13 VON 15 LEISTUNGSPUNKTEN

HAW Hamburg - Prüfungsklausur Computertechnik - SS 2014					
Aufgabe	1	2	3	4	Summe
Punkte	18	50	23	9	100
	15	45	20	8	88

d_n: Datenbit ($d_0 \hat{=} LSB$)

SB: Startbit

PB: Paritätsbit

1 Serielle Übertragung

Vier ASCII Zeichen (7 Bit) werden asynchron im RS232-Standard (Frames mit 7 Bit Nutzdaten) übertragen. Benennen Sie sämtliche Bits in Bild 1 mit Kurznamen und logischen Werten.

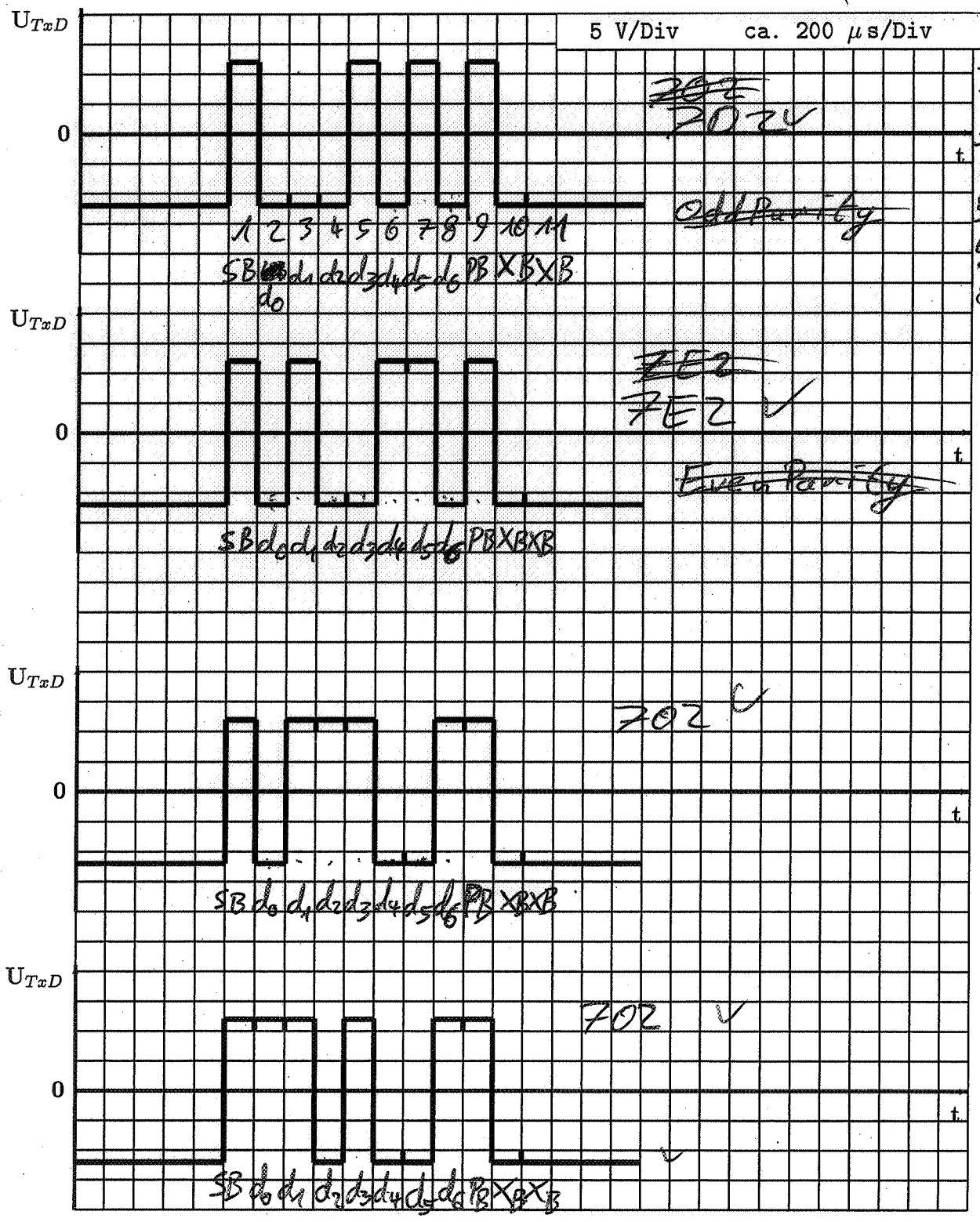


Bild 1. Signale der Zeichen am Anschluss TxD der seriellen Leitungen.

