Klausur zur Vorlesung Digitaltechnik

Name, Vorname: _____ Matrikelnummer: ⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋

Aufgabe 1 Verständnisfragen (10 Punkte)(5 × 2)

Beantworten die folgenden Fragen mit *wenigen kurzen Sätzen*.

a) Was ist der (obere und untere) *Störabstand* bei der Darstellung von Binärwerten als Spannungsbereiche und wofür wird dieser benötigt?

b) Wodurch unterscheiden sich 'X' und 'Z' bei der Vierwertigen Logik?

c) Wie entstehen *Glitches* in kombinatorischen Schaltungen und wann stellen diese in synchronen sequentiellen Schaltungen kein Problem dar?

d) Wodurch unterscheiden sich *Flip-Flops und Latches*?

e) Wie unterscheiden sich Shifter mit konstanter Shift-Weite von *Barrel-Shiftern* bezüglich ihrer kombinatorischen Verzögerungszeit?

Klausur zur Vorlesung Digitaltechnik

Name, Vorname: _____ Matrikelnummer: ⌴⌴⌴⌴⌴⌴⌴

Aufgabe 2 Zahlendarstellungen und binäre Subtraktion (10 Punkte)(5 + 5)

a) Vervollständigen Sie die folgende Tabelle *vorzeichenloser* Zahlendarstellungen. Alle Einträge einer Zeile sollen dabei den gleichen numerischen Wert repräsentieren. Verwenden Sie *möglichst kurze* Ziffernfolgen (ohne führende Nullen).

Dezimal	Binär	Hexadezimal
181_{10}		
	$10\ 1001_2$	
		$7C0_{16}$

b) Wandeln Sie $a = 60_{10}$ und $b = -15_{10}$ in 1 Byte breite *Zweierkomplement-Zahlen* um. *Subtrahieren* Sie die Binärdarstellungen voneinander $(a - b)$. Wandeln Sie das Ergebnis ins *Dezimalformat und* ins 12 bit *Hexadezimalformat* um. Der *Lösungsweg* wird bewertet.

Klausur zur Vorlesung Digitaltechnik

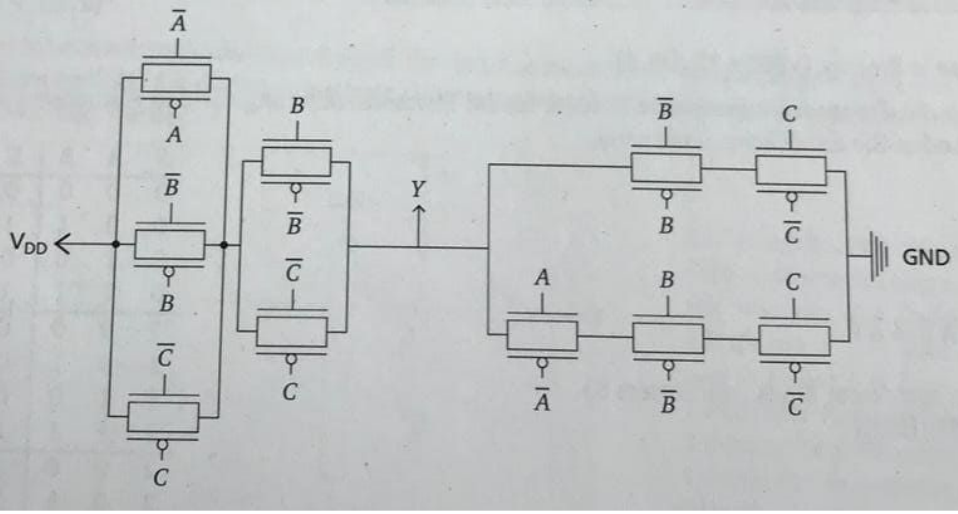
Name, Vorname: _____ Matrikelnummer: ⌊_⌊_⌊_⌊_⌊_⌊_⌊_⌊_⌊

Aufgabe 3 Realisierung von logischen Funktionen (15 Punkte)(8 + 7)

Die folgenden Teilaufgaben hängen *nicht* voneinander ab.

a) Zeichnen Sie eine *CMOS* Schaltung zur Realisierung von $Y = \overline{A+B}\ C$. Verwenden Sie ausschließlich *positive Eingangsliterale*.

b) Geben Sie die von folgender Schaltung realisierte boole'sche Funktion in *konjunktiver Normalform* an.



