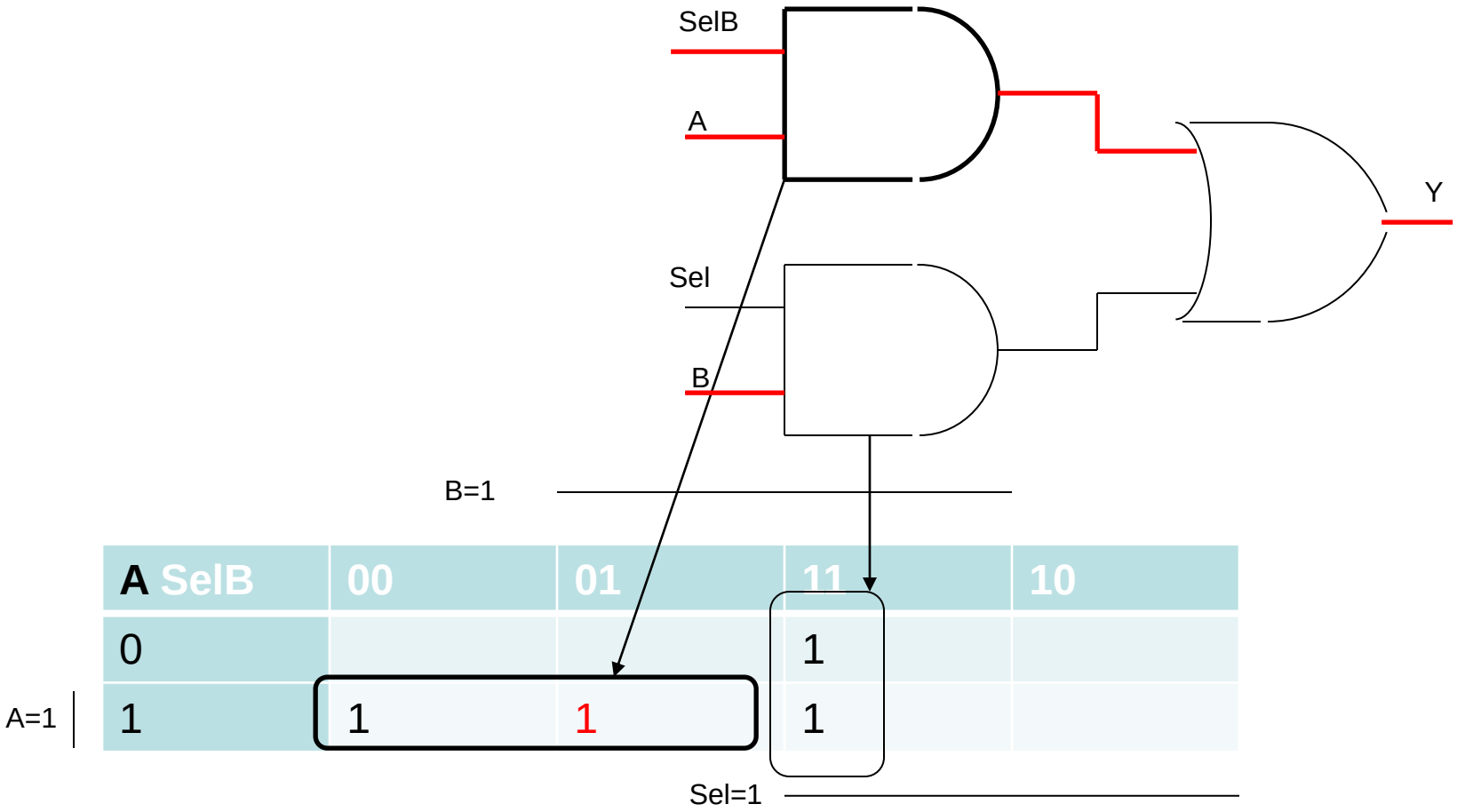
- Man kann die Möglichkeit eines Glitch-es aus Karnaugh Tabelle erkennen
- Zwei Gruppen sind getrennt und liegen nah einander.
- Wenn sich die Variable Sel von 1 auf 0 oder 1 auf 0 ändert, für A = B = 1, wird die Gruppe 1 „ausgeschaltet“ (bzw. ihre UND Funktion wird 0) und 2 eingeschaltet (bzw. ihre UND Funktion wird 1)
- Wenn das nicht synchron passiert, können wir 0 als Glitch bekommen.



