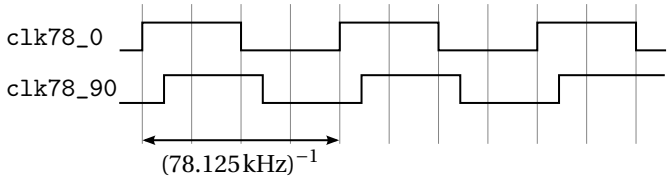
Digitale Schaltungstechnik, WS 2022/2023

0. Übungsblatt — Abgabe: keine

1 Funkuhr 1

Die Empfängerschaltung auf der Erweiterungsplatine liefert ein Ausgangssignal, dessen Wert der Amplitude des 77.5 kHz-Funksignals entspricht. Dieses Signal wird über den Pin `A8` an den FPGA zurückgeführt, um es verwertbar zu machen.

Damit die Empfängerschaltung funktioniert, benötigt sie neben dem durch die Antenne empfangenen Funksignal noch zwei Rechtecksignale mit der Frequenz 78.125 kHz, die um 45° (d. h. eine Achtelperiode) gegeneinander phasenverschoben sind. Das „frühere“ der beiden Rechtecksignale muss am Pin `B4` anliegen, das „spätere“ am Pin `A4`:



Wenn Sie die folgenden Zeilen in eine UCF-Datei einfügen, bezeichnen also `clk78_0` und `clk78_90` die beiden Rechtecksignale (`clk78_90` muss gegenüber `clk78_0` verzögert sein) und `dcf_digital` das von der Empfängerschaltung erzeugte Signal:

```
NET "clk78_0"     LOC = "B4";
NET "clk78_90"    LOC = "A4";
NET "dcf_digital" LOC = "A8";
```

a) Schreiben Sie ein Verilog-Modul, das mithilfe des 50-MHz-Takts der Nexys2-Platine die beiden 78.125-kHz-Rechtecksignale erzeugt.

b) Schließen Sie die Antenne an die Erweiterungsplatine an (die natürlich an das Nexys2 angeschlossen sein sollte). Geben Sie `dcf_digital` auf einer LED aus und finden Sie eine Position der Antenne, bei der Sie den DCF-Sekundentakt deutlich beobachten können (der Empfang ist besser, wenn Sie die Antenne nicht berühren).

