Klausur
EDA
Prof. Dr.-Ing. Peter Gregorius

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Name, Vornamen	[redacted]
Matrikelnummer	[redacted]
Studiengang	TI - B
Unterschrift	[redacted] **Tag der Prüfung:** 26. Januar 2016

Bitte beachten!

1. Prüfen Sie, ob Ihre Klausur vollständig ist. Sie muss aus den durchnummerierten Blättern von 1 bis 12 bestehen. Nehmen Sie die Klausur bitte nicht auseinander. Falls Sie ein unvollständiges Exemplar erhalten haben, lassen Sie sich bitte eine einwandfreie Klausur aushändigen.
2. Zum Bestehen der Klausur sind 50% der Punktzahl erforderlich.
3. Die Bearbeitungszeit beträgt 90 Minuten.
4. Außer einfachen (nicht programmierbaren) Taschenrechnern sind keine Hilfsmittel zugelassen.
5. Das Betreiben von Mobiltelefonen und Computern sind im Prüfungsraum nicht erlaubt bzw. grundsätzlich auszuschalten.
6. Schreiben Sie bitte gut leserlich und nicht mit Bleistift. Ihre Klausur wird ansonsten nicht gewertet. Lassen Sie einen Korrekturrand von mindestens 4 cm frei.

Mit der Unterschrift bestätigen Sie, dass Sie prüfungsfähig sind und bei Beginn der Klausur die vollständigen Unterlagen erhalten habe.

Anmerkung: Maximale Punktzahl= 120 Punkte, 100% = 100 Punkte; Note 1.0 ≥ 95%
(Punkte/Note: 95/1,0; 90/1,3; 85/1,7; 80/2,0; 75/2,3; 70/2,7; 65/3,0; 60/3,3; 55/3,7; 50/4.0)

Aufgabe	1	2	3			
erreichbare Punkte	40	40	40			
erreichte Punkte	30	35	15			

Zusatzleistung: — Punkte: 80 Note: 2,0

Ort und Datum: Berlin, 26.01.16 Unterschrift: [redacted]

Punkte
40

Aufgabe 1 Einfache CRC-Codierung (7/4)

Gegeben ist folgendes Generator-Polynom:

$$G(u) = 1 \cdot u^3 + 1 \cdot u^0 \tag{1}$$

Aufgabenstellung:

a) Berechnen Sie alle Codewörter. Tragen Sie ihre Ergebnisse in die gegebene Wahrheitstabelle ein (10 Pkt.).

b) Entwicklen Sie eine kombinatorische Schaltung zur Erzeugung des Codewortes mit minimaler Gatteranzahl. Erlaubt sind Gatter mit zwei Eingängen (AND, NAND, OR, NOR, XOR, XNOR) (10 Pkt.).

c) Geben Sie das Schaltbild mit Eingangs- und Ausgangsregister (Register-Kombinatorik-Register) an (10 Pkt.).

d) Entwicklen Sie eine VHDL-Beschreibung zur Realisierung der Codeworterzeugung. Für die Registerstufen sind Master-Slave D-FF zu verwenden, aktiv mit der steigenden Flanke. Die Register werden mit einem *High*-aktiven RESET-Signal zurückgesetzt. (10 Pkt.).

Gegeben ist folgende Wahrheitstabelle:

	n= 7 Codewort						
	m= 4				**k= 3**		
	x_3	x_2	x_1	x_0			
$(i)_{10}$	y_6	y_5	y_4	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	1	0
3	0	0	1	1	0	1	1
4	0	1	0	0	1	0	0
5	0	1	0	1	1	0	1
6	0	1	1	0	1	1	0
7	0	1	1	1	1	1	1
8	1	0	0	0	0	0	1
9	1	0	0	1	0	0	0
10	1	0	1	0	0	1	1
11	1	0	1	1	0	1	0
12	1	1	0	0	1	0	1
13	1	1	0	1	1	0	0
14	1	1	1	0	1	1	1
15	1	1	1	1	1	1	0

Hinweis zur Berechnung des Restwertes:

y_6	y_5	y_4	y_3	y_2	y_1	y_0	÷	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0	÷	1	0	0	1

