

Digitaltechnik

Wintersemester 2024/2025

Vorlesung 2



TECHNISCHE
UNIVERSITÄT
DARMSTADT



ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING










- 1 Umrechnen zwischen Zahlensystemen
- 2 Addition von vorzeichenlosen Binärzahlen
- 3 Vorzeichenbehaftete Binärzahlen
- 4 Logikgatter



Harris 2016
Kap. 1.4, 4.2.7,
1.5, 4.2.1

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

- 1 Umrechnen zwischen Zahlensystemen
- 2 Addition von vorzeichenlosen Binärzahlen
- 3 Vorzeichenbehaftete Binärzahlen
- 4 Logikgatter

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



$$16^0 = 8^0 = 2^0 = 1$$

$$2^1 = 2$$

$$2^2 = 4$$

$$8^1 = 2^3 = 8$$

$$16^1 = 2^4 = 16$$

$$2^5 = 32$$

$$8^2 = 2^6 = 64$$

$$2^7 = 128$$

$$16^2 = 2^8 = 256$$

$$8^3 = 2^9 = 512$$

$$2^{10} = 1 \text{ Ki}$$

$$2^{11} = 2 \text{ Ki}$$

$$16^3 = 8^4 = 2^{12} = 4 \text{ Ki}$$

$$2^{13} = 8 \text{ Ki}$$

$$2^{14} = 16 \text{ Ki}$$

$$8^5 = 2^{15} = 32 \text{ Ki}$$

$$16^4 = 2^{16} = 64 \text{ Ki}$$

$$16^5 = 2^{20} = 1 \text{ Mi}$$

$$8^{10} = 2^{30} = 1 \text{ Gi}$$

$$16^{10} = 2^{40} = 1 \text{ Ti}$$



- polyadische Abbildung anwenden:

- $u_{2,5}(1\ 0011_2) = 2^0 + 2^1 + 2^4 = 19_{10}$

- $u_{16,3}(4AF_{16}) = 15 \cdot 16^0 + 10 \cdot 16^1 + 4 \cdot 16^2 = 1199_{10}$



$0000_2 =$	0_{10}	$= 0_{16}$
$0001_2 =$	1_{10}	$= 1_{16}$
$0010_2 =$	2_{10}	$= 2_{16}$
$0011_2 =$	3_{10}	$= 3_{16}$
$0100_2 =$	4_{10}	$= 4_{16}$
$0101_2 =$	5_{10}	$= 5_{16}$
$0110_2 =$	6_{10}	$= 6_{16}$
$0111_2 =$	7_{10}	$= 7_{16}$
$1000_2 =$	8_{10}	$= 8_{16}$
$1001_2 =$	9_{10}	$= 9_{16}$
$1010_2 =$	10_{10}	$= A_{16}$
$1011_2 =$	11_{10}	$= B_{16}$
$1100_2 =$	12_{10}	$= C_{16}$
$1101_2 =$	13_{10}	$= D_{16}$
$1110_2 =$	14_{10}	$= E_{16}$
$1111_2 =$	15_{10}	$= F_{16}$



- Nibble-weise umwandeln
- bei least significant bit beginnen
- führende Nullen weglassen oder ergänzen (je nach geforderter Bitbreite)
- $11\ 1010\ 0110\ 1000_2 = 3A68_{16}$
- $7BF_{16} = 111\ 1011\ 1111_2$

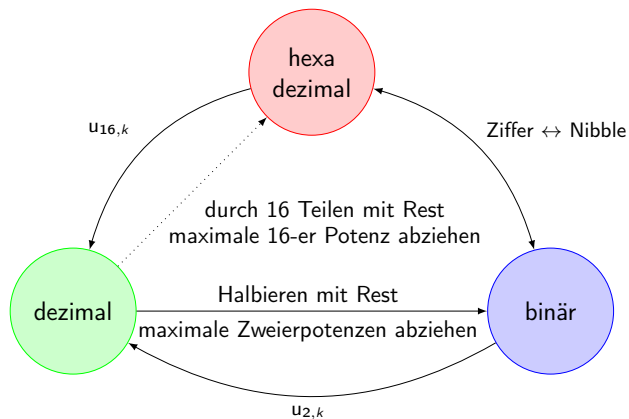


- Methode 1
(links nach rechts):
maximale Zweierpotenzen
abziehen

$$\begin{aligned} & 53_{10} \\ &= 32 + \underline{21} \\ &= 32 + 16 + \underline{5} \\ &= 32 + 16 + 4 + 1 \\ &= 2^5 + 2^4 + 2^2 + 2^0 \\ &= 11\ 0101_2 \end{aligned}$$










- Methode 2
(rechts nach links):
Halbieren mit Rest (sukzessives Durch-2-Teilen)

$$\begin{aligned} & 53_{10} \\ &= 2 \cdot \underline{26} + 1 \\ &= 2 \cdot (2 \cdot \underline{13} + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot \underline{6} + 1) + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot (2 \cdot \underline{3} + 0) + 1) + 0) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot (2 \cdot (\underline{1} + \underline{1}) + \underline{0}) + \underline{1}) + \underline{0}) + \underline{1}) + \underline{1} \\ &= 11\ 0101_2 \end{aligned}$$



Zweierpotenzen verinnerlichen!