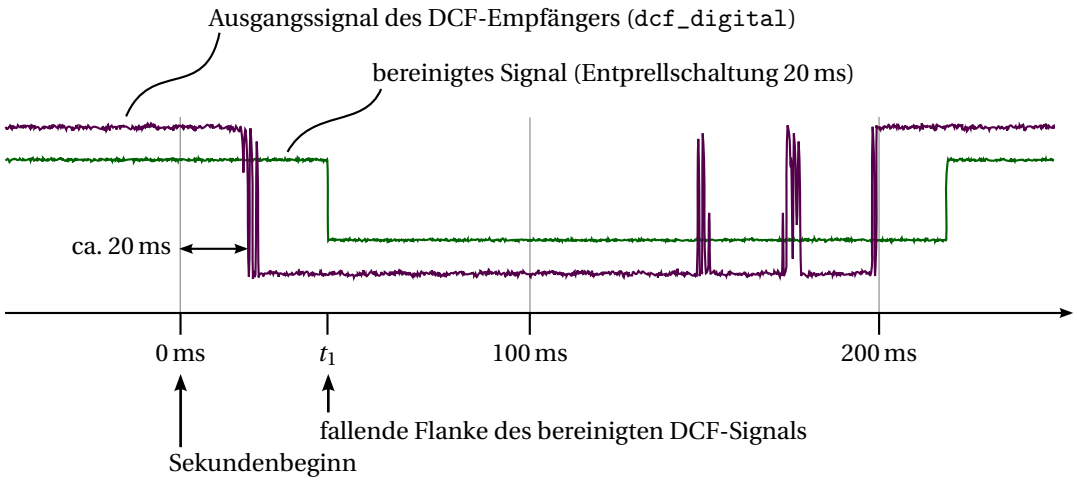


Abbildung 1: Mit dem Oszilloskop gemessener Verlauf des DCF-Signals während der Absenkung zum Sekundenbeginn (lila). Durch eine Entprellschaltung mit einer Zeitkonstante von 20 ms kann das Signal bereinigt werden (grün). Die tatsächliche Dauer der Absenkung des Signals `dcf_digital` ist immer ca. 20 ms zu kurz, d. h. sie beträgt 80 bzw. 180 ms. Gehen Sie davon aus, dass der *Beginn* der Absenkung 20 ms zu spät passiert, der wahre Beginn der Sekunde ist also 20 ms *vor* dem Beginn der Absenkung von `dcf_digital`.

2 Funkuhr 2

Als grundlegendes Element für die Uhr benötigen Sie einen stabilen Sekundentakt. Dies soll ein Signal `CE_1Hz` sein, das Sie in der folgenden Weise verwenden können, um z. B. einen Zähler genau zu Beginn jeder Sekunde (wie er durch `dcf_digital` definiert wird) zu erhöhen:

```
always @(posedge clk) begin
  if (CE_1Hz)
    counter <= counter + 1;
end
```

a) Das Signal `CE_1Hz` muss einmal pro Sekunde für die Dauer von einer Periode des 50-MHz-Takts `clk` den Wert „1“ haben und ansonsten den Wert „0“.

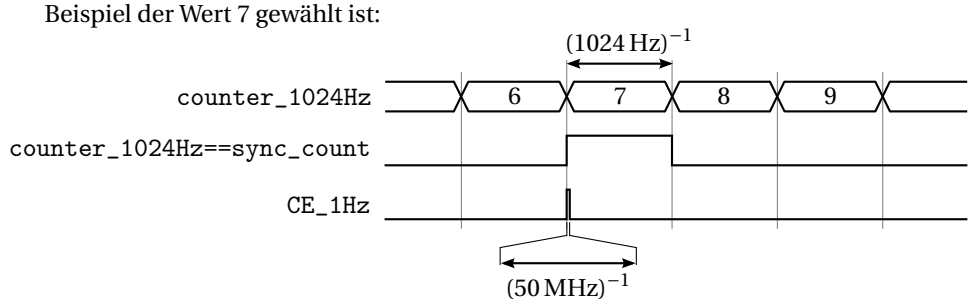
Man kann `CE_1Hz` *nicht* einfach aus der fallenden Flanke des DCF-Signals `dcf_digital` ableiten. Dafür gibt es mehrere Gründe:

- Erstens enthält `dcf_digital` meistens Störungen, wodurch `CE_1Hz` viel zu häufig ausgelöst würde (siehe Abbildung 1).
- Zweitens fehlt die Absenkung bei der letzten Sekunde jeder Minute, es würde also selbst unter idealen Empfangsbedingungen ein Sekundentakt pro Minute fehlen.
- Drittens würde man bei schlechtem Empfang, etwa durch eine ungünstige Position der Antenne, garkein Signal mehr erhalten, wodurch die Uhr komplett stehenbleiben würde.

Um `CE_1Hz` auf sinnvolle Weise zu erzeugen, können Sie wie folgt vorgehen:

- Legen Sie einen 10-Bit-Zähler an, der mit der Frequenz 1024 Hz betrieben wird. Dieser Zähler (im folgenden mit `counter_1024Hz` bezeichnet) durchläuft also einmal pro Sekunde alle möglichen Werte von 0 bis 1023.

