SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

Systemarchitektur SS 2021

Lösungsskizze 4

Aufgabe 4.1: Abhängigkeitsbegriff

1. Im Folgenden werden wir mit x_i (wobei $1 \leq i \leq N$) das i-te element eines Tupels $x \in \mathbb{B}^N$ bezeichnen.

$$dep_f := \{(i,j) \mid \exists x, y \in \mathbb{B}^n : (f(x)_j \neq f(y)_j) \wedge (\forall i' : (i' \neq i) \Rightarrow (x_{i'} = y_{i'}))\}$$

2.

x_1	x_2	s_1	s_2
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	0

Aufgabe 4.2: Inkrementierer

1. Die rekursive Definition eines n-Inkrementierers ist in Abbildung 1 gezeigt[1].
2. Beweis durch Induktion über $n \in \mathbb{N}_+$:
   - Induktionsanfang: $n = 1 \Rightarrow a = a_0$:
     Inc_1 is ein Halbaddierer, der als Eingabe (a_0, c_0) hat. Ein Halbaddierer ist korrekt. $\Rightarrow$ Für die Ausgabe s gilt $\langle s = \langle c_1, s_0 \rangle = \langle a_0 \rangle + c_0$.
   - Induktionsvorraussetzung: Die Aussage sei für ein beliebiges aber festes $n \in \mathbb{N}_+$ gezeigt.
   - Induktionsschritt: $n - 1 \rightarrow n$
     Eingabe an Inc_n: $(a_{n-1}, ..., a_0)$
     Zeige für Ausgabe $(s_n, s_{n-1}, ..., s_0)$ gilt:
     $\langle s \rangle = \langle c_n, s_{n-1}, ..., s_0 \rangle = \langle a_{n-1}, ..., a_0 \rangle + c_0$.

$$\begin{aligned}
\langle c_n, s_{n-1}, \ldots, s_0 \rangle &= 2^{n-1} \langle c_n, s_{n-1} \rangle + \langle s_{n-2}, \ldots, s_0 \rangle \\
&= 2^{n-1} \cdot (\langle a_{n-1} \rangle + \langle c_{n-1} \rangle) + \langle s_{n-2}, \ldots, s_0 \rangle && \text{(Halbaddierer)} \\
&= 2^{n-1} \langle a_{n-1} \rangle + \langle c_{n-1}, s_{n-2}, \ldots, s_0 \rangle \\
&= 2^{n-1} \langle a_{n-1} \rangle + \langle a_{n-2}, \ldots, a_0 \rangle + \langle c_0 \rangle && \text{(I.V.)} \\
&= \langle a_{n-1}, \ldots, a_0 \rangle + \langle c_0 \rangle
\end{aligned}$$

■

3. Die Kosten und Tiefe sind:

$$C(Inc_1) = C(HA) = 2, \;\; C(Inc_n) = C(HA) + C(Inc_{n-1})$$

$$D(Inc_1) = D(HA) = 1, \;\; D(Inc_n) = D(HA) + D(Inc_{n-1})$$

[1] Entnommen aus W. J. Paul: Sysbook, SS 2013

4. • $C(Inc_n) = C(HA) * n = 2n$
   Beweis durch Induktion über n.
   IA: n = 1
   $$C(Inc_1) = C(HA) = 2 = 2 * 1 = 2n$$
   IV:
   $$\text{Für beliebiges aber festes } n \in \mathbb{N} : C(Inc_n) = 2n$$
   IS: n $\rightarrow$ n+1
   $$C(Inc_{n+1}) = C(HA) + C(Inc_n) \overset{IV}{=} C(HA) + 2n = 2 + 2n = 2(n+1)$$
   $\blacksquare$

   • $D(Inc_n) = D(HA) * n = n$
   Beweis durch Induktion über n.
   IA: n = 1
   $$D(Inc_1) = D(HA) = 1 = n$$
   IV:
   $$\text{Für beliebiges aber festes } n \in \mathbb{N} : D(Inc_n) = n$$
   IS: n $\rightarrow$ n+1
   $$D(Inc_{n+1}) = D(HA) + D(Inc_n) \overset{IV}{=} D(HA) + n = 1 + n = (n+1)$$
   $\blacksquare$