- 1 Umrechnen zwischen Zahlensystemen
- 2 Addition von vorzeichenlosen Binärzahlen
- 3 Vorzeichenbehaftete Binärzahlen
- 4 Logikgatter

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



■ Dezimal

		1	1		Übertrag
	3	7	3	4	Summand
+	5	1	6	8	Summand
<hr/>					
=	8	9	0	2	Summe

■ Binär

		1	1		Übertrag
	1	0	1	1	Summand
+	0	0	1	1	Summand
<hr/>					
=	1	1	1	0	Summe












■ Binär	1	1	1		Übertrag		
		1	0	1	1	Summand	
	+		0	1	1	0	Summand
<hr/>							
	=	1	0	0	0	1	Summe

Überlauf

- Digitale Systeme arbeiten i.d.R. mit festen Bitbreiten
 - Langzahlarithmetik nur in Software (Bitbreite nur durch verfügbaren Arbeitsspeicher beschränkt)
 - Overflow-flag zum Signalisieren arithmetischer Ausnahmen in Hardware
- Operation (bspw. Addition) läuft über, wenn Ergebnis nicht mit der verfügbaren Bitbreite dargestellt werden kann
- für 4 bit Addierer gilt zum Beispiel: $11 + 6 = 1$ (siehe oben)

- 1 Umrechnen zwischen Zahlensystemen
- 2 Addition von vorzeichenlosen Binärzahlen
- 3 Vorzeichenbehaftete Binärzahlen
- 4 Logikgatter

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



Definition: Zweierkomplement

Die Funktion s_k bildet eine Bitfolge der Breite $k \in \mathbb{N}$ auf eine ganze Zahl ab:

$$s_k : (a_{k-1} \dots a_1 a_0) \in \mathbb{B}^k \mapsto a_{k-1} \cdot (-2^{k-1}) + \sum_{i=0}^{k-2} a_i \cdot 2^i \in \mathbb{Z}$$

- auch für Basen $b > 2$ verallgemeinerbar: $s_{b,k}$
 - wird aber heute kaum noch verwendet



- niedrigstwertige Stelle: a_0
- höchstwertige Stelle: a_{k-1}
- kleinste darstellbare Zahl: $1 \cdot (-2^{k-1}) + \sum_{i=0}^{k-2} 0 \cdot 2^i = -2^{k-1}$
- größte darstellbare Zahl: $0 \cdot (-2^{k-1}) + \sum_{i=0}^{k-2} 1 \cdot 2^i = 2^{k-1} - 1$
- Anzahl der darstellbaren Werte: 2^k
- eineindeutig (bijektiv) abbildbar auf Wertebereich $\{-2^{k-1}, \dots, 2^{k-1} - 1\}$ für festes k



■ Beispiele

$$s_4(1010_2) = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot (-2^3) = -6_{10}$$

$$s_4(0110_2) = 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot (-2^3) = +6_{10}$$

■ *kompatibel* mit binärer (unsigned) Addition:

$$\begin{array}{rcccccc} & & 1 & 1 & & & \\ & & 1 & 0 & 1 & 0 & = -6_{10} \\ + & & 0 & 1 & 1 & 0 & = +6_{10} \\ \hline = & 1 & 0 & 0 & 0 & 0 & = 0_{10} \checkmark \end{array}$$

■ *kein Überlauf* bei Addition positiver und negativer Zahl gleicher Breite



- Methode 1 (links nach rechts):
größtmögliche Zweierpotenzen
abziehen

$$\begin{aligned}-53_{10} &= -64 + \underline{11} \\ &= -64 + 8 + \underline{3} \\ &= -64 + 8 + 2 + 1 \\ &= -2^6 + 2^3 + 2^1 + 2^0 \\ &= 100\ 1011_2\end{aligned}$$

- Methode 2 (rechts nach links):
Betrag negieren =
Komplement (bitweise \bar{a})
und Inkrement (+1)
(Reihenfolge beachten!)

$$\begin{aligned}-53_{10} &= \overline{53_{10}} + 1 \\ &= \overline{011\ 0101_2} + 1 \\ &= 100\ 1010_2 + 1 \\ &= 100\ 1011_2\end{aligned}$$