- Erzeugen Sie ein Signal, das genau dann den Wert „1“ hat, wenn `counter_1024Hz` identisch zu einem bestimmten Wert `sync_count` ist. Das ergibt also einmal pro Sekunde einen Puls mit der Breite von ca. einer Millisekunde. Mit einem Flankendetektor können Sie nun daraus das gewünschte Signal `CE_1Hz` erzeugen. In der folgenden Abbildung ist dies gezeigt, wobei für `sync_count` als Beispiel der Wert 7 gewählt ist:



b) Das auf diese Weise erzeugte Signal `CE_1Hz` hat bereits den richtigen zeitlichen Verlauf, aber es ist gegenüber dem wahren Sekundentakt, der durch `dcf_digital` vorgegeben wird, um eine konstante Zeit verschoben. Außerdem wird Ihre Uhr so noch minimal zu schnell oder zu langsam laufen, da Sie die benötigten 1024 Hz nicht genau genug erreichen können.

Um den Sekundentakt `CE_1Hz` mit dem DCF-Signal zu synchronisieren, muss der Wert `sync_count` nachgeregelt werden, so dass `CE_1Hz` genau zum richtigen Zeitpunkt aktiv ist:

- Verwenden Sie ein Modul, das Sie schon zum Entprellen der Tasten benutzt haben, um das DCF-Signal zu bereinigen. Als Zeitkonstante sind hier 20 ms geeignet (siehe Abbildung 1). Erzeugen Sie mit einem Flankendetektor ein Signal, das die fallende Flanke dieses bereinigten DCF-Signals anzeigt.
- Der Zeitpunkt t_1 der fallenden Flanke des bereinigten DCF-Signals ist gegenüber dem wahren Beginn der Sekunde (t_0) um 20 ms aufgrund der Entprellschaltung und um weitere 20 ms durch den zu späten Beginn der DCF-Absenkung verzögert: $t_1 = t_0 + 40\,\text{ms}$

  Um `CE_1Hz` genau zum Beginn der wahren Sekunde auszulösen, muss also

  $$\texttt{counter\_1024Hz} = \texttt{sync\_count} + 1024\,\text{Hz} \cdot 40\,\text{ms} = \texttt{sync\_count} + 41$$

  gelten, wenn der Zählerstand von `counter_1024Hz` zum Zeitpunkt der fallenden Flanke des bereinigten DCF-Signals ausgewertet wird.

  Der Wert `sync_count` muss nun *schrittweise* so verringert oder erhöht werden, dass diese Bedingung erfüllt wird. Die Änderung darf aber nicht zu schnell erfolgen, damit die Uhr nicht sofort aus dem Takt gerät, sobald der Empfang des DCF-Signals gestört ist.

Der Verilog-Code, mit dem diese Regelung beschrieben wird, sollte in etwa so aussehen:

```
reg signed [9:0] count_delta; // korrekte Behandlung von negativen Werten
always @(posedge clk) begin
  if (negedge_dcf_clean) begin // fallende Flanke des bereinigten DCF-Signals
    count_delta <= sync_count + 41 - counter_1024Hz;
    if (count_delta < 0) begin // sync_count zu klein, CE_1Hz zu früh
      sync_count <= ...
    end
    else if (count_delta > 0) begin // sync_count zu groß, CE_1Hz zu spät
      sync_count <= ...
    end
  end
end
```

3 Funkuhr 3

Um die Dauer der Absenkung („Pulsbreite“) von `dcf_digital` zuverlässiger zu messen, ist es sinnvoll, das Signal nur in dem Zeitbereich zu betrachten, in dem man die Absenkung erwartet, d. h. ab dem `CE_1Hz`-Puls für eine Dauer von bis zu 200 ms.

a) Erzeugen Sie ein Signal `CE_sample_pw`, das den gleichen Verlauf wie `CE_1Hz` hat, aber um 250 ms nach hinten verschoben ist. Durch dieses Signal wird das Ende der Zeitmessung definiert.

   Dazu können Sie wieder den Zähler `counter_1024Hz` verwenden. Das neue Signal muss, 250 ms nachdem der Zähler den Wert `sync_counter` erreicht hat, einen kurzen Puls (einen 50-MHz-Takt lang) geben, was einer Differenz von 256 des Zählerstands entspricht.

