S4_VHDL Specifications

Circuit Design (WS2020/21)
Prof. Dr. -Ing Andreas Siggelkow

Nils Schlegel, 32067 & Tara Jaishi, 32289

17.01.2021

History - Change Log

Target Spec.	Current version: 1.0, 2021-01-17	
	Previous version: 0.4, 2021-01-15	
	17.01.2021	Repository added
	17.01.2021	Program Design added
	15.01.2021	Testbanches added
	11.01.2021	FiniteStateMachines added
	10.01.2021	Block descriptions added
	10.01.2021	Block diagrams created and updated
	10.01.2021	Restructured
	06.11.2020	General description added
	06.11.2020	Block diagram added
	06.11.2020	Functional description added
	06.11.2020	Document created

Contents

Chapter 1

General description

IC4 is a single chip based application containing processing capabilities to detect and keep track of the amount of people in a room. It is part of a system solution to fullfill the covid-19-restrictions and regulate the amount of people in an area. This solution is only meant for a chamber with only one doorway available to enter or to exit.
The current state of the room is beeing indecated by two LEDs (room=full;room!=full). Additionally the current number of people in the room gets transmitted through a serial connection to a Computer. The application also allows to connect external hardware like the IC_S3 via a 3 wire interface which transmitts the current event as well as the current number of people in the chamber.

Chapter 2

Requirements

ID	Requirement	Priority	Verifiable	Description
General				
G01	Gen.: #persons	High	Testbench	The number of persons in a room must be known.
G02	Gen.: max	High	Testbench	The number of persons in a room must not exceed a given limit.
G03	Gen.: only one pers.	High	N/A	Only one person can either enter or leave the room at a time.
G04	Gen.: three light sensors	Medium	Testbench	Along the doorway, there are three light-curtains to allow directiontracking of possible visitors.
G05	Gen.: only one door	High	N/A	Only one door exists.
Sound				
S01	Sound: entered	High	N/A	A person entered the room, play a unique sound.
S02	Sound: left	High	N/A	A person left the room, play a unique sound.
S03	Sound: stop	High	N/A	The room is full, play a unique sound.
LED				
LED01	LED: red	High	Testbench	The maximal number of persons reached.
LED02	LED: green	High	Testbench	The maximal number of persons not reached.

ID	Requirement	Priority	Verifiable	Description
UART				
UART01	UART: 9600 baud	High	Testbench	The speed of the serial transmission should be set to 9600 baud.
UART02	UART: 8 bit	High	Testbench	The data width of the serial transmission should be set to 8 bit.
UART03	UART: no parity	High	Testbench	The serial transmission should not be checked with a parity bit.
UART04	UART: one stop bit	High	Testbench	The serial transmission should have only one stop bit.
UART06	UART: #persons	High	Testbench	The #persons should be transmitted to a PC.
PC				
PC01	PC: language	Medium	N/A	Information should be displayed on a PC, the language is C++.
PC02	PC: timestamp	Low	N/A	Every event should have a unique timestamp.
IC_S3				
IC01	IC_S3: interface	Low	Testbench	Use a three wire IF.
IC02	IC_S3: events	Low	Testbench	All events should be transmitted via the three wire IF.
IC03	IC_S3: #persons	Low	Testbench	The #persons should be transmitted via the three wire IF.

Chapter 3

Architecture Concepts

Clock Concept

This project uses only a 12MHz clock with a rising edge trigger signal. This allows to syncronize the modules within the IC_S4. Each block has the same clock cycle starting with the rising edge event. With the help of clock divider rates with less than 12MHz can be created.

Reset Concept

To reset the whole project an active low is required. That is due to FPGA board, which has buttons with active low signals build in.

FSM Concept

The FSMs (Finite State Machines) in this project are 3 process FSMs. This allows to easily construct and modify them quickly if needed.