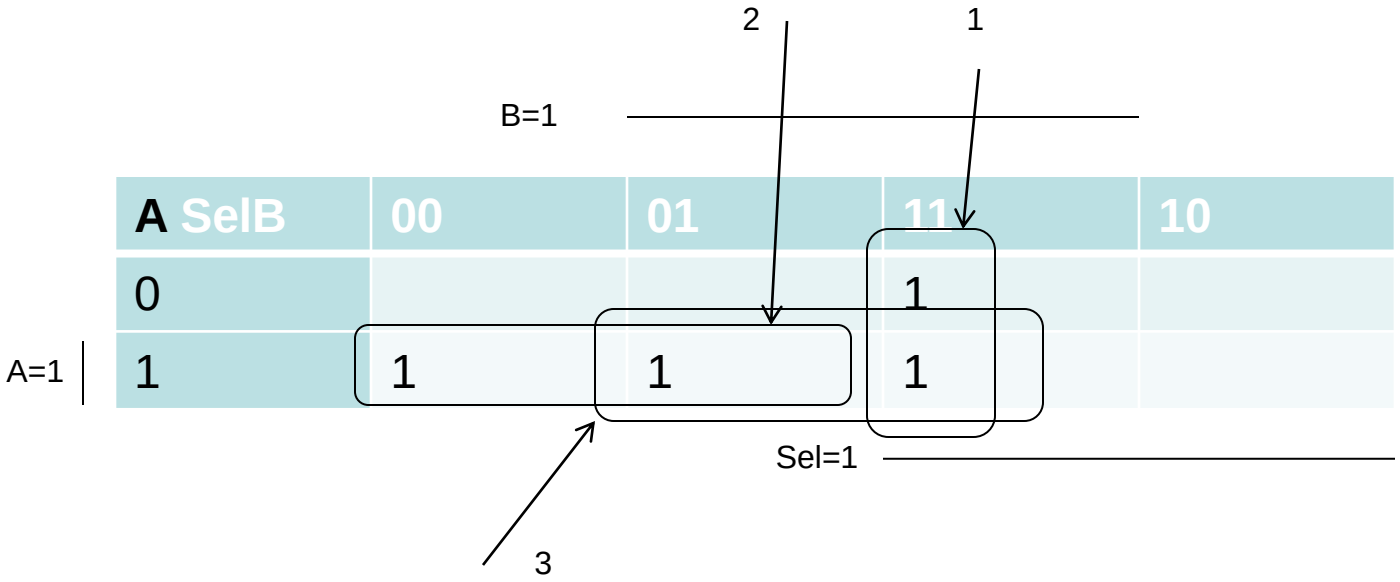

SelB
A
Y
Sel
B
B=1
A SelB
00
01
11
10
0
1
A=1
1
1
1
Sel=1



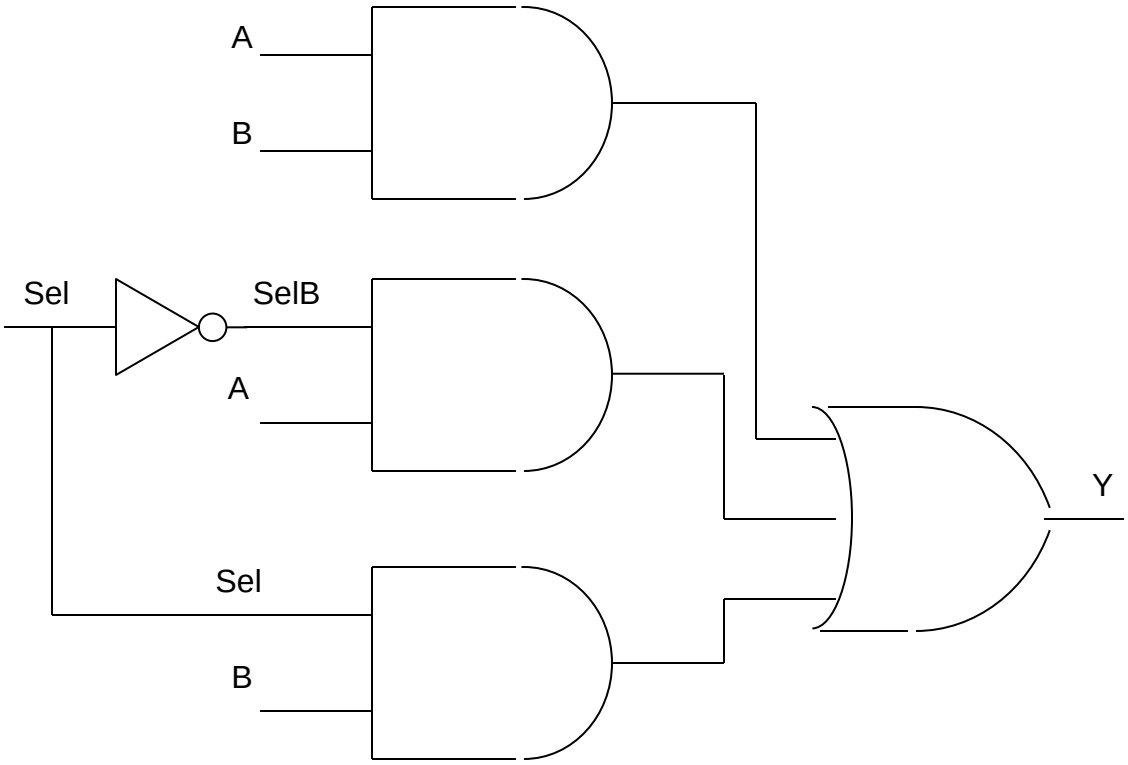
A SelB	00	01	11	10
0			1	
1	1	1	1	


SelB
A
Y
Sel
B
B=1
A SelB
00
01
11
10
0
1
1
1
1
A=1
Sel=1

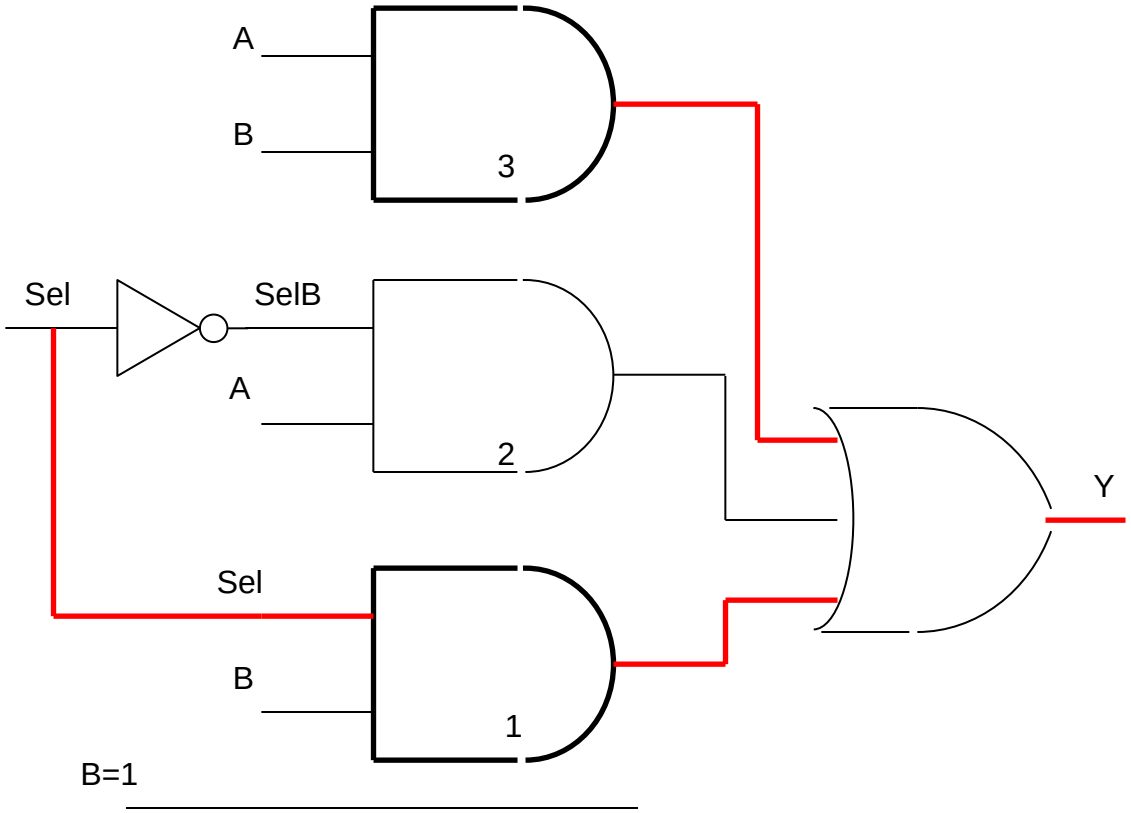
- Man kann ein Glitch verhindern indem man eine zusätzliche Gruppe 3 hinzufügt die als „Brücke“ zwischen den Gruppen 1 und 2 dient.
- A & B
- Bei Sel Änderung (für A = B = 1) wird die Gruppe 3 nicht ausgeschaltet – Select ist nicht als Variable vorhanden. Das verhindert ein 0-Glitch.