Klausur zur Vorlesung Digitaltechnik

Name, Vorname: _____ Matrikelnummer: ⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋

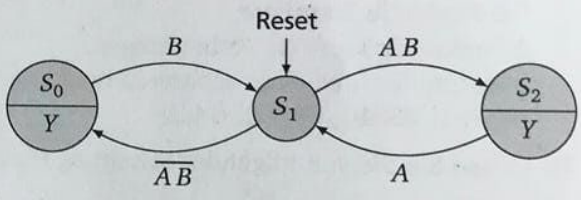
Aufgabe 4 Endliche Automaten (35 Punkte)(6 + 5 + 3 + 7 + 6 + 8)

a) Zeichnen Sie das Diagramm eines *Mealy-Automaten* mit zwei Eingängen A und B und einem Ausgang Y, welcher genau dann $Y = 1$ ausgibt, wenn B der *geraden Parität* der seit dem Reset gelesenen Bitfolge von A entspricht. Das aktuell gelesene A zählt bereits zu dieser Bitfolge hinzu. Verwenden Sie möglichst wenige Zustände.

$A\,B$

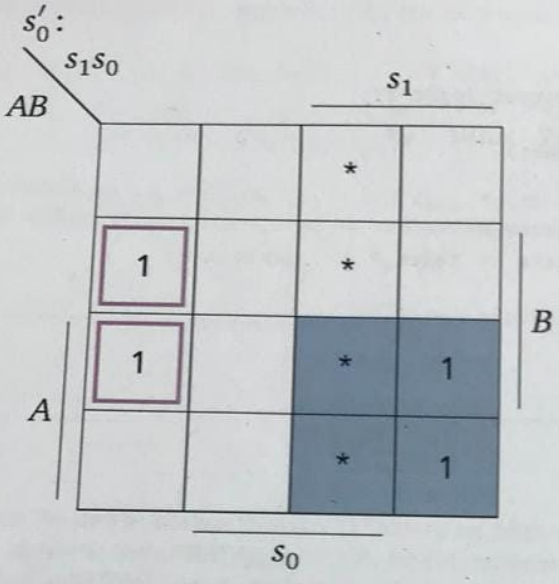
b) Vervollständigen Sie die folgende Tabelle so, dass sie das taktweise Ein-/Ausgabeverhalten des nebenstehenden Automaten mit *asynchronem Reset* wiedergibt.

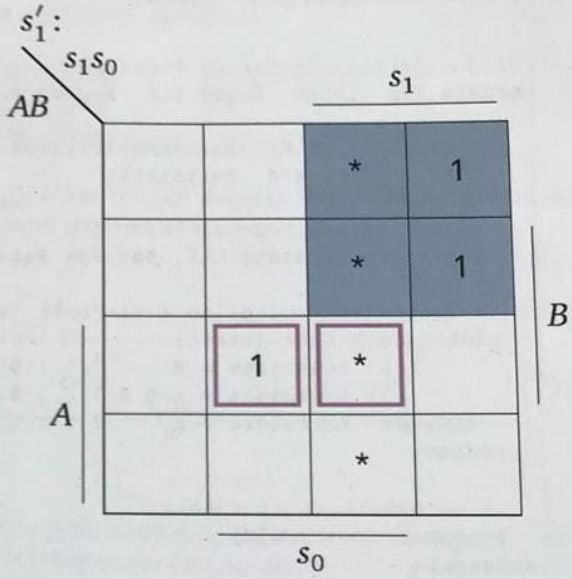
Reset	1	1	0	0	0	0	0	0		0
A	1	0	0	0	0	1		0	0	0
B	1	1	1	0		1	1	0	1	0
Y						0		1	0	



c) Geben Sie die *Zustandsübergangstabelle* ohne binäre Zustandskodierung für den Automaten aus Aufgabenteil b) an. Verwenden Sie dabei *keine don't cares*.

d) Geben Sie die Zustandsübergangsfunktionen für den Automaten aus Aufgabenteil b) mit kanonischer Zustandskodierung ($S_0 = 00, S_1 = 01, S_2 = 10$) an. Verwenden Sie dafür die folgenden Karnaugh-Diagramme.





e) Zeichnen Sie das *Schaltwerk inklusive Ausgabelogik* für den Automaten aus Aufgabenteil b) mit der Zustandskodierung aus Aufgabenteil d). Sie dürfen invertierte Gattereingänge (Inverterblasen) verwenden. Die Reset-Logik muss nicht berücksichtigt werden.

f) Implementieren Sie den Automaten aus Teilaufgabe b) als *SystemVerilog* Modul namens `fsm` inklusive *Modulsch*
stelle. Kommentieren Sie Ihre Lösung.