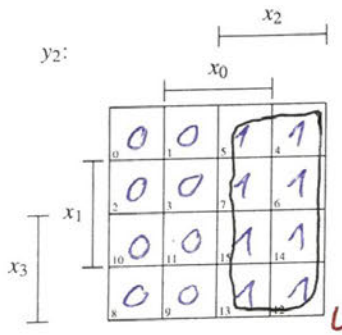
y_6	y_5	y_4	y_3	y_2	y_1	y_0	÷	c_3	c_2	c_1	c_0
0	0	0	1	0	0	0	÷	1	0	0	1

y_6	y_5	y_4	y_3	y_2	y_1	y_0	÷	c_3	c_2	c_1	c_0
0	0	1	0	0	0	0	÷	1	0	0	1

y_6	y_5	y_4	y_3	y_2	y_1	y_0	÷	c_3	c_2	c_1	c_0
0	0	1	1	0	0	0	÷	1	0	0	1

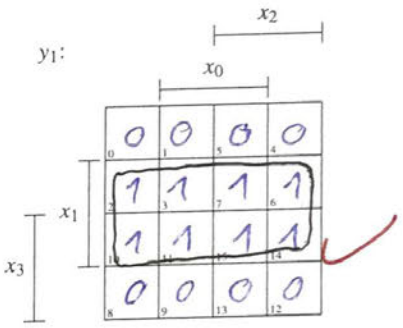
usw.

y_2:



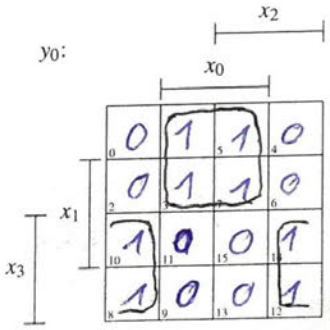
$y_2 = x_2$

y_1:



$y_1 = x_1$

y_0:



$y_0 = (x_0 \wedge \neg x_3) \vee (\neg x_0 \wedge x_3)$

$y_0 = x_0 \not\leftrightarrow x_3$

- Lückentext für das Eingangsregister

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.std_logic_unsigned.all;
4  USE ieee.numeric_std.ALL;
5
6  ENTITY PIPO4Bit is
7      port(
8           CLK, CLR : IN  std_logic;
9           D        : IN  std_logic_vector(6 downto 0);
10          Q        : OUT std_logic_vector(6 downto 0)
11     );
12 end PIPO4Bit;
13
14 ARCHITECTURE PIPO4Bit_arc OF PIPO4Bit IS
15 SIGNAL ...............................
16 SIGNAL ...............................
17 BEGIN
18     PIPO4Bit: PROCESS ..... IS
19     BEGIN
20         .........
21         .........
22         .........
23         .........
24     END PROCESS PIPO4Bit;
25     ............
26 END PIPO4Bit_arc;
```

- Lückentext für das Ausgangsregister

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.std_logic_unsigned.all;
4  USE ieee.numeric_std.ALL;
5
6  ENTITY PIPO7Bit is
7      port(
8           .........
9           .........
10          .........
11     );
12 end PIPO7Bit;
13
14 ARCHITECTURE PIPO7Bit_arc OF PIPO7Bit IS
15 SIGNAL ...................................
16 SIGNAL ...................................
17 BEGIN
18     PIPO7Bit : PROCESS ..... IS
19     BEGIN
20         ..........................................
21         ..........................................
22         ..........................................
23         ..........................................
24     END PROCESS PIPO7Bit;
25     ..............
26 END PIPO7Bit_arc;
```

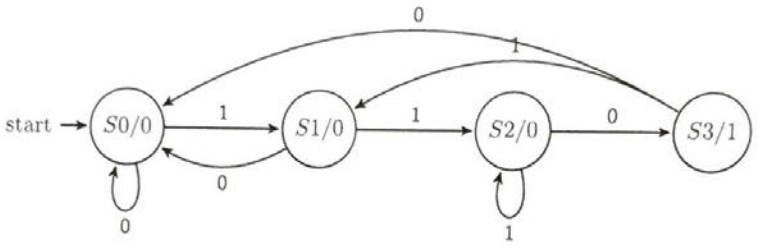
- Lückentext für die Einheit zur CRC-Generierung