- in beiden Fällen auf korrekte/geforderte Bitbreite achten
- ggf. müssen führende Null(en) schon für Betragsdarstellung eingefügt werden

- Methode 1:
polyadische Abbildung
direkt anwenden

$$\begin{aligned}100\ 1011_2 \\&= -2^6 + 2^3 + 2^1 + 2^0 \\&= -64 + 8 + 2 + 1 \\&= -53_{10}\end{aligned}$$

- Methode 2:
Betrag berechnen =
Komplement (bitweise \bar{a})
und Inkrement (+1)
Nur falls MSB=1

$$\begin{aligned}100\ 1011_2 &= -(\overline{100\ 1011_2} + 1) \\&= -(011\ 0100_2 + 1) \\&= -011\ 0101_2 \\&= -53_{10}\end{aligned}$$

- Methode 2:
Betrag berechnen =
direkt
Nur falls MSB=0

$$000\ 1011_2 = 11$$

- notwendig, um unterschiedlich breite Bitfolgen zu addieren
- *zero extension*:
 - Auffüllen mit führenden Nullen für vorzeichenlose Darstellung

$$u_{2,k+1}(0a_{k-1} \dots a_0) = 0 \cdot 2^k + \sum_{i=0}^{k-1} a_i \cdot 2^i = u_{2,k}(a_{k-1} \dots a_0)$$

- *sign extension*:
 - Auffüllen mit Wert des Vorzeichen-Bits für Zweierkomplement Darstellung

$$\begin{aligned} s_{k+1}(a_{k-1}a_{k-1} \dots a_0) &= a_{k-1} \cdot \underbrace{(-2^k)}_{2 \cdot (-2^{k-1})} + a_{k-1} \cdot 2^{k-1} + \sum_{i=0}^{k-2} a_i \cdot 2^i \\ &= a_{k-1} \cdot (-2^{k-1} - 2^{k-1} + 2^{k-1}) + \sum_{i=0}^{k-2} a_i \cdot 2^i \\ &= s_k(a_{k-1} \dots a_0) \end{aligned}$$



- -5_{10} im Zweierkomplement von 4 auf 8 Bit erweitern:










$$\begin{aligned} 5_{10} &= 0101_2 \\ \Rightarrow -5_{10} &= \overline{0101_2} + 1 \\ &= 1010_2 + 1 \\ &= 1011_2 \\ &= \textcolor{red}{1111} 1011_2 \end{aligned}$$

$$\begin{aligned} \textit{Probe} : -(-5_{10}) &= \overline{1111 1011_2} + 1 = 0000 0100_2 + 1 \\ &= 0000 0101_2 \\ &= 5_{10} \end{aligned}$$

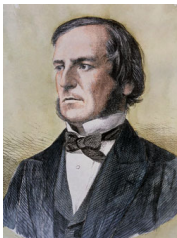
Quiz I - Die dunkle Bedrohung

(Hat nichts mit dem Moodle-Quiz zu tun)

- 1 Umrechnen zwischen Zahlensystemen
- 2 Addition von vorzeichenlosen Binärzahlen
- 3 Vorzeichenbehaftete Binärzahlen
- 4 Logikgatter

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

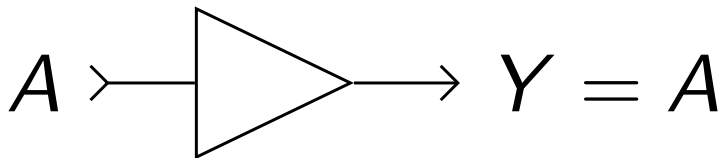
- in einfachen Verhältnissen geboren
 - brachte sich selbst Mathematik bei
 - Professor am Queen's College in Irland
 - „An Investigation of the Laws of Thought“ (1854)
- ⇒ grundlegende logische Variablen und Operationen



Harris 2016
Kap. 1.5, 4.2.1

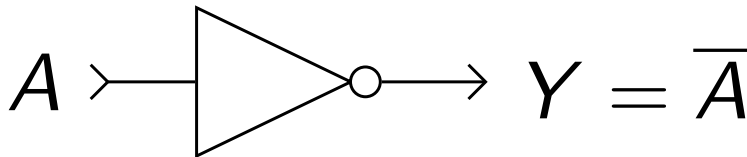


- verknüpfen binäre Werte: $\mathbb{B}^n \rightarrow \mathbb{B}^k$
- zunächst $k = 1$
- Beispiele für
 - $n = 1$: NOT
 - $n = 2$: AND, OR, XOR
 - $n = 3$: MUX
- Charakterisierung durch Wahrheitwertetabellen