Name, Vorname: _____ Matrikelnummer: ⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋⌊⌋

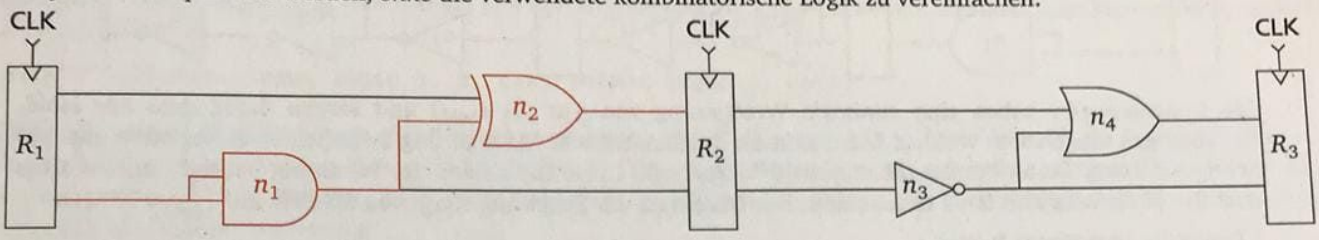
Aufgabe 5 Timing Analyse

(10 Punkte)(3 + 3 + 4)

Für diese Aufgabe werden *ausschließlich* die wie folgt spezifizierten Logikgatter verwendet:

	XOR	AND	OR	NOT
t_{cd}	8 ns	4 ns	3 ns	2 ns
t_{pd}	10 ns	7 ns	5 ns	2 ns

Darüber hinaus sei $t_{ccq} = 0{,}5\,\text{ns}$, $t_{pcq} = 1\,\text{ns}$, $t_{setup} = 2\,\text{ns}$ und $t_{hold} = 3\,\text{ns}$ für alle Register. Folgendes Schaltwerk soll analysiert und optimiert werden, *ohne* die verwendete kombinatorische Logik zu vereinfachen:



a) Geben Sie den *kritischen Pfad* der Schaltung an. Mit welcher Frequenz kann das Schaltwerk *maximal getaktet* werden, ohne die *Setup-Bedingung* von R_2 und R_3 zu verletzen? Begründen Sie Ihre Antwort.

b) Wird die *Hold-Bedingung* von R_2 und R_3 eingehalten? Begründen Sie Ihre Antwort.

Name, Vorname: _____ Matrikelnummer: ⌴⌴⌴⌴⌴⌴⌴

Aufgabe 6 Sequentielle Addierer

(10 Punkte)(10)

Ein *sequentieller Addierer* berechnet die Summe zweier Zahlen mit einer sequentiellen statt einer kombinatorischen Schaltung, wobei in jedem Takt ein Ergebnis-Bit generiert wird. Dazu sind neben den arithmetischen Ein- und Ausgaben weitere Steuersignale notwendig:

- `CLK` - das Taktsignal
- `START` - wird für einen Takt auf 1 gesetzt, wenn eine neue Berechnung starten soll
- `DONE` - wird für einen Takt auf 1 gesetzt, wenn eine Berechnung fertig ist.

Ein solcher Addierer benötigt unabhängig von der Bitbreite der Eingänge nur einen einzigen Volladdierer mit folgender Modulschnittstelle:

```
module fulladder(input logic a, b, cin, output logic s, cout);
```

Der Volladdierer wird taktweise mit den entsprechenden Eingabebits beschaltet. Dazu können die Eingaben bspw. in Registern gespeichert werden, deren Inhalte in jedem Takt um ein Bit nach rechts geschoben werden. Die Summen-Bits können auch in einem Shift-Register gesammelt werden. Für das Übertragen des carry-Bits in den nächsten Takt ist ebenfalls ein Register notwendig.

Implementieren Sie einen sequentiellen Addierer mit folgender Schnittstelle in *SystemVerilog*. *Kommentieren* Sie Ihre Lösung. Beachten Sie, dass die *Summe ein Bit breiter* als die Summanden sind.

```
module seqadd
 #(parameter    WIDTH = 8)        // Bitbreite der Eingaben
  (input  logic CLK,              // Takt
                START,            // Eingaben liegen an
   input  logic [WIDTH-1:0] A, B, // zu addierende Eingaben
   output logic [WIDTH:0]   S,    // Summe der Eingaben
   output logic DONE);            // Berechnung fertig
```