a) Welches der drei Protokolle 7N1, 701 oder 7E1 wird benutzt ?

7O2
Allerdings im 2. Frame 7E2

1 v. 1

b) Welche vier Zeichen werden übertragen? Geben Sie die Daten binär, hexadezimal und als ASCII-Zeichen an.

1. Frame (binär/hex/ASCII): ~~0101~~ 1010 111 / 0x57 / "W"
2. Frame (binär/hex/ASCII): 0100 1101 / 0x4D / M
3. Frame (binär/hex/ASCII): 0011 0001 / 0x31 / ~~1~~ 1
4. Frame (binär/hex/ASCII): 0011 0100 / 0x34 / 4

=> WM14

8 v. 8

c) Welche der folgenden Datenraten wird genutzt ? (Bitte ankreuzen)

☐ 110 bit/s	☐ 300 bit/s	☐ 1200 bit/s	☐ 2400 bits/s	☒ 4800 bits/s
☐ 9600 bit/s	☐ 19200 bit/s	☐ 38200 bit/s	☐ 57600 bit/s	☐ 115200 bit/s

1 v. 1

d) Welche Spannungsbereiche werden für die Wertedarstellung vor Line-Driver-Chip (RS232-Transceiver-Chip) beim RS232-Standard benutzt, welche nach diesem Chip ? Geben Sie auch die Namen der Bereiche an.

Vor: Logic Level (CMOS/TTL-Pegel)
- High (3,3V) } Bereiche - 1P
- Low (0V)

Nach: RS-232-Pegel
- Mark (-15V) } Bereiche - 1P.
- Space (+15V)

2

15 v. 18

	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0x0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x2...	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Tabelle 1: ASCII-Symbole (niedrigwertige Bits in den Spalten, höherwertige Bits in den Zeilen)

2 Gatterprüfmaschine

Es soll eine universelle Gatterprüfmaschine mit dem Controller LM3S9B92 erstellt werden. Sie kann Gatter der Typen AND, OR, NAND, NOR, XOR, XNOR prüfen. Die Schaltung mit Eingangs- und Ausgangspins ist in Bild 2 dargestellt.

Das Controllerprogramm erzeugt eine Kombination der Eingangswerte für das Testgatter für jeweils 10ms. Danach erfolgt Prüfung des Ausgangswerts. Dann wird die nächste Kombination der Eingangswerte angelegt, usw. ...
Welches der 6 möglichen Gatter aktuell angeschlossen ist, wird vom Programm erkannt.

Das Programm erkennt auch, wenn ein defektes (oder kein) Gatter angeschlossen ist, also wenn keine der o.g. logischen Funktionen erfüllt wird. Eine LED an Pin PJ6 zeigt diesen Fall an.

Beispiel: Die LED an PJ0 leuchtet, wenn das Testgatter bei in allen vier Eingangswert-Kombinationen als AND-Gatter korrekt funktioniert.