Chapter 4

Top Level View

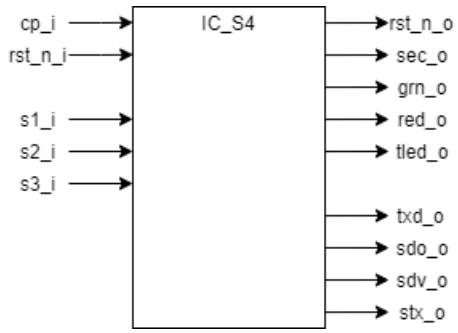

Figure 4.1: IC_S4 Top View

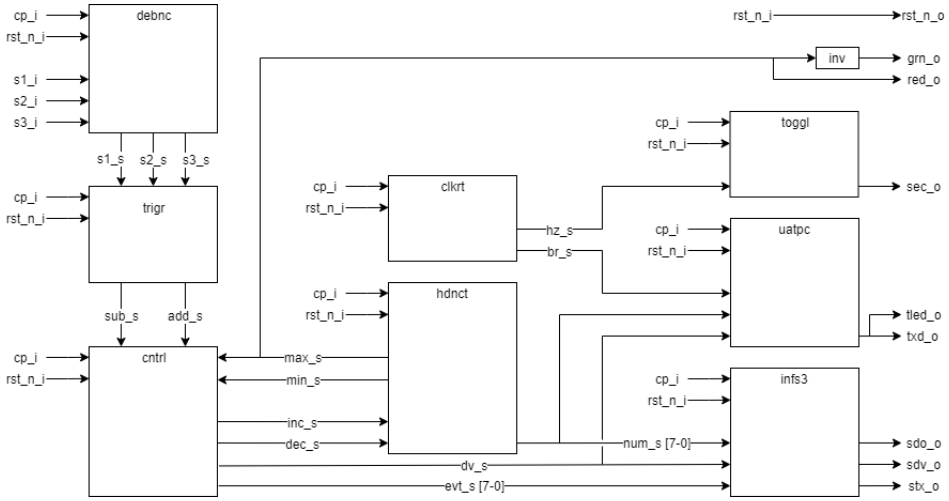

Figure 4.2: Top Level Block Diagram

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
cp_i	IN	1	Syscp, @ 12MHz
s1_i	IN	1	Sensor 1
s2_i	IN	1	Sensor 2
s3_i	IN	1	Sensor 3
rst_n_o	OUT	1	Reset state LED
sec_o	OUT	1	Pulse LED
grn_o	OUT	1	Green LED, go ahead
red_o	OUT	1	Red LED, stop, access denied
tled_o	OUT	1	Transmission LED
txd_o	OUT	1	Transmission RS-232-driver, 9k6,8N2,ASCII,to PC
sdi_o	OUT	1	S3 data value
sdv_o	OUT	1	S3 data valid
stx_o	OUT	1	S3 transmission active
br_s	SIGNAL	1	9600 baud rate signal
hz_s	SIGNAL	1	1Hz signal
add_s	SIGNAL	1	trigger signal that someone entered
sub_s	SIGNAL	1	trigger signal that someone left
inc_s	SIGNAL	1	increment headcount signal
dec_s	SIGNAL	1	decrement headcount signal
min_s	SIGNAL	1	headcount reached min
max_s	SIGNAL	1	headcount reached max
num_s	SIGNAL	[7:0]	contains the headcount number
evh_s	SIGNAL	[7:0]	contains the current event (ASCII)

OSI Layers

Layer	Part	Explanation
Application	C++ Program	Everything is printed out on the PC in a Console for better interaction
Presentation	C++ Program	The handling of the received data is also done in C++
Session	C++ Program	The program opens the serial port and catches the transmitted bits
Transport	-	The bits in this project have only one way/application to go, that is why redirecting is not needed
Network	-	Due to direct connection between the IC_S4 and the PC/IC_S3 there is no routing or addressing needed
Data Link	-	Our system does not provide any parity or checkbits to confirm the correctness
Physical	UART & 3-WireInterface	Those units are responsibe to convert the bits into a transmittable signal for physical transmittion

Functionality

When some one passes the sensors, the IC_S4 will recognize if the person passing enters the room or leaves it. According to the event the headcounter gets adjusted (either increased or decreased). With a red LED is indicated that the maximum amount of people in the room is reached. Else a green LED indicates otherwise.

Assumptions:

- Everyone completes the enter/leave process compleatly.
- No one enters, when the room is full.
- No one leaves, when room is already empty.

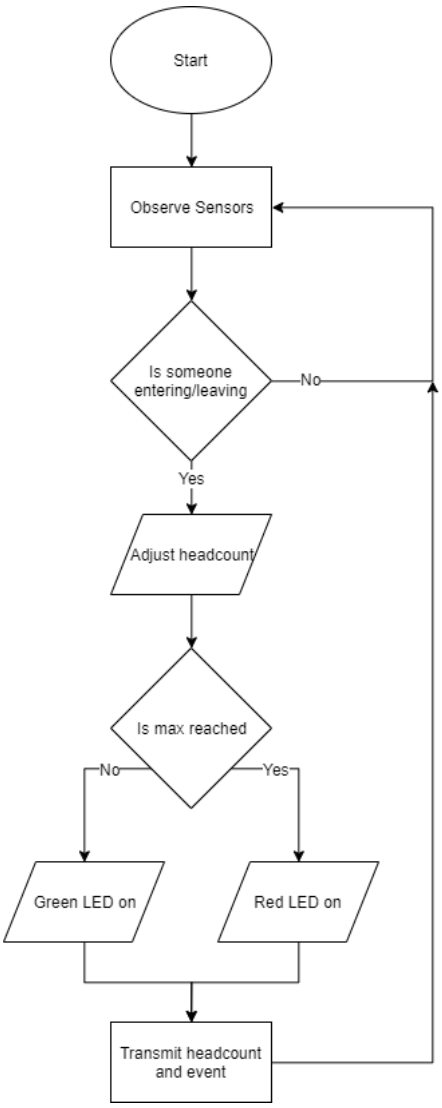

Figure 4.3: Flowchart

Chapter 5

VHDL Design

5.1 debnc: Signal Debouncing

This module debounces three incomming signals into unbounced pulsed signals. To do that it uses the module dbpul three times.

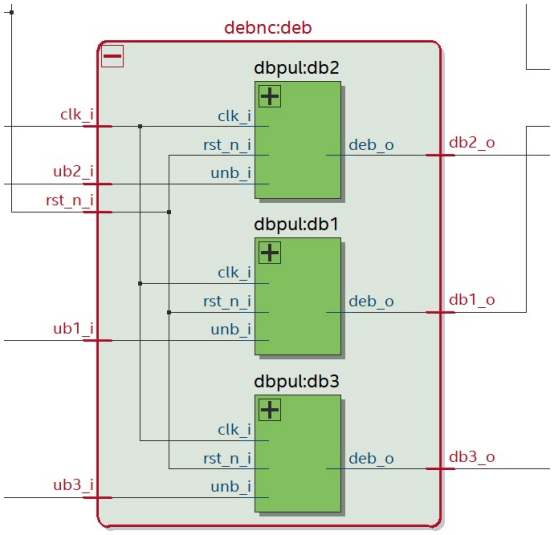

Figure 5.1: debnc_deb

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
ub1_i	IN	1	Unbounced Input 1
ub2_i	IN	1	Unbounced Input 2
ub3_i	IN	1	Unbounced Input 3
db1_o	OUT	1	Debounced Output 1
db2_o	OUT	1	Debounced Output 2
db3_o	OUT	1	Debounced Output 3

5.1.1 dbpul: Pulse Debouncer

An unbounced signal from e.g. a Button can create multiple signals when only once pressed. To get a more reliable signal the input needs to be debounced. Therefor the module waits a specified amount of time when a change happens, until the signal is in a static state. Than this module creates a single one clock cycle pulse.