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY CRC74 is
    port(
         CLK, CLR  : IN  std_logic;
         x   : IN  std_logic_vector(3 downto 0);
         y   : OUT std_logic_vector(6 downto 0)
        );
end CRC74;

ARCHITECTURE CRC74_struc OF CRC74 IS

COMPONENT ..... IS
      ..... ... .....
      .....
      .....
END COMPONENT;

COMPONENT ..... IS
      .....
      .....
      .....
END COMPONENT;

SIGNAL ..... .....
SIGNAL ..... .....
SIGNAL ..... .....

BEGIN
    .....
    .....
    .....
    .....
    .....
    .....
END CRC74_struc;
```

Punkte
40

Aufgabe 2 Einfacher Moore-Automat zur Erkennung einer Bit-Sequenz

Es wird ein Schaltwerk entworfen, mit dem bei einer binäre Eingangsfolge $X(t)$ die Sequenz 110 detektiert wird. Am Ausgang des Moore-Automaten soll dies durch eine $\underline{Y} = [1]$ angezeigt werden. Ansonsten soll der Ausgang $\underline{Y} = [0]$ anzeigen. In der Abbildung unten ist das Zustandsdiagramm des Schaltwerks dargestellt.



Hinweis: In den Knoten werden der Zustand und die dazugehörige Ausgabe dargestellt (S_n/Y_n). Auf den Kanten wird der jeweilige Zustandsübergang dargestellt $(\underline{X})$.

a) Erstellen Sie die Zustandstabelle und die Kodierung der Zustände (10 Pkt.).

b) Entwickeln Sie eine Realisierung mit MS JK-FF (10 Pkt.).
Hinweis: Zustandsfolgetabelle des JK-FF

$(i)_{10}$	C	J	K	Q	Q^+
0	$0 \rightarrow 1$	0	X	0	0
1	$0 \rightarrow 1$	1	X	0	1
2	$0 \rightarrow 1$	X	1	1	0
3	$0 \rightarrow 1$	X	0	1	1

c) Beschreiben Sie den Automaten in VHDL als verhaltenssteuernden (Zustandsdiagramm) Drei-Prozess-Entwurf. Nutzen Sie bitte den Lückentext. Alle Zustandsänderungen und Signalwechsel erfolgen nach 20 ns! Achten Sie darauf, dass die Signale RESET und CLK inkludiert sind (10 Pkt.).

d) Der Eingang des Automaten wird mit dem Vektor $\underline{X} = [011010111000]$ beaufschlagt. Vervollständigen Sie das Impulsdiagramm (10 Pkt).

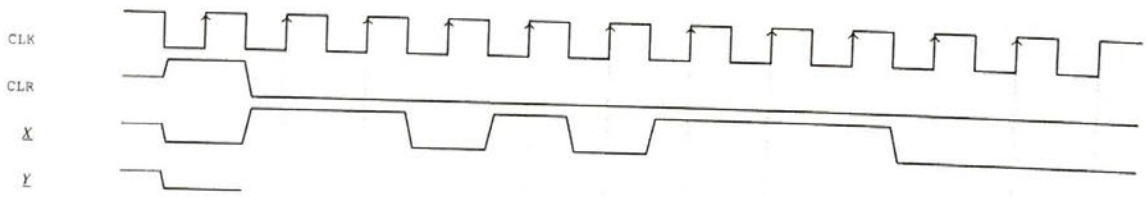


Abbildung 1: Impulsdiagramm der Bit-Sequenzerkennung.

Kodierung der Zustände / Erstellen der Ansteuerungstabelle für die JK-FF:

	Eingang	Zustand $\underline{S}$		Zustand $\underline{S}^+$		JK-FF 1		JK-FF 0		Ausgang
i_{10}	$\underline{X}$	Q_1	Q_0	Q_1^+	Q_0^+	J_1	K_1	J_0	K_0	$\underline{Y}$
0	0	S_0 0	0	0	0	0	X	0	X	0
1	1	S_0 0	0	0	1	0	X	1	X	0
2	0	S_1 0	1	0	0	0	X	X	1	0
3	1	S_1 0	1	1	0	1	X	X	1	0
4	0	S_2 1	0	1	1	X	0	1	X	0
5	1	S_2 1	0	1	0	X	0	0	X	0
6	0	S_3 1	1	0	0	X	1	X	1	1
7	1	S_3 1	1	0	1	X	1	X	0	1

$\underline{Y} = g(\underline{X}, \underline{S})$:

i_{10}	Q_1	Q_0	$\underline{Y}$
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

Übergangsfunktionen $f(\underline{X}, \underline{S})$: Dazu werden spaltenweise die Karnaugh-Diagramme für J_1, K_1, J_0 und K_0 aufgestellt.

$J_1 = f(Q_1, Q_0, X)$:

Q_0 \ Q_1, X	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$K_1 = f(Q_1, Q_0, X)$:

Q_0 \ Q_1, X	00	01	11	10
0	X	X	0	0
1	X	X	1	1

$J_0 = f(Q_1, Q_0, X)$:

Q_0 \ Q_1, X	00	01	11	10
0	0	1	0	1
1	X	X	X	X

$K_0 = f(Q_1, Q_0, X)$:

Q_0 \ Q_1, X	00	01	11	10
0	X	X	X	X
1	1	1	0	1

$J_1 = X \wedge Q_0$ ✓

$K_1 = Q_0$ ✓

$J_0 = (X \wedge \neg Q_1) \vee (\neg X \wedge Q_1)$ ✓ = ?

$K_0 = \neg Q_1 \vee (\neg X \wedge Q_1)$ ⇒ nicht minimiert

- Lückentext für den Zustandsautomaten