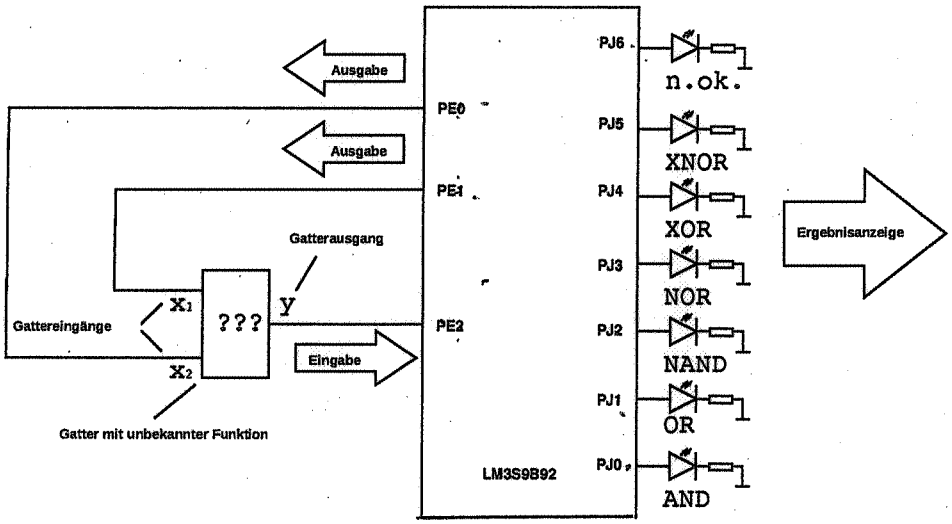


Bild 2: Übersicht über den zu erstellenden Paritätsgenerator

a) Vervollständigen Sie die folgende Tabelle der Gatterfunktionen.

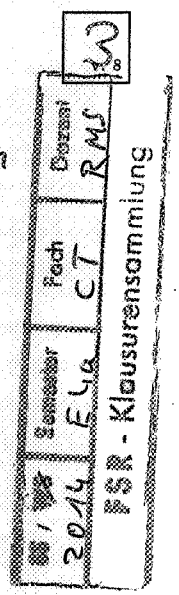
	Gattereingang		Gatterausgang y						
	x_1	x_2	AND	OR	NAND	NOR	XOR	XNOR	
0	0	0	0	0	1	1	0	1	1
1	0	1	0	1	1	0	1	0	2
2	1	0	0	1	1	0	1	0	4
3	1	1	1	1	0	0	0	1	8
			8	14	7	1	6	9	

Tabelle 2: Logische Funktionen, die geprüft werden

b) Programmieren Sie das Programm der Gatterprüfmaschine, zunächst die Funktion `main()`. Konfigurieren Sie die Ports E und J und einen General Purpose Timer für die 10ms Verzögerung. Eine Funktion `test_gate()` für den Prüfablauf und LED-Ausgabe können Sie schon benutzen.
Die Wartefunktion `warte_10ms()` soll in der Funktion `test_gate()` genutzt werden. Beide werden erst nachfolgend programmiert.
Der Ablauf der Gatterprüfung soll fortlaufend ohne Unterbrechung erfolgen. Kommentieren Sie ausführlich.

```
#define PART_LM3S9B92
#include "inc/lm3s9b92.h"
#define TIMERTIME 160000-1 // Ticks des Timers festlegen
void warte_10ms(void); // Prototype of the wait function for 10ms
void test_gate(void);  // Prototype of test function

void main(void)
{
 int wt=0;   // 16MHz Oszillator aktivieren
 SYSCTL_RCC_R =((SYSCTL_RCC_R|0x0540)&~0x02B3);
 wt++;
 SYSCTL_RCGC2_R|= 0x0110; // Takt für E & J zuschalten
 wt++;
 GPIO_PORTJ_DEN_R=0xFF; // PortJ 0...6 aktivieren  Digitale Fkt
 GPIO_PORTJ_DIR_R =0xFF; // PortJ.0...6 = Output
 GPIO_PORTE_DEN_R = 0x07; // PortE 0,1,2 dig.Fkt.aktivieren
 GPIO_PORTE_DIR_R =0x03; // PortE 0,1 = Output
 // Timer 3B initialisieren
 SYSCTL_RCGC1_R|= 0x080000; // Takt zuschalten
 wt++;
 TIMER3_CTL_R&= ~0x0100; // Timer stoppen
 TIMER3_CFG_R = 0x04;   // 16-Bit Modus
 TIMER3_TBPR_R=160-1;   // Prescaler setzen
 TIMER3_TBMR_R |= 0x02; // periodischen Modus einschalten
 TIMER3_TBMR_R&= ~0x10; // herunter zählen
 TIMER3_TBILR_R= TIMERTIME; // Startwert festlegen
 while(1){
  test_gate(); // Gatter prüfen
 }
```