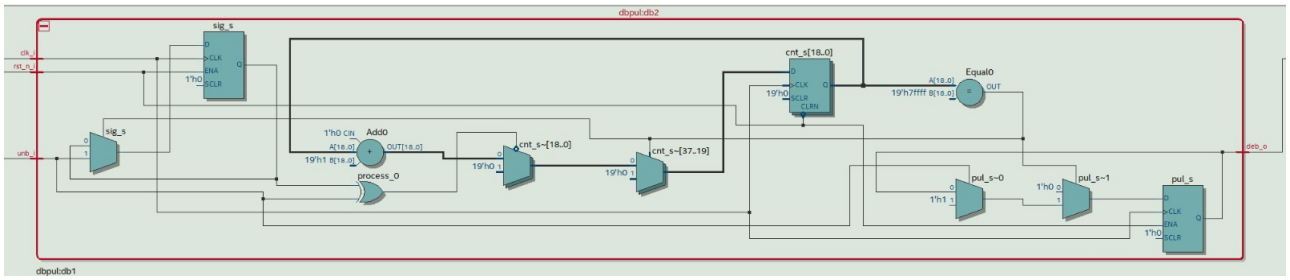

Figure 5.2: dbpul_db2

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
unb_i	IN	1	Unbounced Input
deb_o	OUT	1	Debounced Output
cnt_s	SIGNAL	1	Counter
sig_s	SIGNAL	1	Stored signal
pul_s	SIGNAL	1	Pulsed signal

Generic	Type	Value	Description
dbc_width	integer	19	Duration of debouncing (vector size)

5.2 toggl: Signal Toggle

This module works as a flipflop. Every time it gets a impulse the output toggles. It creates the hartbeat LED pulse from the 1Hz clock signal.

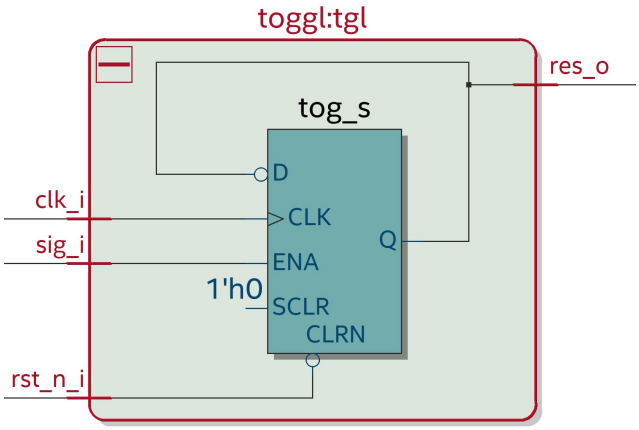

Figure 5.3: toggle_tgl

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
sig_i	IN	1	Pulseing signal
res_o	OUT	1	Toggeled output
tog_s	SIGNAL	1	Toggeling signal

5.3 clkrt: Clock Rate Generator

This module generates an 1Hz signal as well as an 9600Hz baud rate. To generate those it uses the module clkgn two times.

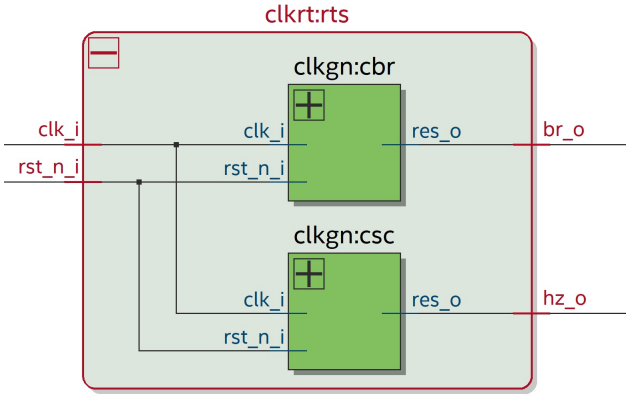

Figure 5.4: clkrt_rts

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
br_o	OUT	1	Baud Rate @9600Hz
hz_o	OUT	1	Alive Pulse @1Hz
br_s	SIGNAL	1	Baud Rate Signal (9600Hz)
hz_s	SIGNAL	1	Alive Pulse (1Hz)

5.3.1 clkgn: Clock Generator

This module has a counter, which counts the clock cycles. When the pre-set amount of clock cycles is counted a pulsed is released.

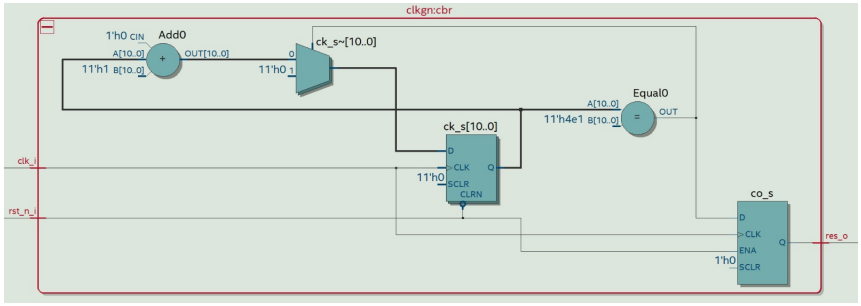

Figure 5.5: clkgn_cbr

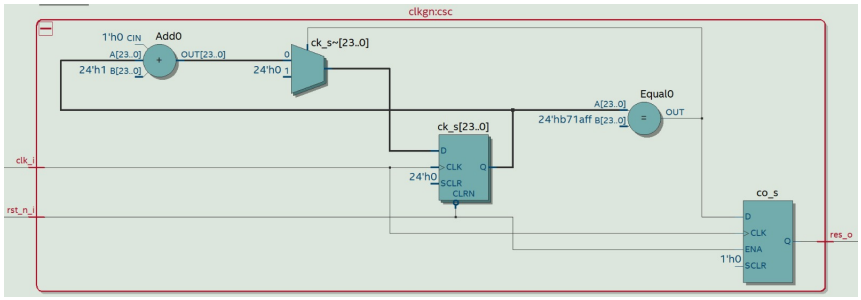

Figure 5.6: clkgn_csc

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
res_o	OUT	1	Resulting Ticks
ck_s	SIGNAL	11/24	Amount of clock cycles
co_s	SIGNAL	1	Resulting signal

Generic	Type	Value	Description
cnt_width	integer	11/24	Counter bit vector
div_cnt	integer	1.25k/12M	Clock cycle durations

5.4 trigr: Sensor Handling

This module handles the debounced and pulsed sensor inputs and detects the events. It can recognize if someone is entering or leaving the room. According to the detection a signal is send.

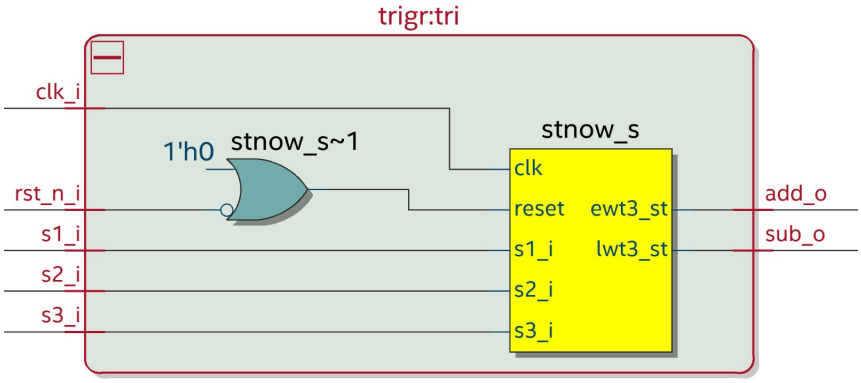

Figure 5.7: trigr_tri

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
s1_i	IN	1	Sensor 1
s2_i	IN	1	Sensor 2
s3_i	IN	1	Sensor 3
add_o	OUT	1	Person entered
sub_o	OUT	1	Person left
stnow_s	SIGNAL	-	Present state signal
stnxt_s	SIGNAL	-	Next state signal

trigr - Finite State Machine

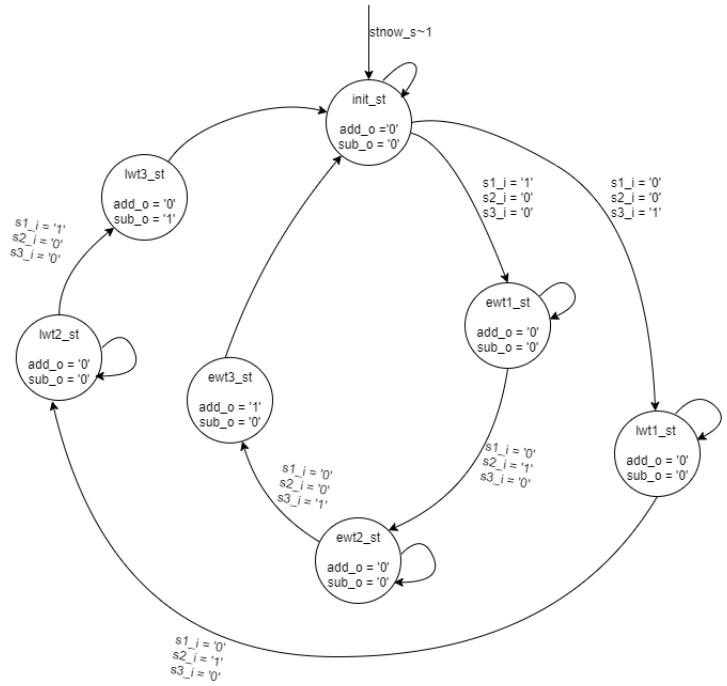

Figure 5.8: Trigger FSM

State Name	Description
init_st	Wait for either sensor 1 or sensor 2 triggered
ewt1_st	Someone entering, sensor 1 triggered wait for sensor 2
lwt1_st	Someone leaving, sensor 3 triggered wait for sensor 2
ewt2_st	Someone entering, sensor 2 triggered wait for sensor 3
lwt2_st	Someone leaving, sensor 2 triggered wait for sensor 1
ewt3_st	Someone entering, sensor 3 triggered -¿ back to init
lwt3_st	Someone leaving, sensor 1 triggered -¿ back to init

5.5 hdcnt: HeadCounter

This module stores the current number of people in the room. It increments or decrements the number if needed. Additionally it indecates if min or max number of people is reached.

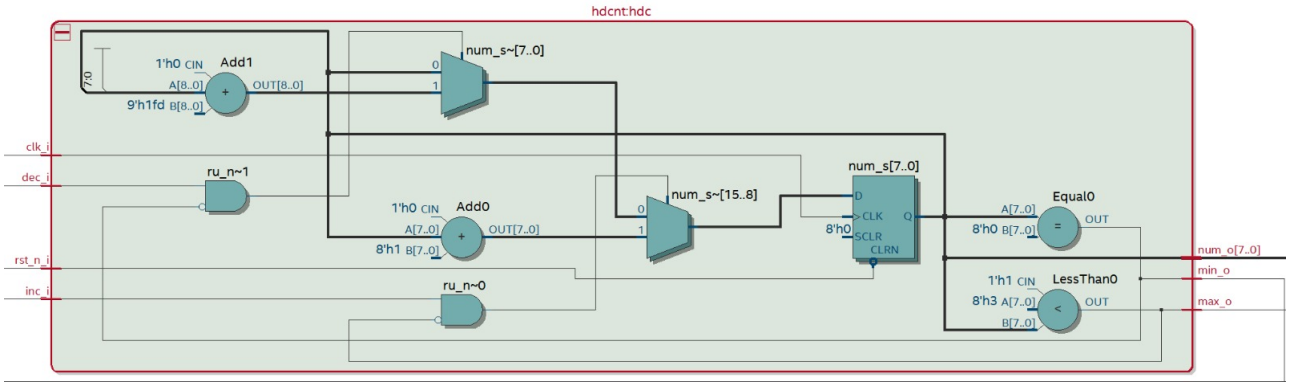

Figure 5.9: hdcnt_hdc

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
inc_i	IN	1	Increment Counter Signal
dec_i	IN	1	Decrement Counter Signal
min_o	OUT	1	Min persons in room
max_o	OUT	1	Max persons in room
num_o	OUT	[7:0]	Contains the number
num_s	SIGNAL	[7:0]	Number of people in room
min_s	SIGNAL	1	Min number reached
max_s	SIGNAL	1	Max number reached

Generic	Type	Value	Description
cnt_width	integer	8	Counter bit vector size
max_cnt	integer	3	Trigger number

5.6 cntrl: Controller

This module controls the whole system and decides what to do next according to the inputs and the current states.

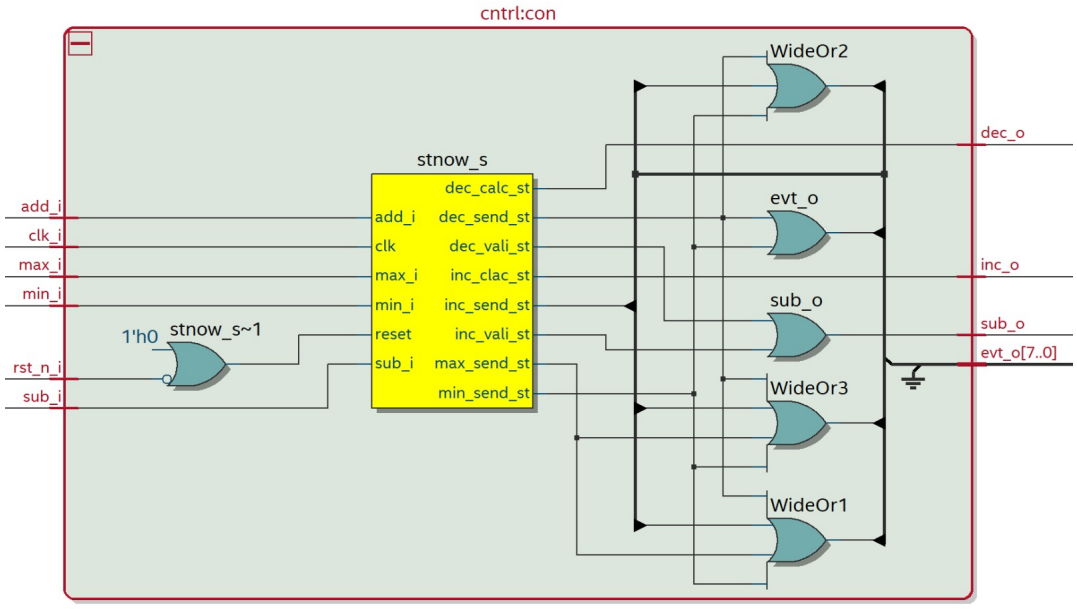

Figure 5.10: cntrl_con

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
add_i	IN	1	Person entered
sub_i	IN	1	Person left
min_i	IN	1	Min persons in room
max_i	IN	1	Max persons in room
inc_o	OUT	1	Increment Counter Signal
dec_o	OUT	1	Decrement Counter Signal
evt_o	OUT	[7:0]	Happened event char
sub_o	OUT	1	Submitt/Send Data
stnow_s	SIGNAL	-	Present state signal
stnxt_s	SIGNAL	-	Next state signal

cntrl - Finite State Machine

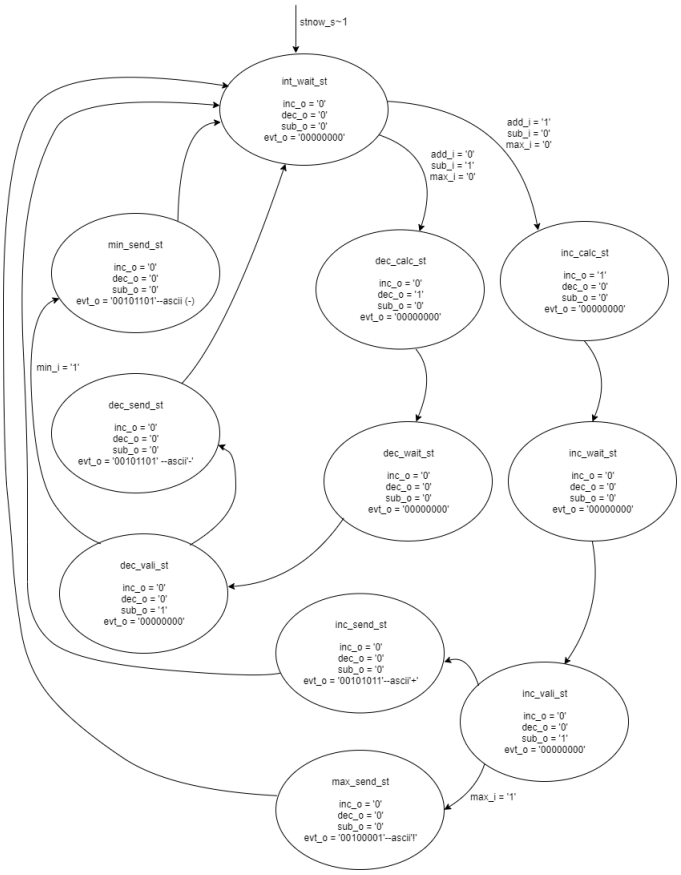

Figure 5.11: Control FSM

State Name	Description
ini_wait_st	Wait until
inc_clac_st	Send increase signal to headcounter
dec_calc_st	Send decrease signal to headcounter
inc_wait_st	Wait until headcount calculated
dec_wait_st	Wait until headcount calculated
inc_vali_st	Done calculating, check if max
dec_vali_st	Done calculating, check if min
inc_send_st	Start sending num and ascii
dec_send_st	Start sending num and ascii
min_send_st	Start sending num and ascii
max_send_st	Start sending num and ascii

5.7 uatpc: UART to PC

This module connects the IC_S4 to a PC via the RS232. It uses a 9600 baud rate, 8 bit, no parity and 1 stop bit (8N1 coding). When the send/data valid bit is received it starts sending the current headcount number byte bit by bit.

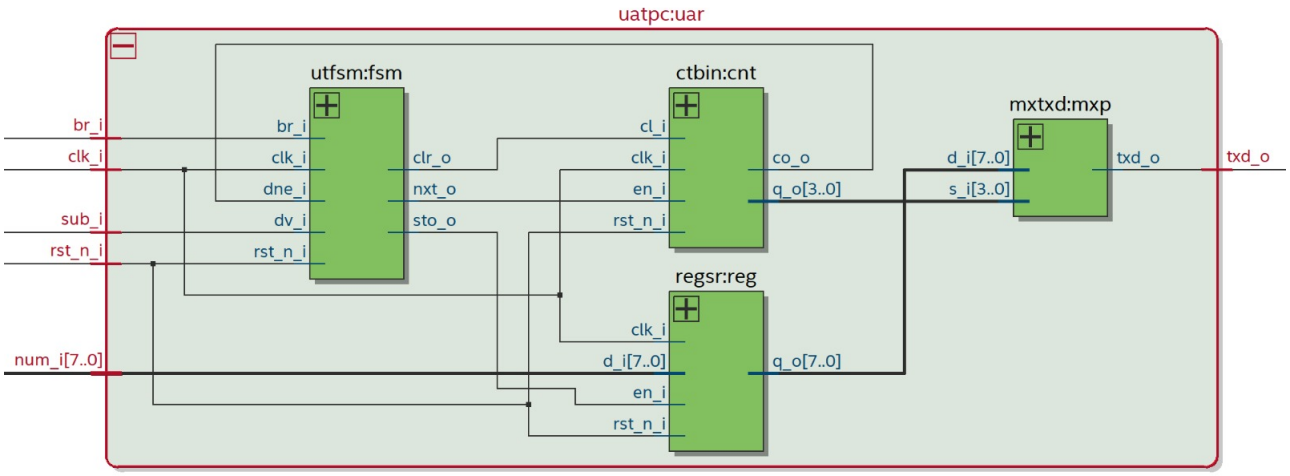

Figure 5.12: uatpc_uar

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
br_i	IN	1	Baud rate
sub_i	IN	1	Submitt/Send Data
num_i	IN	[7:0]	Headcount number
txd_o	OUT	1	Serial output
dat_s	SIGNAL	[7:0]	Headcount number
sel_s	SIGNAL	[3:0]	Select mutliplexer
ld_s	SIGNAL	1	Load in register
nxt_s	SIGNAL	1	Next Bit
clr_s	SIGNAL	1	Clear Counter
dne_s	SIGNAL	1	Last bit send
txd_s	SIGNAL	1	Transmitting data

UART transmission - Output

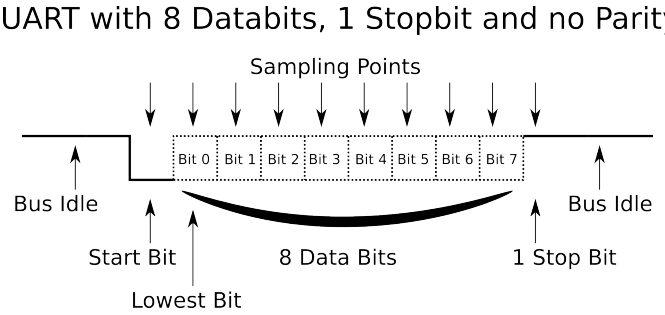

Figure 5.13: Uart FSM

When sending a byte with the UART the signal goes from the initial high state to low indicating the start bit. Then the byte is transmitted starting with the least significant bit to the highest significant bit. In the end the signal goes back to the initial high state. Every change of the signal syncronized and happens with the baud rate of 9600Hz.

5.7.1 regsr: Register to store bits

To secure the number it gets loaded and stored in the register until something new is stored.

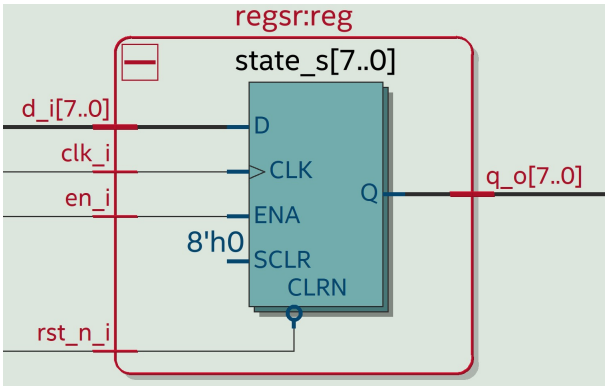

Figure 5.14: regsr_reg

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
en_i	IN	1	Store Data
d_i	IN	1	Input Data
q_o	OUT	1	Stored Data
state_s	SIGNAL	[15:0]	Stored Data

Generic	Type	Value	Description
dta_width	integer	8	Data bit vector size

5.7.2 ctbin: Binary Counter

This binary counter counts until a pre-set value. It increments, when an enable counter signal is received. The current number and the carry can always be seen.

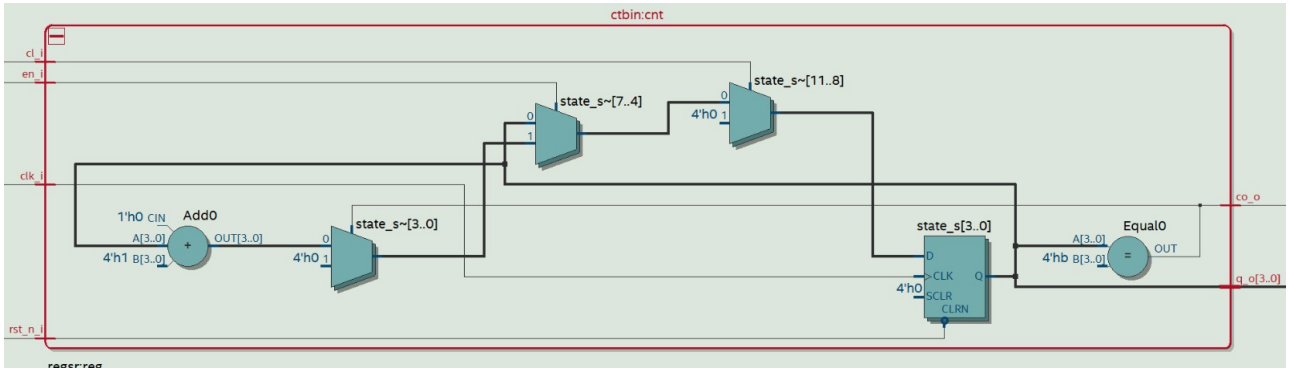

Figure 5.15: ctbin_cnt

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
en_i	IN	1	Enable Count
cl_i	IN	1	Clear Counter
co_o	OUT	1	Carry Out
q_o	OUT	[3:0]	Counter Value
state_s	SIGNAL	[3:0]	Counter Value

Generic	Type	Value	Description
cnt_width	integer	4	Counter bit vector size
cnt_max	integer	11	Trigger number

5.7.3 mxtxd: Multiplexer for TXD

This multiplexer has some special states, where it has the required stopbit and initial state, for the UART transmission build in. In total it pushes 10 bits one by one.

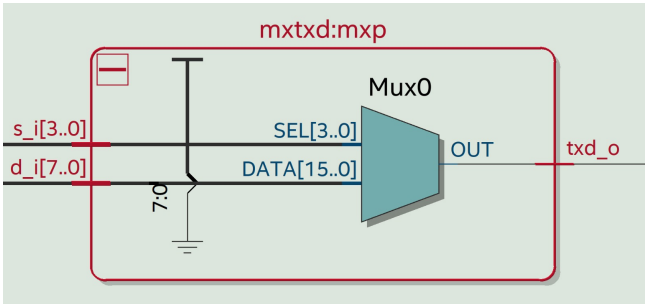

Figure 5.16: mxtxd_mxp

Signal	Direction	Width	Description
s_i	IN	[3:0]	Bit position
d_i	IN	[7:0]	Bit vector
txd_o	OUT	1	Txd, Serial Output

5.7.4 utfsm: FSM for UART

This module contains the FSM for the UART.

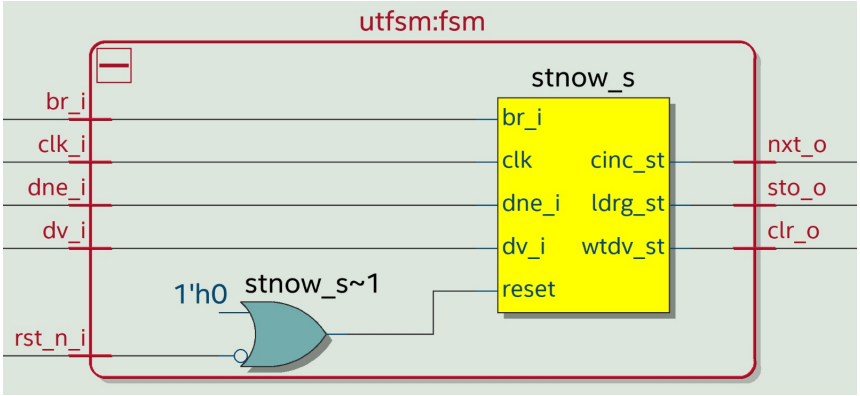

Figure 5.17: utfsm_fsm

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
dv_i	IN	1	Have new RTC or GPS-Data
br_i	IN	1	Baud-Rate to ena Counter
dne_i	IN	1	Last Bit transmitted
sto_o	OUT	1	Enable register load
clr_o	OUT	1	Clear Bit-Counters
nxt_o	OUT	1	Next Bit, inc count
stnow_s	SIGNAL	-	Present state signal
stnxt_s	SIGNAL	-	Next state signal