- Glitch-freie Schaltungen sind normalerweise komplizierter als die minimalen Schaltungen



- …



A SelB	00	01	11	10
0			1	
1	1	1	1	

A=1

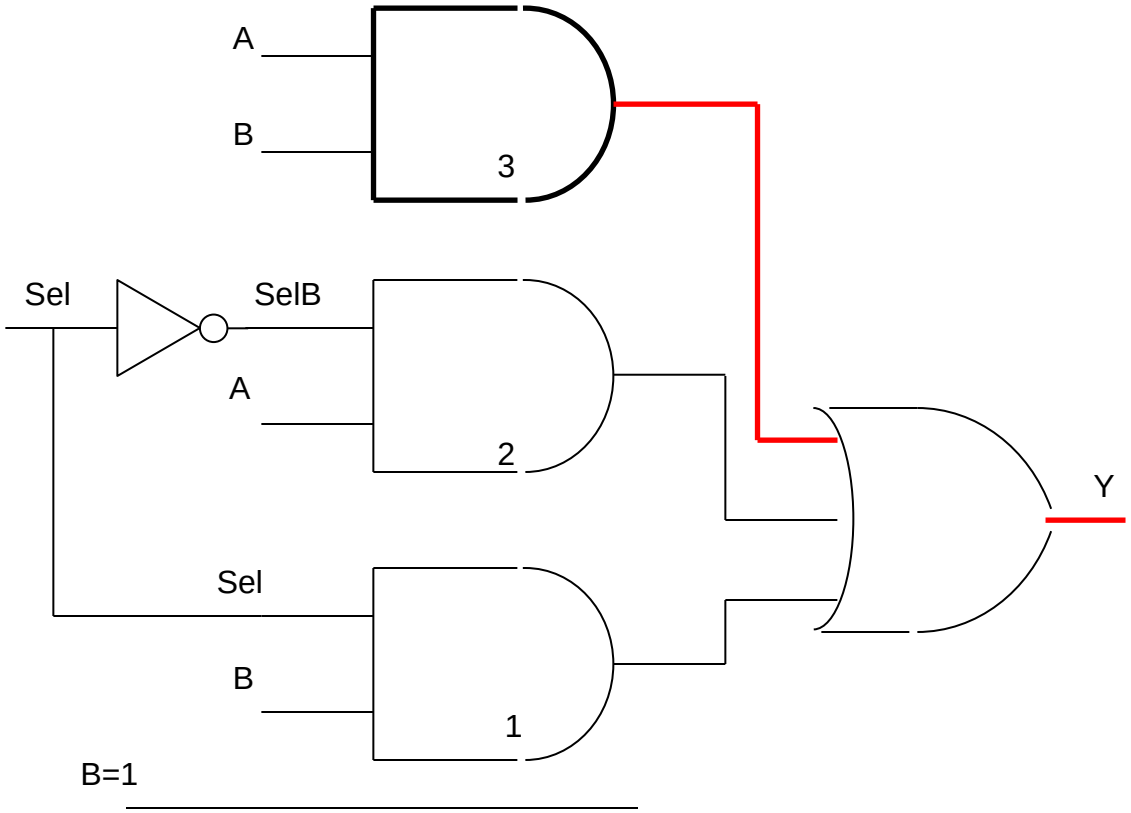
2

1

3

Sel=1

- …



A SelB	00	01	11	10
0			1	
1	1	1	1	

A=1

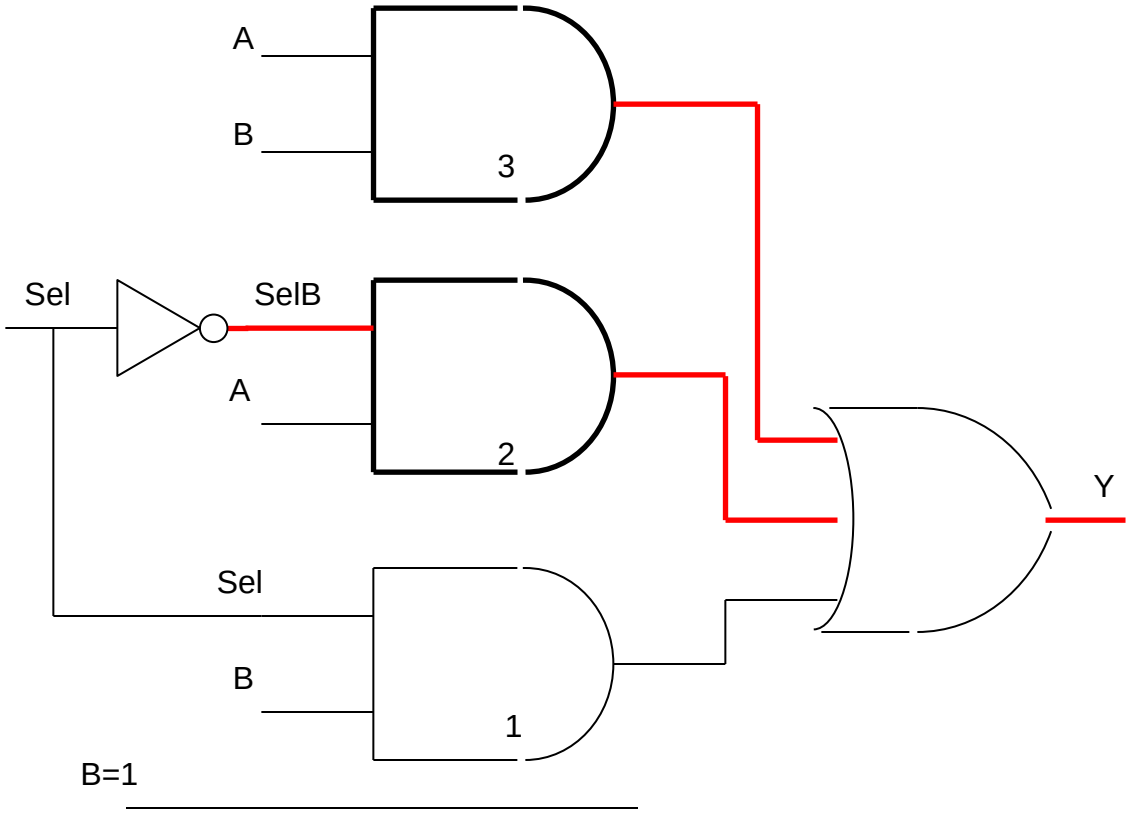
2

1

3

Sel=1

- …



A SelB	00	01	11	10
0			1	
1	1	1	1	

A=1

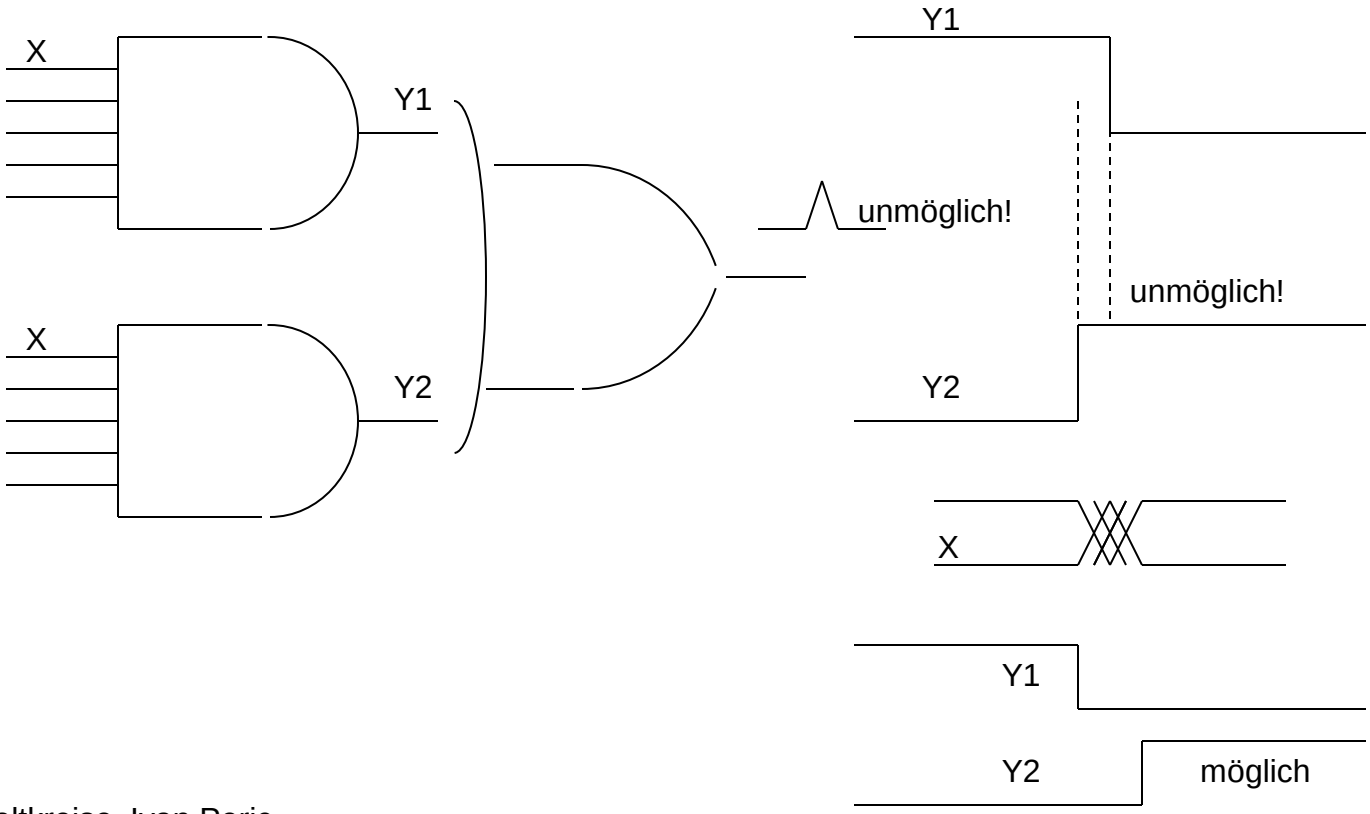
2

1

3

Sel=1

- Beachten wir, dass die kombinatorischen Schaltungen, implementiert als disjunktive Normalform, kein 1-Glitch erzeugen können (unter Annahme dass sich nur eine Eingangsvariable ändert)