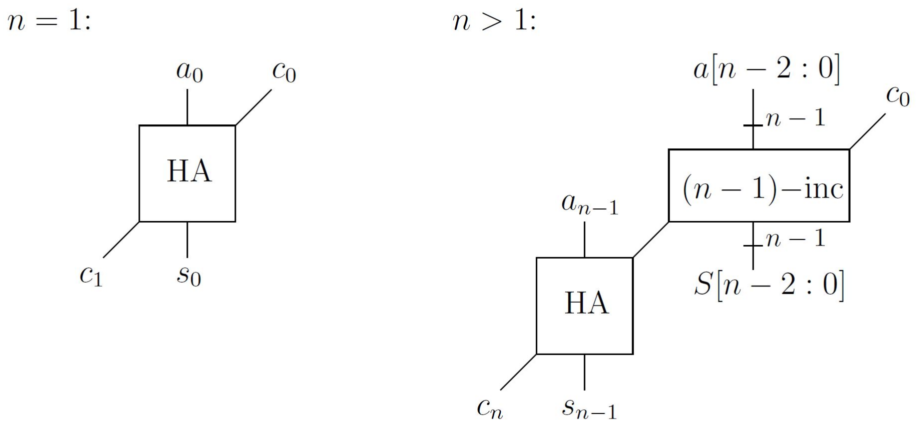
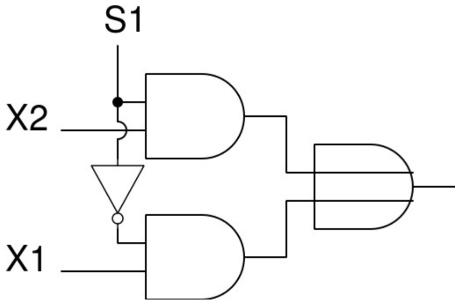


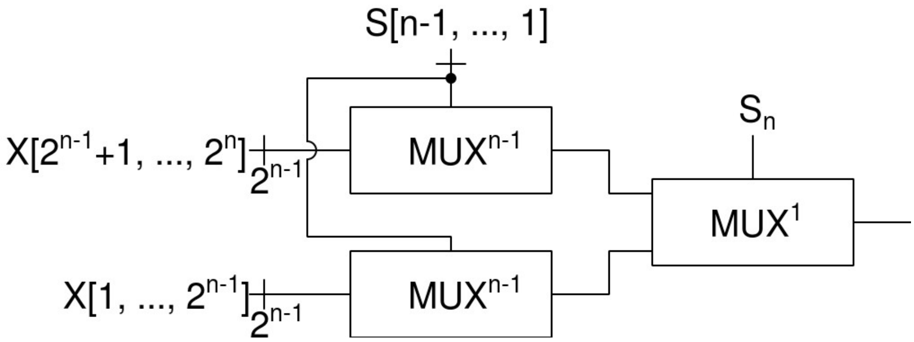
Abbildung 1: n-Inkrementierer

Aufgabe 4.3: Modellierung durch Schaltkreise

(a) Konstruktion:



(b) Konstruktion:



(c) Ein so konstruierter MUX^n hat die Tiefe $3 + 2(n - 1) = 2n + 1$ und besteht aus $2^n - 1$ einfachen MUXen.

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

System Architecture SS 2021

Solution Sketch 4

Problem 4.1: Notion of Dependency

1. In the following, we will use x_i (with $1 \leq i \leq N$) to refer to the i-th element of a tuple $x \in \mathbb{B}^N$.

$$dep_f := \{(i,j) \mid \exists x, y \in \mathbb{B}^n : (f(x)_j \neq f(y)_j) \wedge (\forall i' : (i' \neq i) \Rightarrow (x_{i'} = y_{i'}))\}$$

2.

x_1	x_2	s_1	s_2
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	0

Problem 4.2: Incrementer

1. The recursive definition of an n-bit incrementer is shown in Figure 2[2].

2. Proof by induction on $n \in \mathbb{N}_+$:
   - Base case: $n = 1 \Rightarrow a = a_0$:
     Inc_1 is a half adder which has as input (a_0, c_0). A half adder is correct. Thus, for the output s we have $\langle s \rangle = \langle c_1, s_0 \rangle = \langle a_0 \rangle + c_0$.
   - Induction hypothesis (IH): The statement holds for an arbitrary but fixed $n \in \mathbb{N}_+$.
   - Induction step: $n - 1 \to n$
     Input for Inc_n: $(a_{n-1}, ..., a_0)$
     We have to show that for the output $s = c_n s_{n-1} \dots s_0$ the following holds:
     $\langle s \rangle = \langle c_n, s_{n-1}, ..., s_0 \rangle = \langle a_{n-1}, ..., a_0 \rangle + c_0$.

$$\begin{aligned}
\langle c_n, s_{n-1}, \dots, s_0 \rangle &= 2^{n-1}\langle c_n, s_{n-1} \rangle + \langle s_{n-2}, \dots, s_0 \rangle \\
&= 2^{n-1} \cdot (\langle a_{n-1} \rangle + \langle c_{n-1} \rangle) + \langle s_{n-2}, \dots, s_0 \rangle && \text{(Half adder)} \\
&= 2^{n-1}\langle a_{n-1} \rangle + \langle c_{n-1}, s_{n-2}, \dots, s_0 \rangle \\
&= 2^{n-1}\langle a_{n-1} \rangle + \langle a_{n-2}, \dots, a_0 \rangle + \langle c_0 \rangle && \text{(IH)} \\
&= \langle a_{n-1}, \dots, a_0 \rangle + \langle c_0 \rangle
\end{aligned}$$