c) Programmieren Sie eine Funktion `warte_10ms()`, die den Programmablauf 10ms verzögert. Nutzen Sie einen General Purpose Timer, den Sie am Beginn der Funktion starten und am Ende der Funktion stoppen. Der Timer ist in `main()` bereits konfiguriert. Kommentieren Sie ausführlich.

```
void warte_10ms(void)
{
 TIMER3_CTL_R|=0x0100; // Timer 3B starten
 while((TIMER3_RIS_R&0x0100)==0x00); // Warten bis Timer abgelaufen
 TIMER3_CTL_R&=~0x0100; // TIMER 3B stoppen
 TIMER3_ICR_R|= 0x0100; // Timeout flag löschen
```

d) Programmieren Sie die Funktion `test_gate()`, welche die Tests mit allen Eingangswertkombinationen durchführt, diese auswertet und die LED-Ausgaben vornimmt. Kommentieren Sie ausführlich.

```
void test_gate(void)
{
  int y[4]={0,0,0,0}; // Vektor für Ergebnis anlegen
  int out=0;          // auszugebende Kombinationen
  int lauf=0;         // Laufvariable
  int ergebnis=0;     // Endergebnis
  for(lauf=0; lauf<=3; lauf++) // nacheinander Kombinationen prüfen & Ergebnisvektor füllen
  {
    GPIO_PORTE_DATA_R=out; // 1. Lauf: out=0, 2. Lauf: out=1 ... out=3
    out++;
    warte_10ms();  // Verzögerung
    if((GPIO_PORTE_DATA_R&0x04)){ // wenn (maskiertes) Bit gesetzt
      y[lauf]=1;                  // Ergebnisvektor an dieser Stelle=1
    }else
      y[lauf]=0;                  // sonst=0
  }
  ergebnis=y[0]*1+y[1]*2+y[2]*4+y[3]*8; // Endergebnis berechnen
  switch(ergebnis){
    case 8:  // AND-Gatter
      GPIO_PORTJ_DATA_R=0x01; // LED an PJ0 an
      break;
    case 14: // OR-Gatter
      GPIO_PORTJ_DATA_R=0x02; // LED an PJ1 an
      break;
    case 7:  // NAND
      GPIO_PORTJ_DATA_R=0x04; // LED an PJ2 an
      break;
    case 1:  // NOR
      GPIO_PORTJ_DATA_R=0x08; // LED an PJ3 an
      break;
    case 6:  // XOR
      GPIO_PORTJ_DATA_R=0x10; // LED an PJ4 an
      break;
    case 9:  // XNOR
      GPIO_PORTJ_DATA_R=0x20; // LED an PJ5 an
      break;
    default: // Fehler / nok.
      GPIO_PORTJ_DATA_R=0x40; // LED an PJ6 an
      break;
  }
}
```

34 v. 34

45 v. 50

3 Adressen

a) Die Speicherbelegung eines ARM-Cortex-M3-Controller ist in Bild 3 dargestellt. Ergänzen Sie dort die fehlenden Adressen.