```
1  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
   USE ieee.std_logic_unsigned.all;
5
   ENTITY Moore_110 IS
7  PORT ( CLK : IN STD_LOGIC;          Eingang Datentyp Standardlogik
          RST : IN STD_LOGIC;          Eingang Datentyp Standardlogik
9         X : IN STD_LOGIC;            Eingang Datentyp Standardlogik
          Y : OUT STD_LOGIC;           Ausgang Datentyp Standardlogik
11 END Moore_110;
13                          Verhaltensmodell
15 ARCHITECTURE SEQUENCE OF Moore_110 IS
   TYPE ZUSTAENDE IS (S0,S1,S2,S3);    Aufzaehlungstyp
17 SIGNAL ZUSTAND, FOLGE_S : ZUSTAENDE;  Kommunikation
19                          Zustandsaktualisierung
   BEGIN
21 S_SPEICHER: PROCESS (CLK, RST)
       BEGIN
23       IF RST = '
         .....
25       .....
         .....
27     END PROCESS S_SPEICHER;
29                          Folgezustandsberechnung
   UE_SN: PROCESS (X, ZUSTAND)
31     BEGIN  Y <= '0' ; FOLGE_S <=S0;
           CASE state is
33           WHEN S0 => IF X = '1' then FOLGE_S <= S1 AFTER 20ns;
                        ELSE FOLGE_S <= S0 AFTER 20ns; END IF;
35
             WHEN S1 => IF X = '1' then FOLGE_S <= S2 AFTER 20ns;
37                      ELSE FOLGE_S <= S0 AFTER 20ns; END IF;
             WHEN S2 => IF X = '0' then FOLGE_S <= S3 AFTER 20ns;
39                      ELSE FOLGE_S <= S2 AFTER 20ns; END IF;
             WHEN S3 => Y <= '1';
41                      IF X = '1' then FOLGE_S <= S1 AFTER 20ns;
                        ELSE FOLGE_S <= S0 AFTER 20ns; END IF;
43
         END CASE;
45 END PROCESS UE_SN;
47                          Ausgangssberechnung
   AS_SN: PROCESS (ZUSTAND)
49
51
       BEGIN
53       CASE ZUSTAND IS
55       WHEN S3 => Y <= '1' AFTER 20ns;
         WHEN OTHERS => Y <= '0' AFTER 20ns;
57     END CASE;
   END PROCESS AS_SN;
59 END SEQUENCE;
```

Aufgabe 3 Realisierung eines vordefinierten Impulsdiagramms

Punkte: 40

In Abbildung 2 ist das Impulsdiagramm einer Steuereinheit für einen Laser Pointer dargestellt. Wird der Taster x gedrückt, soll der Laser für drei Taktperioden aufleuchten, unabhängig von der Dauer der Betätigung des Tasters. Wird der Taster dauerhaft gedrückt, soll der Laser weiter aufleuchten. Wird der Taster losgelassen, bleibt der Laser für drei Taktperioden noch aktiv.

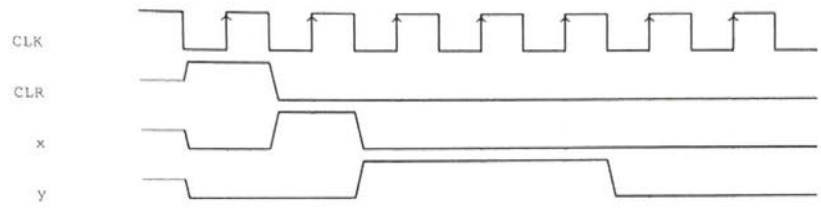


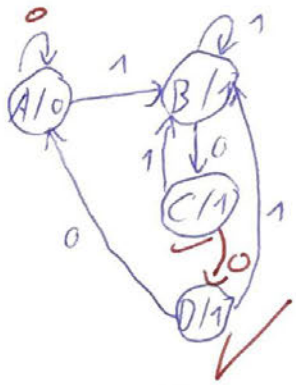
Abbildung 2: Impulsdiagramm einer Steuereinheit.

Aufgabenstellung:

a) Entwickeln Sie den Zustandsgraphen für die Realisierung als Moore-Automaten (5 Pkt.). Füllen Sie die Zustandstabelle aus (5 Pkt.).

A = 00
B = 01
C = 10
D = 11

i_{10}	Eingang x	Zustand $\underline{S}$	Zustand $\underline{S}^+$	Ausgang y
0	0	A = 0 0	0 0 = A	0
1	1	A = 0 0	0 1 = B	0
2	0	B = 0 1	1 0 = C	1
3	1	B = 0 1	0 1 = B	1
4	0	C = 1 0	1 1 = D	1
5	1	C = 1 0	0 1 = B	1
6	0	D = 1 1	0 0 = A	1
7	1	D = 1 1	0 1 = B	1



b) Entwickeln Sie eine Schaltung und geben Sie diese auf Gatter-Ebene an. Zur Realisierung verwenden Sie MS D-FF (10 Pkt.).

Tragen Sie die Übergangsfunktionen und die Ausgangsfunktion in die Tabelle ein:

Übergangsfunktionen
$D_1 =$
$D_0 =$
Ausgangsfunktion
$y =$

c) Entwickeln Sie eine strukturbasierte (Blockschaltbild, Aufgabenteil b) VHDL-Beschreibung zur Realisierung des Automaten. Der Automat ist nach der von Hufmann Normalform als Zwei-Prozess-Automat umzusetzen. Achten Sie darauf, dass die Signale RESET und CLK inkludiert sind. Nutzen Sie den gegebenen Lückentext (10 Pkt.).

Nutzen Sie die gegebenen Tabellen und Karnaugh-Diagramme zur Löseung der Aufgabe.

	Eingang	**Zustand** $\underline{S}$		**Zustand** $\underline{S}^+$		**Ausgang**
i_{10}	x	Q_1	Q_0	Q_1^+	Q_0^+	y
0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	0	0	1	1	0	1
3	1	0	1	0	1	1
4	0	1	0	1	1	1
5	1	1	0	0	1	1
6	0	1	1	0	0	1
7	1	1	1	0	1	1

Erstellen der Ansteuerungstabelle für die D-FF:

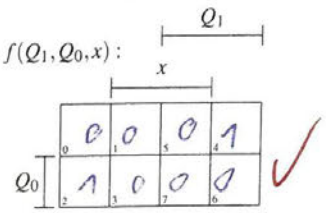
	Eingang	**Zustand** $\underline{S}$		**Zustand** $\underline{S}^+$		D-FF 1	D-FF 0	**Ausgang**
i_{10}	x	Q_1	Q_0	Q_1^+	Q_0^+	D_1	D_0	y
0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1	0
2	0	0	1	1	0	1	0	1
3	1	0	1	0	1	0	1	1
4	0	1	0	1	1	1	1	1
5	1	1	0	0	1	0	1	1
6	0	1	1	0	0	0	0	1
7	1	1	1	0	1	0	1	1

$y = g(x, \underline{S})$:

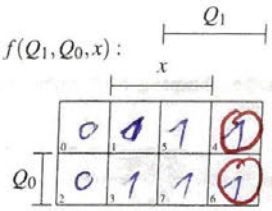
i_{10}	Q_1	Q_0	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

Übergangsfunktionen $f(\underline{X}, \underline{S})$: Dazu werden spaltenweise die Karnaugh-Diagramme für D_1 und D_0 aufgestellt.

$D_1 = f(Q_1, Q_0, x)$:



$D_0 = f(Q_1, Q_0, x)$:



- Lückentext für den Speicher

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY PIPO2Bit is
    port(
        ....................................
        ....................................
        ....................... ..............
        );
end PIPO2Bit;

ARCHITECTURE PIPO2Bit_arc OF PIPO2Bit IS
SIGNAL ..... .....
SIGNAL ... ..... .....
BEGIN
    PIPO2Bit: PROCESS ... ..... . IS
    BEGIN
        ........
        ........
        ..... ...
        ........
    END PROCESS PIPO2Bit;
    ..........
END PIPO2Bit_arc;
```