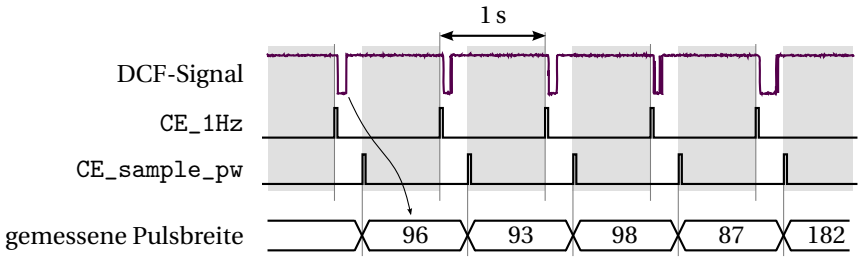
   Hinweis: Für den Fall, dass `sync_counter` weniger als 256 vom Maximalwert 1023 entfernt ist (also ≥ 768 ist), muss für die Ermittlung des Vergleichswerts für `CE_sample_pw` der Überlauf korrekt behandelt werden. Dies geschieht automatisch, wenn der Wert 256 als 10-stellige Binärzahl angegeben wird:

```
count_1024Hz == sync_count + 10'd256
```

   Fehlt die Angabe `10'd`, werden für die Addition beide Zahlenwerte auf eine Breite von 32 Bit vergrößert und das Ergebnis kann auch Werte über 1023 annehmen. Hat `sync_count` bspw. den Wert 800, würde `count_1024Hz` mit 1056 statt korrekterweise mit 32 verglichen.

b) Legen Sie einen Zähler `counter_pw` an, der immer zählt, wenn das DCF-Signal Null ist. Die Dauer der Absenkung erhalten Sie, wenn Sie diesen Zähler bei einem `CE_1Hz`-Puls zurücksetzen und seinen Stand bei einem `CE_sample_pw`-Puls in ein Register übernehmen.

   Der beschriebene Ablauf ist in der folgenden Grafik dargestellt (die grauen Flächen kennzeichnen die Zeitbereiche, in denen das DCF-Signal für die Messung der Pulsbreite ignoriert wird):



   Geben Sie die gemessenen Pulsbreiten auf der Siebensegmentanzeige aus und beobachten Sie, welche Werte Sie erhalten und wie sich die Werte ändern, wenn Sie die Antenne berühren oder in eine andere Position mit schlechterem Empfang bewegen.

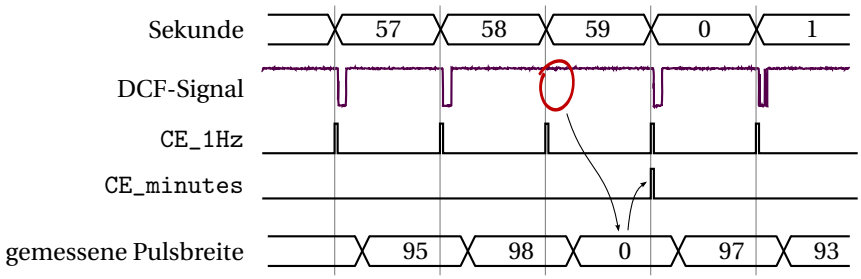
4 Funkuhr 4

In der letzten (der 59.) Sekunde jeder Minute fehlt die Absenkung des DCF-Signals, ihre Dauer ist also Null. In dieser Aufgabe entwerfen Sie nun eine Schaltung, die daraus den Minutenbeginn automatisch erkennt.

a) Versuchen Sie zunächst selbst, anhand der gemessenen Pulsbreiten, die Sie auf der Siebensegmentanzeige ablesen, den Beginn einer neuen Minute zu erkennen.

b) Erzeugen Sie mithilfe des Sekundentakts `CE_1Hz` und der gemessenen Pulsbreite die beiden folgenden Signale:

- Ein Register, das den aktuellen Wert der Sekunde enthält (also z. B. um 15:46:31 Uhr den Wert 31) und (evtl. daraus abgeleitet)
- ein Signal `CE_minutes`, das genau zu Beginn jeder Minute (d. h. zu Beginn der Sekunde 0) für eine Periode des 50-MHz-Taktsignals den Wert „1“ hat, und ansonsten den Wert „0“.