- Einzige Möglichkeit für Glitch 1 wäre wenn ein UND zu früh logisch 1 wird und das andere zu spät 0 wird. Das kann nicht passieren wenn sich nur eine Variable ändert, weil UND erst dann 1 wird wenn alle Variablen 1 sind



- Kombinatorischen Schaltungen können auch als konjunktive Normalform implementiert werden.
- UND Verknüpfung von vielen ODER Funktionen.
- Mit ODER „Summen“ stellt man die Zeilen in der Wahrheitstabelle dar, die null sind (Variablen die 1 sind werden negiert)
- Eine Konjunktive Normalform kann keine 0-Glitches haben wenn sich nur eine Variable ändert.
- Konjunktive Normalform ist für die Funktionen geeignet die „viele Einsen“ als Ergebnis haben.
- Bsp. Y = !A || B || !C || D

Y = !A || B || !C || D

A=1

DC BA	00	01	11	10
00	1	1	1	1
01	1	0	1	1
11	1	1	1	1
10	1	1	1	1

C=1

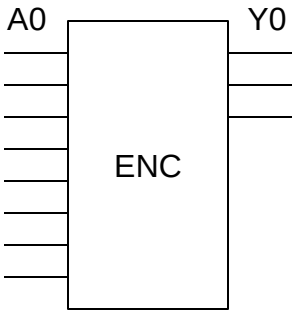
D=1

B=1

Kodierer


Snapshots at jasonlove.com
WARNING
Are you sure you want to reformat the hard drive?
Yes
No
© Love 2004