A	Y
0	0
1	1

SystemVerilog: `assign Y = A;`



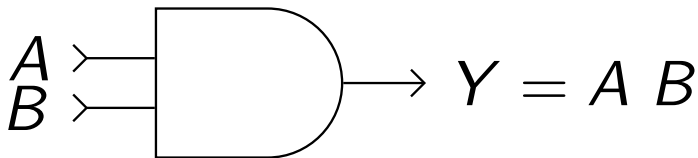
A	Y
0	1
1	0

alternativ: $Y = !A = \complement A = \neg A$ SystemVerilog: `assign Y = ~A;`

AND : $\mathbb{B}^2 \rightarrow \mathbb{B}$

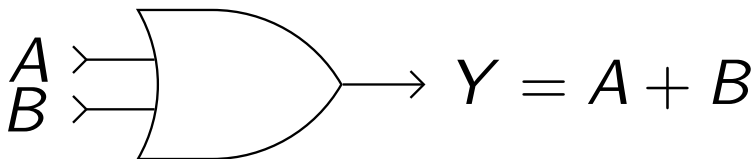


ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING



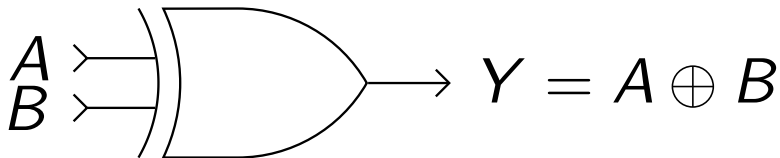
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

alternativ: $Y = A \cdot B = A \& B = A \wedge B$ SystemVerilog: `assign Y = A & B;`



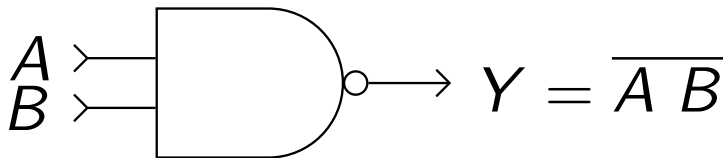
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

alternativ: $Y = A|B = A \vee B$ SystemVerilog: `assign Y = A | B;`



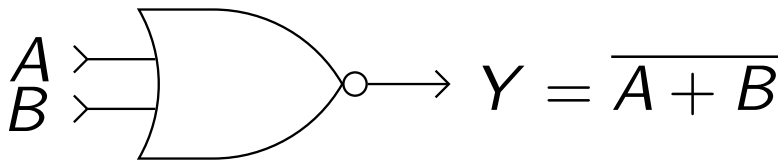
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

alternativ: $Y = A \hat{=} B$ SystemVerilog: `assign Y = A ^ B;`



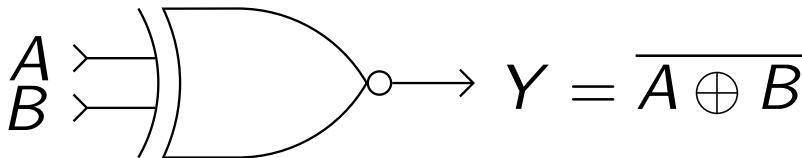
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

SystemVerilog: `assign Y = A ~& B;`



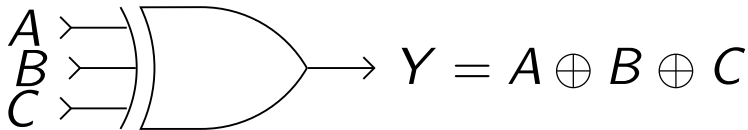
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

SystemVerilog: `assign Y = A ~| B;`



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

entspricht Test auf Gleichheit SystemVerilog: `assign Y = A ~^ B;`



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

SystemVerilog: `assign Y = A ^ B ^ C;`



- „repräsentiert“ die Anzahl der Einsen an Eingängen (modulo 2)

⇒ Paritätsfunktion $p : (a_{k-1} \dots a_0) \in \mathbb{B}^k \mapsto a_{k-1} \oplus \dots \oplus a_0 \in \mathbb{B}$

- $p(a) = 0 \Rightarrow$ Quersumme von a ist gerade
- $p(a) = 1 \Rightarrow$ Quersumme von a ist ungerade












- 1 Umrechnen zwischen Zahlensystemen
- 2 Addition von vorzeichenlosen Binärzahlen
- 3 Vorzeichenbehaftete Binärzahlen
- 4 Logikgatter

nächste Vorlesung beinhaltet

- physikalische Realisierung von Logikgattern

Moodle-Quiz zu Vorlesung 01 muss bis nächste Woche Mittwoch 12:00 abgegeben werden.

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen