FB Elektrotechnik und Informationstechnik
Prof. Dr.-Ing. Norbert Wehn
Dozent:
Uwe Wasenmüller
Raum 12-213, wa@eit.uni-kl.de

Task 4_B
Decoder for DCF-77 Radio Clock Receiver

1. Objective

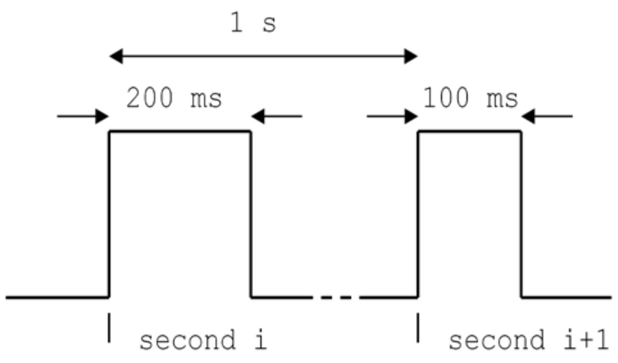
Objective of this task is, to realize a complex task of digital signal processing. You will describe a synthetizable VHDL model (RTL level). In this exercise you will model and verify by simulation a decoder of the radio clock signal (DCF77) in VHDL. The transmitted time information must be decoded and displayed in a specified order to an external display. In the next exercise you will synthesize this design and implement it on laboratory hardware.

The architecture of the design is just specified. There are five modules plus the top level design with given entity declarations. You will find the declarations in your account in directory V4und5_B_DCF77.

2. Basics

The DCF77-time-symbol-transmitter emits the timestamp in Binary Coded Decimal code (BCD-code). The code includes year, month, day, hour, minute and weekday. Each second one Bit of the encoded timestamp is transmitted. After every full minute the timestamp is complete. The timestamp bit encoding is done through impulse length variation of the second impulses. An impulse of 100 ms length is interpreted as logical 0 and an impulse of 200 ms length is interpreted as logical 1.

Example of impulse pattern of DCF77-time-signal:



To detect the start of a frame no impulse (neither 100 ms nor 200 ms) is transmitted in second 59. The complete time frame of DCF-77 is depicted on next page.

Time frame of DCF77 (see ELVjournal 6/94, page 27 and 28)

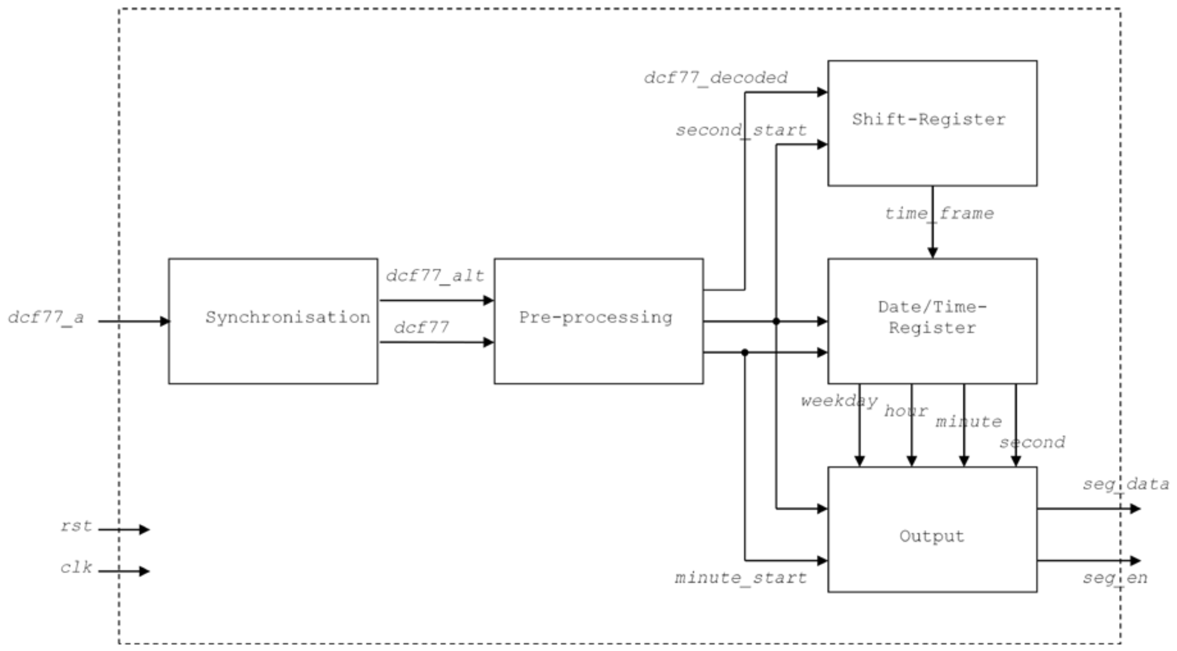
Bit	Meaning	Weight	Bit	Meaning	Weight
0	minute begin (Low = 0)	-	36	calendar day ones	1
1...14	not used	-	37	calendar day ones	2
15	reserve antenna	-	38	calendar day ones	3
16	clock change notice	-	39	calendar day ones	4
17	time zone bit 1	-	40	calendar day tens	1
18	time zone bit 2	-	41	calendar day tens	2
19	leap second notice	-	42	weekday	1
20	time frame begin	-	43	weekday	2
21	minute ones	1	44	weekday	4
22	minute ones	2	45	month ones	1
23	minute ones	4	46	month ones	2
24	minute ones	8	47	month ones	4
25	minute tens	1	48	month ones	8
26	minute tens	2	49	month tens	1
27	minute tens	4	50	year ones	1
28	parity bit 1	-	51	year ones	2
29	hour ones	1	52	year ones	4
30	hour ones	2	53	year ones	8
31	hour ones	4	54	year tens	1
32	hour ones	8	55	year tens	2
33	hour tens	1	56	year tens	4
34	hour tens	2	57	year tens	8
35	parity bit 2	-	58	parity bit 3	-

In second 59 there is no impulse transmitted!

Specifications

- It is a DCF77-receiver installed on the laboratory board. The received signal is conducted to the FPGA (signal *dcf77_a*). Analysis of this signal shall be implemented in the FPGA.
- The Laboratory board includes 8 seven digit displays, each with an upstream latch. One enable signal per display (signal *seg_en(7:0)*) is used to store a value (BCD number) in the latch. This value is displayed automatically. All 8 displays share a common data bus (signal *seg_data(0:7)*).
- There are not enough displays to show the full time information. Only the transmitted weekday, hour and minute information shall be shown. On top of that, a second display is to be realised using a counter.
- The laboratory board is driven by a 1 kHz clock. All modules will be clocked with this 1 kHz clock. The DCF77 time signal is asynchronous to the system clock and has to synchronized inside the FPGA.
- An asynchronous reset is available in the board. This reset must be used for all modules.

Partioning of the design.



3. Approach

You have to realize the five shown modules. For each module use the following approach

- Analysis of specification
- Analysis of interface
- Analysis of test stimuli
- Derive the expected output
- Implementation
- Verification by simulation using test stimuli
- Extension of given stimuli for better verification

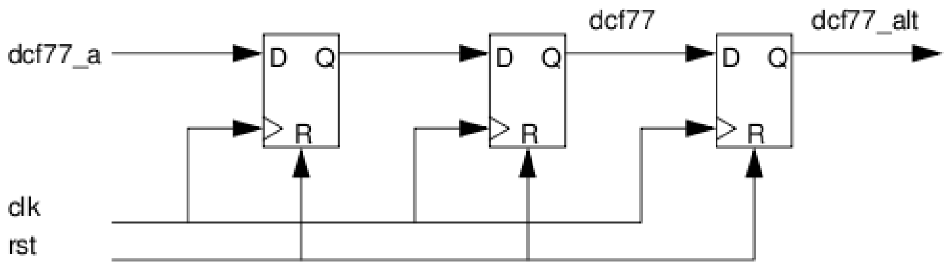
In this exercise the first modules for a radio receiver are to be designed. These are the modules ***synchronisation***, ***vorverarbeitung***, s***hift_register*** , ***date_time_register and ausgabe***. The radio receiver is to be implemented on the evaluation board which runs with 1 kHz clock rate, so you have to change the resolution in the simulator from "ns" to "µs" (represented in vsim by "us"). All time statements without unit are to be interpreted as µs.

4. Implementation of the module *Synchronization*

Complete the file **src/synchronisation.vhd**

Specification

Describe the following hardware in VHDL:



The flip flops are D-flip flops which react on a rising edge of the *clk* signal. These Flipflop have an asynchronous reset input (*rst*) which is active high (reset on *rst* = logic 1). **Use this type of flip flop in all further descriptions in this exercise!**

Implementation

Implement the module according to the specification. Take care, that you really model exactly 3 flipflops (not 4 or more).

Simulation

A force file was prepared (**dofiles/test_synchronisation.do**). The force file is plain ASCII text file which can be opened and viewed in a text editor.

Start the simulator. Execute the force file by using the menue object ***"Tools -> Execute Macro"***.

Use the given force file and add additional stimuli to verify your implementation.

5. Implementation of the module for preprocessing (vorverarbeitung)

Complete the file **src/vorverarbeitung.vhd**

Specification

In this clocked process the two input signals *dcf77* and *dcf77_alt* are decoded according to the bit which was transmitted. This bit must be available on the output signal *dcf77_dekodiert*. Additionally this process generates the signals *sekunde_start* and *minute_start*. Both signals are high for exactly one clock cycle, once a second and minute respectively is passed.

Implementation

The implementation is not simple. Therefore here are some hints.

At least two auxiliary signals are needed. You must specify a counter (integer, range 0 to 1024) and a single bit signal, which is set once a missing transmission bit (impulse) is detected.

An edge on the external signal *dcf77_a* can internally be detected by comparing the signals *dcf77* and *dcf77_alt*. On *dcf77* = '1' and *dcf77_alt* = '0' a rising edge, and on *dcf77* = '0' and *dcf77_alt* = '1' a falling edge is present.

On each rising edge of the DCF-signal the counter should be set to zero and the signal *sekunde_start* to logic one. If the counter exceeds its maximum value, the missing 59th second (impulse) is detected. Also in this case set the counter to zero. In all other cases increment the counter by one.

At a falling edge of the DFC-Signal the counter has to be evaluated. According to the counter value a logical 0 or logical 1 has to be decoded (*dcf77_dekodiert*).

Last thing missing is the generation of the signal *minute_start*. Derive yourself when this has to be set to high.

Simulation

Load the module by using the menu object "File -> Close -> Dataset -> sim" to close the previous model and then load the new module. Use the prepared force file (**dofiles/test_vorverarbeitung.do**) and use additional stimuli to verify your implementation.

6. Implementation of the module *Shift_register*

Complete the file **src/shift_register.vhd**

Specification

In this clocked process the decoded signal *dcf77_dekodiert* must be assembled to a complete time frame (59 bits). At each clock cycle in which *sekunde_start* is high, the decoded signal is to be shifted into the existing time frame. Shift is a right shift, i.e. each new *dcf77_dekodiert* is at the leftmost position of the shift register.

Implementation

Implement the module according to the specification.

Hint: Initialize the signals and variables with large widths (as the time frame for instance) with the statement "<signal_name> <= (**others** => '0');"

Simulation

Verify the VHDL-model by simulation. You will find a prepared force file in the directory **dofiles**.

7. Implementation of the module *Date_Time_Register*

Complete the file **src/date_time_register.vhd**

Specification

After each minute this clocked process writes the assembled information from the time frame into the registers *wochentag*, *stunde* and *minute*. Additionally a BCD-counter *sekunde* (BCD = Binary Coded Decimal) for the seconds is to be implemented. This counter must be incremented every second.

Implementation and simulation

Implement and simulate the VHDL-model of the module *Date_Time_Register*.

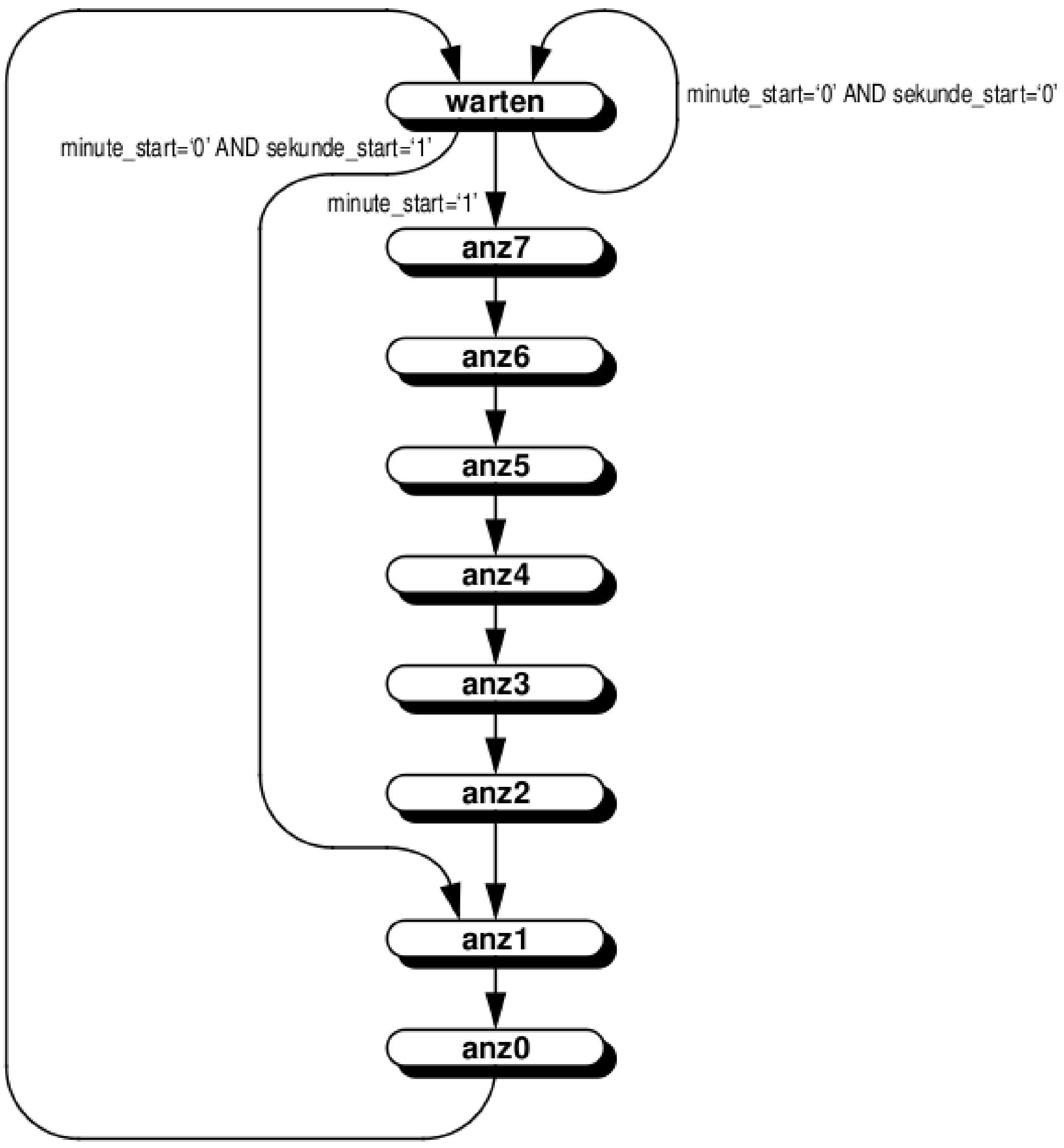
Hint: Ones and tens of the seconds should be separately counted.