- Lückentext für den Automaten

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY Moore_LPT_2P is
PORT  ( ..........                    Eingang Datentyp Standardlogik
        ..........                    Eingang Datentyp Standardlogik
        ..........                    Eingang Datentyp Standardlogik
        ..........);                  Ausgang Datentyp Standardlogik
END Moore_LPT_2P;

ARCHITECTURE STRUCTURE OF Moore_LPT_2P IS

COMPONENT ..... IS
    PORT(.....
         .....
         .....);
END COMPONENT;

SIGNAL    .....
SIGNAL    .....

BEGIN
  PROCESS(.....)
  BEGIN
       CASE ..... IS
            WHEN .....
                 .....
                 .....
                 .....
                 .....
            WHEN .....
                 .....
                 .....
                 .....
            WHEN .....
                 .....
                 .....
                 .....
            WHEN .....
                 .....
                 .....
                 .....
            WHEN OTHERS => .....
                 .....
                 .....
                 .....
    END CASE;
END PROCESS;

 u0: ..........................................

END STRUCTURE;
```

① a)

$G(u) = u^3 + 1$ 1001

1: 0001000 : 1001
⊕ 1001
0001

2: 0010000 : 1001
⊕ 1001
00010

3: 0011000 : 1001
⊕ 1001
00110
⊕ 1001
0011

4: 0100000 : 1001
⊕ 1001
00100

5: 0101000 : 1001
⊕ 1001
00110
⊕ 1001
0101

6: 0110000 : 1001
⊕ 1001
01010
⊕ 1001
00110

7: 0111000 : 1001
⊕ 1001
01110
⊕ 1001
01110
⊕ 1001
0111

8: 1000000 : 1001
⊕ 1001
0001000
⊕ 1001
0001

9: 1001000 : 1001
⊕ 1001
0000

10: 1010000 : 1001
⊕ 1001
00110
⊕ 1001
01010
⊕ 1001
0011

[illegible]

11: 1011000 : 1001
⊕ 1001
0001000
⊕ 1001
00010

12: 1100000 : 1001
⊕ 1001
01010
⊕ 1001
00110
⊕ 1001
0101

13: 1101000 : 1001
⊕ 1001
01000
⊕ 1001
000100

15: 1111000 : 1001
⊕ 1001
01100
⊕ 1001
01010
⊕ 1001
00110

14: 1110000 : 1001
⊕ 1001
01110
⊕ 1001
01110
⊕ 1001
0111

b)

$y_2 = x_2$
$y_1 = x_1$
$y_0 = x_0 \leftrightarrow x_3$