- Kodierer (Encoder)
- Um eine Information bearbeiten zu können, muss sie in einem geeignetem Zahlsystem z.B. im binären System dargestellt werden
- Bsp. Tastatur
- Jede Taste entspricht einem Digitaleingang
- Kodierer erzeugt den binären Code, der der aktivierten Taste entspricht
- Der einfachste Kodierer setzt voraus dass nur ein Eingang in einem Moment aktiv ist




```
case (A)
   8'b00000001   : Y = 0;
   8'b00000010   : Y = 1;
   8'b00000100   : Y = 2;
   8'b00001000   : Y = 3;
   8'b00010000   : Y = 4;
   8'b00100000   : Y = 5;
   8'b01000000   : Y = 6;
   8'b10000000   : Y = 7;
endcase
```

A0	A1	A2	A3	A4	A5	A6	A7	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0
	1							0	0	1
		1						0	1	0
			1					0	1	1
				1				1	0	0
					1			1	0	1
						1		1	1	0
							1	1	1	1

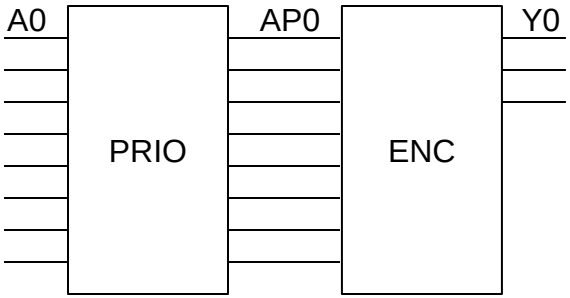
- Kodierer
- Y0 = A1 || A3 || A5 || A7
- Y1 = A2 || A3 || A6 || A7
- Y2 = A4 || A5 || A6 || A7
- Umgekehrte Funktionalität wie der Decoder

A0	A1	A2	A3	A4	A5	A6	A7	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0
	1							0	0	1
		1						0	1	0
			1					0	1	1
				1				1	0	0
					1			1	0	1
						1		1	1	0
							1	1	1	1

- Beachten wir dass der Kodierer nur dann richtig funktioniert, wenn nur ein Eingangssignal aktiv ist. Wenn z.B. A3 und A4 gleichzeitig aktiv werden, bekommen wir am Ausgang den Code Y0 = Y1 = Y2 = 1 -> 7 statt 3 oder 4.
- In den Systemen wo mehrere Eingänge gleichzeitig aktiv werden können werden die Prioritätskodierer benutzt. Diese erzeugen den Code des Eingangs mit höchster Priorität – z.B. den größeren Code.

A0	A1	A2	A3	A4	A5	A6	A7	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0
	1							0	0	1
		1						0	1	0
			1					0	1	1
				1				1	0	0
					1			1	0	1
						1		1	1	0
							1	1	1	1

- Man kann den Prioritätskodierer mithilfe eines einfachen Kodierers und eines Prioritäts-Netzwerks aufbauen. Das Prioritätsnetzwerk soll gewährleisten, dass nur ein Ausgang aktiv ist, ungeachtet von der Zahl der aktiven Eingängen. Z.B., wenn A3 und A4 Eingänge aktiv sind, soll nur AP4 aktiv werden.
- Das Prioritätsnetzwerk kann mit folgenden Funktionen beschrieben werden:
- AP7 = A7
- AP6 = A6 & !A7
- AP5 = A5 & !A6 & !A7
- …
- AP0 = A0 & !A1 & !A2 & !A3 & !A4 & !A5 & !A6 & !A7




```
if(A[7]) Y = 7;
else if(A[6]) Y = 6;
else if(A[5]) Y = 5;
else if(A[4]) Y = 4;
else if(A[3]) Y = 3;
else if(A[2]) Y = 2;
else if(A[1]) Y = 1;
else if(A[0]) Y = 0;
else Y = 0;
```

- AP7 = A7
- AP6 = A6 & !A7
- AP5 = A5 & !A6 & !A7
- …
- AP0 = A0 & !A1 & !A2 & !A3 & !A4 & !A5 & !A6 & !A7