Nicht jede Sekunde, in der keine Absenkung des DCF-Signals gemessen wird, ist zwangsweise die letzte Sekunde einer Minute: Bei schlechtem Empfang kann es passieren, dass man ständig Pulsbreiten von Null erhält. Ein Kriterium, um den wahren Beginn einer Minute von diesem Zustand zu unterscheiden, ist, dass die Pulsbreite unmittelbar vor dem Auftreten einer Null 59-mal in Folge (nämlich von der 0. bis zur 58. Sekunde einer Minute) größer als Null gewesen sein muss.

c) Überlegen Sie sich einen Mechanismus, der verhindert, dass der Minutentakt durch schlechten Empfang des DCF-Signals oder sonstige ungünstige Umstände auf einen falschen Zeitpunkt gesetzt wird, nachdem er bereits korrekt eingestellt war.

Geben Sie den von Ihrer Schaltung erkannten Sekundenstand auf der Siebensegmentanzeige aus und vergleichen Sie ihn mit einer Uhr, von der Sie wissen, dass sie die Sekunden korrekt anzeigt (die Sekundenanzeige auf `http://www.aktuelle-uhrzeit.de` ist z. B. ziemlich genau).

5 Funkuhr 5

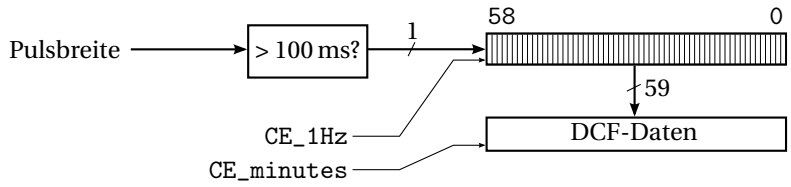
In jeder Sekunde wird ein Bit Information übertragen, indem die Dauer der DCF-Absenkung einen von zwei möglichen Werten annimmt; entweder 100 ms oder 200 ms (wie auf dem vorherigen Aufgabenblatt erwähnt, ist die tatsächliche Dauer der Absenkung des Signals, das von der Empfangsschaltung erzeugt wird, kürzer, nämlich entweder ca. 80 ms oder ca. 180 ms).

Da Sie in der letzten Aufgabe einen Minutentakt einsynchronisiert haben, können Sie nun die 59 Bits, die in jeder Minute übertragen werden, richtig interpretieren, um die aktuelle Uhrzeit zu ermitteln.

a) Legen Sie ein Schieberegister mit 59 Bits an. In jeder Sekunde (`CE_1Hz`) soll das aus der gemessenen Pulsbreite hervorgehende Bit eingeschoben werden („1“, falls die Absenkung länger als 100 ms war, sonst „0“).

Die Richtung des Schieberegisters ist so zu wählen, dass die zuerst empfangenen Bits weiter rechts stehen als die zuletzt empfangenen Bits. Auf diese Weise entspricht am Ende einer Minute das Bit an der Stelle `0` auch der 0. Sekunde der Minute und das Bit an der Stelle `58` der 58. Sekunde (in der 59. Sekunde wird ja kein Bit übermittelt).