∎

3. The cost and depth are:

$$C(Inc_1) = C(HA) = 2, \;\; C(Inc_n) = C(HA) + C(Inc_{n-1})$$

$$D(Inc_1) = D(HA) = 1, \;\; D(Inc_n) = D(HA) + D(Inc_{n-1})$$

[2] Taken from W. J. Paul: Sysbook, SS 2013

4.
- $C(Inc_n) = C(HA) * n = 2n$
  Proof by induction on n.
  Base case: n = 1
  $$C(Inc_1) = C(HA) = 2 = 2 * 1 = 2n$$
  Induction hypothesis (IH):
  $$\text{For an arbitrary but fixed } n \in \mathbb{N} : C(Inc_n) = 2n$$
  Induction step: n $\rightarrow$ n+1
  $$C(Inc_{n+1}) = C(HA) + C(Inc_n) \stackrel{IH}{=} C(HA) + 2n = 2 + 2n = 2(n+1)$$
  $\blacksquare$

- $D(Inc_n) = D(HA) * n = n$
  Proof by induction on n.
  Base case: n = 1
  $$D(Inc_1) = D(HA) = 1 = n$$
  Induction hypothesis (IH):
  $$\text{For an arbitrary but fixed } n \in \mathbb{N} : D(Inc_n) = n$$
  Induction step: n $\rightarrow$ n+1
  $$D(Inc_{n+1}) = D(HA) + D(Inc_n) \stackrel{IH}{=} D(HA) + n = 1 + n = (n+1)$$
  $\blacksquare$

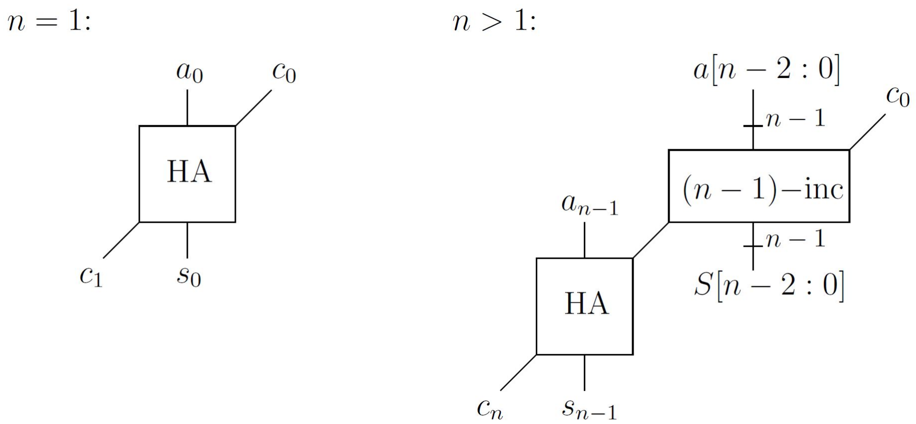
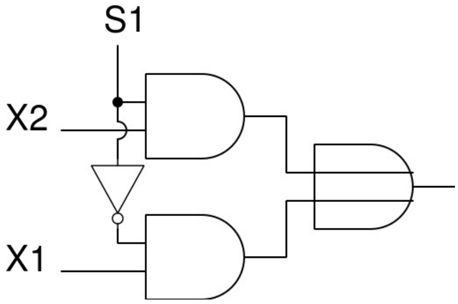


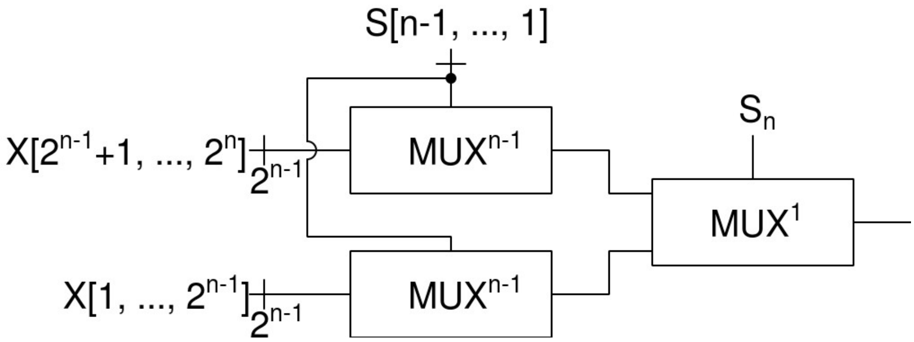
Figure 2: n-bit incrementer

Problem 4.3: Modeling Through Circuits

(a) Construction:



(b) Construction:



(c) A MUX^n constructed in this way has depth $3 + 2(n-1) = 2n + 1$ and consists of $2^n - 1$ simple MUXes.