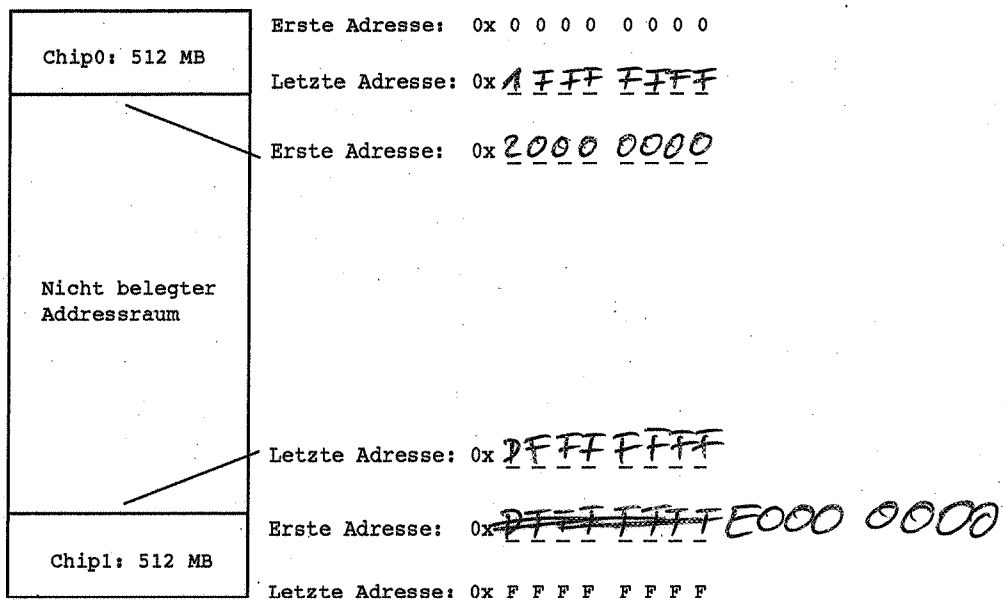


Bild 3: Speicherbelegung des Adressraums mit zwei Speicherchips

b) Wieviele Adressleitungen haben die beiden Speicherchips ? 32

c) Welche der folgenden Schaltungen sind geeignete Adressdecoder für den Chip 0 ? Streichen Sie alle ungeeigneten Schaltungen in Bild 4 durch.

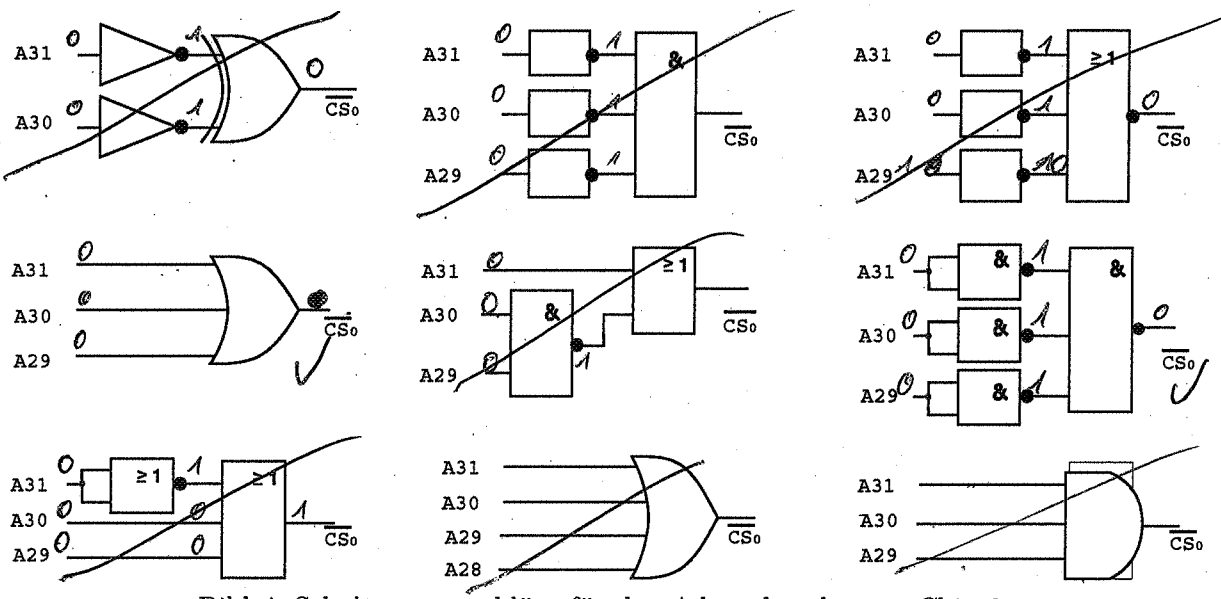


Bild 4: Schaltungsvorschläge für den Adressdecoder von Chip 0

chip0: 1. Nibble:
0x0000 => 0b0000
0x1FFF => 0b0001

d) Welche der folgenden Schaltungen sind geeignete Adressdecoder für den Chip 1 ? Streichen Sie alle ungeeigneten Schaltungen in Bild 5 durch.