8. Implementation of the module "a*usgabe*"

Complete the file **src/ausgabe.vhd**

Specification

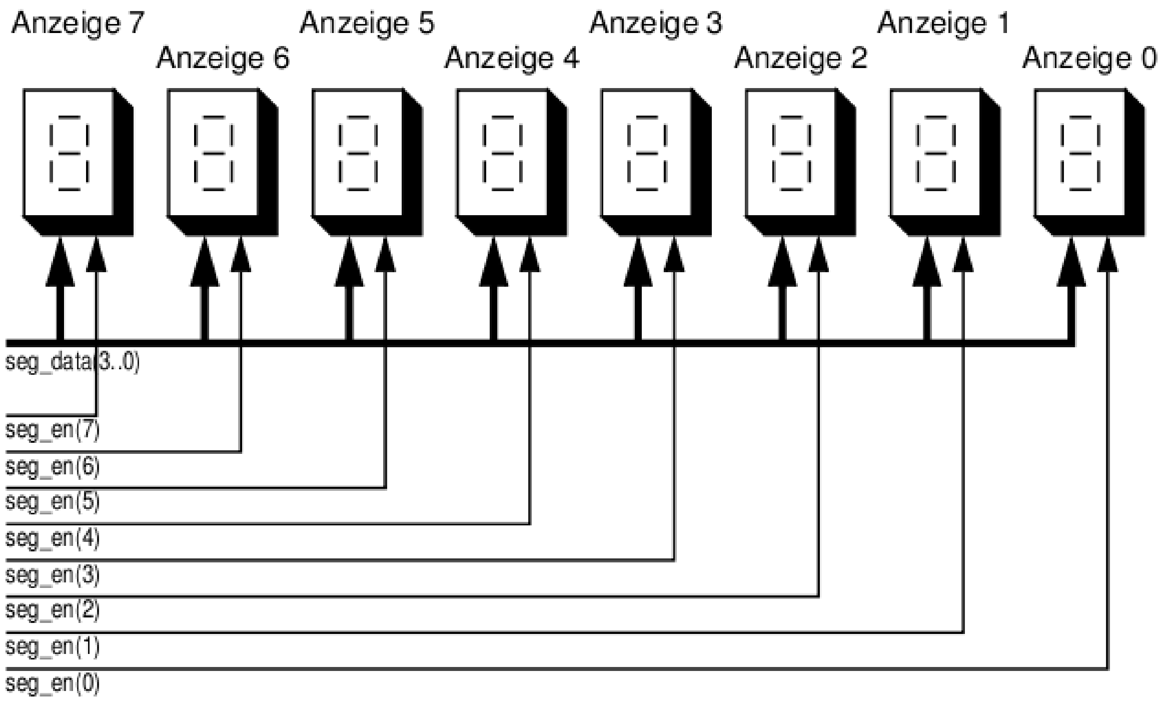
Describe the following control flow (Finite State Machine) as a finite state machine in VHDL:


warten
minute_start='0' AND sekunde_start='0'
minute_start='0' AND sekunde_start='1'
minute_start='1'
anz7
anz6
anz5
anz4
anz3
anz2
anz1
anz0

After a reset (asynchronous) the machine will be in the state "*warten*". The other transitions and transition dependencies are noted in the diagram. The output of the finite state machine is not described in the diagram, but specified in the following text.

The seven segment displays (in German “Anzeige”) on the laboratory board shall display from left (display 7) to right (display 0) the following values: weekday, empty, hour (tens), hour (ones), minute (tens), minute (ones), seconds (tens), seconds (ones). A BCD-display stays empty (not lighted) if a non-displayable value is stored in its buffer (i.e. A-F).

In the states "anz7" to "anz0" the according displays are to be updated (from left to right). This means in state "anz7" updates the display 7 (far left) with the current weekday by applying the appropriate input values to the data signal *data_seg* and set the enable signal *seg_en(7)* to high for one cycle.



Implementation and simulation

Implement and simulate (verify) the machine. There is a prepared force file in the **dofiles** directory named **test_ausgabe.do**.