Zum richtigen Zeitpunkt (`CE_minutes`) muss nun der Inhalt des Schieberegisters ausgelesen werden, um die Information über die Uhrzeit zu dekodieren.



b) In den 59 Bits, die in jeder Minute übertragen werden, ist das Datum und die Uhrzeit kodiert, die in der *darauf folgenden* Minute gelten. Die Bedeutung der einzelnen Bits ist in der folgenden Tabelle dargestellt. Als Beispiel sind die Daten angegeben, die am 12. Januar 2012 zwischen 10:39:21 und 10:39:58 empfangen worden sind. Die Bits, die von der 0. bis zur 20. Sekunde einer Minute gesendet werden, sind hier ohne Belang (siehe z. B. http://www.dcf77.de/kodierschema-des-dcf77-signals.html).

Position	Bedeutung	Beispiel:	Donnerstag, 12. Januar 2012, 10:40
58	Parität Datum	0	
57:54	Jahr (Zehnerstelle)	0001	1
53:50	Jahr (Einerstelle)	0010	2
49	Monat (Zehnerstelle)	0	0
48:45	Monat (Einerstelle)	0001	1
44:42	Wochentag	100	4
41:40	Tag (Zehnerstelle)	01	1
39:36	Tag (Einerstelle)	0010	2
35	Parität Stunden	1	
34:33	Stunden (Zehnerstelle)	01	1
32:29	Stunden (Einerstelle)	0000	0
28	Parität Minuten	1	
27:25	Minuten (Zehnerstelle)	100	4
24:21	Minuten (Einerstelle)	0000	0

Wie Sie sehen, werden alle Ziffern der Dezimaldarstellung des Datums und der Uhrzeit einzeln in Binärdarstellung übermittelt *(Binary Coded Decimal, BCD)*. Das hat den Vorteil, dass Sie das Register, in dem die Daten jede Minute gespeichert werden, direkt mit der Siebensegmentanzeige verbinden können. Der Nachteil ist, dass Sie die Zahlen in Binärdarstellung umwandeln müssen, wenn Sie sie in der Schaltung weiterverarbeiten wollen.

Außerdem werden drei Paritätsbits übermittelt, die jeweils angeben, ob die Anzahl der Einsen in der BCD-Darstellung des Datums, der Stunde und der Minute jeweils gerade oder ungerade ist. Dadurch ist es möglich, festzustellen, *dass bei der Übertragung ein Fehler aufgetreten ist*. Es ist *nicht* möglich, festzustellen, dass *kein* Fehler aufgetreten ist.

Um die Parität zu überprüfen, können Sie den XOR-Operator in der Schreibweise `^(a)` verwenden, bei der alle Bits des Vektors `a` miteinander verknüpft werden, so dass genau die Parität von `a` herauskommt.

Geben Sie die dekodierte Uhrzeit auf der Siebensegmentanzeige sowie das Ergebnis der Paritätsprüfung auf den LEDs aus. Wie zuverlässig wäre die Uhr in diesem Zustand?

c) Erstellen Sie nun aus den bisher vorhandenen Schaltungsteilen eine vollständige Uhr, die möglichst zuverlässig die richtige Uhrzeit anzeigt.

Dazu geben Sie die dekodierte Uhrzeit am besten nicht direkt aus, sondern legen eine interne Uhr an, deren Stand bei jedem `CE_minutes`-Puls erhöht wird. Die dekodierten DCF-Daten nutzen Sie dazu, die

interne Uhr bei Bedarf auf den richtigen Wert zu setzen. Überlegen Sie sich einen Mechanismus, mit dem Ihre Uhr erkennt, ob sie die empfangenen Daten akzeptieren oder ignorieren soll.

Um zwischen dem BCD-Format (für die Anzeige) und der normalen Binärdarstellung (für die interne Uhr) zu konvertieren, können Sie die beiden Module `binary2bcd` und `bcd2binary` verwenden:

- `http://sus.ziti.uni-heidelberg.de/Lehre/WS_DST/files/binary2bcd.v`
- `http://sus.ziti.uni-heidelberg.de/Lehre/WS_DST/files/bcd2binary.v`