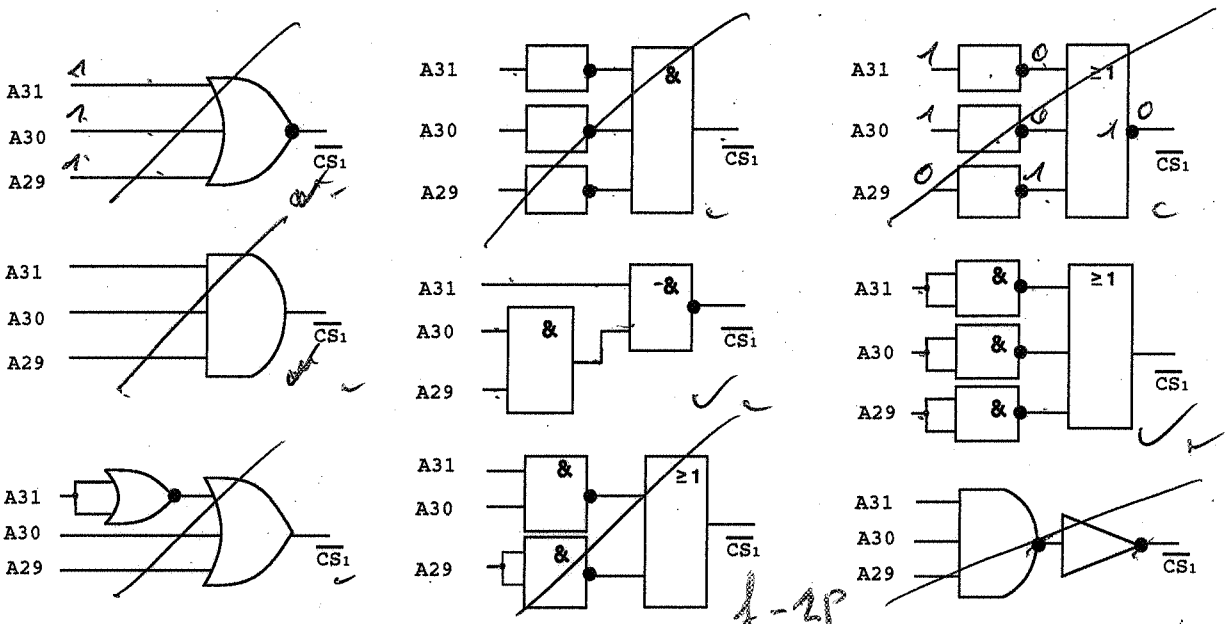


Bild 5: Schaltungsvorschläge für den Adressdecoder von Chip 1

Chip 1:
0xE000... 0b111X
0xFFFF... 0b111X

4 Fragen:

a) Erklären Sie den Unterschied zwischen einer Interruptquelle und einem Interrupthandler.

(Interrupt)Quelle ruft Interrupthandler auf, Quelle kann z.B. ein Pin des Controllers oder eine abgeschlossene AD-Wandlung des (internen) ADC sein. Der Interrupthandler "behandelt" die Interruptursache und ist (entfernt) vergleichbar mit einer Funktion in C.

b) Warum kann ein Interrupthandler - der in C wie eine Funktion geschrieben wird - grundsätzlich keine Parameter haben oder Rückgabewerte zurückgeben ?

Weil unklar ist, wann der Handler aufgerufen wird. Bei Aufruf des Interrupthandlers wird der ursprüngliche Programmablauf unterbrochen. Außerdem kann auch der Handler von einer nochrangigeren Quelle unterbrochen werden. (Race-Conditions)

c) Nennen Sie mindestens 5 Funktionseinheiten der CPU. Ordnen Sie diese den beiden übergeordneten Hauptteilen der CPU zu.

Control Unit:	Data Processor:
- Instruction Decoder	- ALU
- Instruction Control Sequencer	- General Purpose Register set
- Program Counter	- status Register
	- Temporary Buffer