utfsm - Finite State Machine

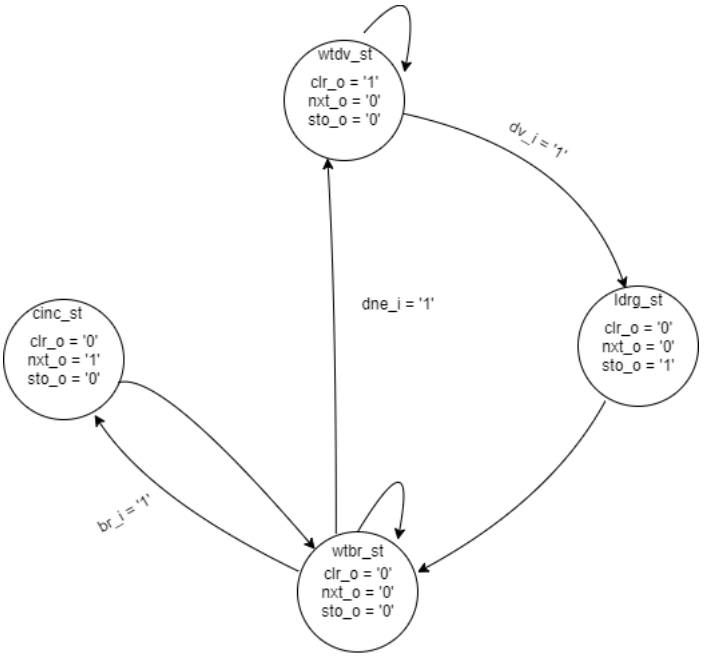

Figure 5.18: Uart FSM

State Name	Description
wtdv_st	Wait until data valid
ldrg_st	Load data in register
wtbr_st	Wait till baud rate is 1 or goto wtdv when done transmitting
cinc_st	Get next bit (increment counter)

5.8 infs3: Interface to S3

This module connects the IC_S4 to a IC_S3 via a 3-wire-interface. It passes the current event as well as the number of people in the room.

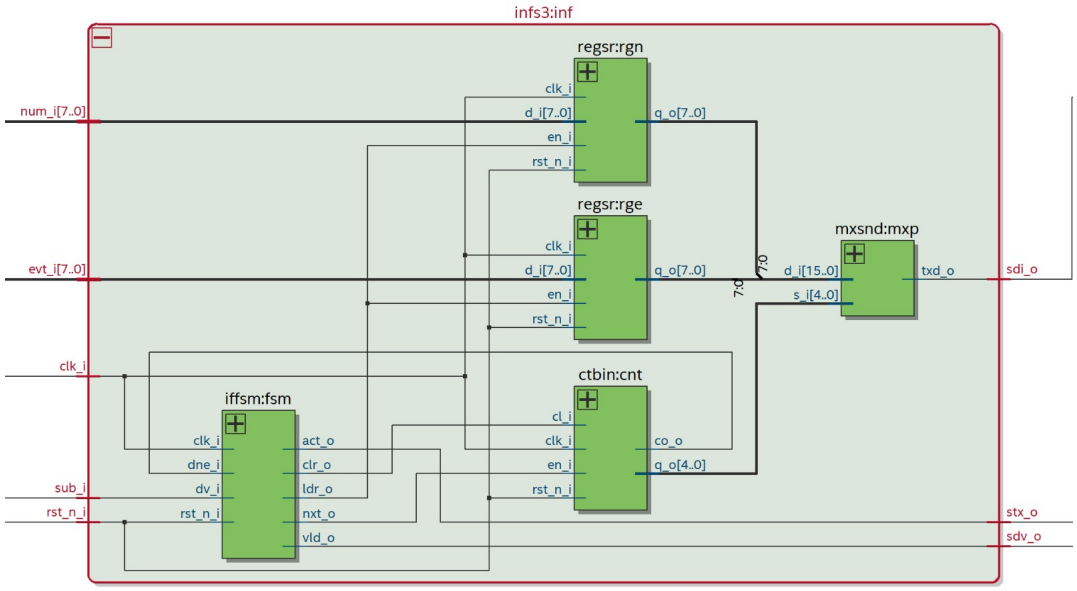

Figure 5.19: infs3_inf

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
sub_i	IN	1	Submitt/Send Data
evt_i	IN	[7:0]	Occured event char
num_i	IN	[7:0]	Head count number
sdi_o	OUT	1	S3 data value
sdv_o	OUT	1	S3 data valid
stx_o	OUT	1	S3 transmission active
ld_s	SIGNAL	1	Load in register
clr_s	SIGNAL	1	Clear Counter
dne_s	SIGNAL	1	Last bit transmitted
nxt_s	SIGNAL	1	Next Bit
sel_s	SIGNAL	[4:0]	Select mutliplexer
dat_s	SIGNAL	[15:0]	Headcount number and ASCII Event

3-Wire-Interface transmission - Output

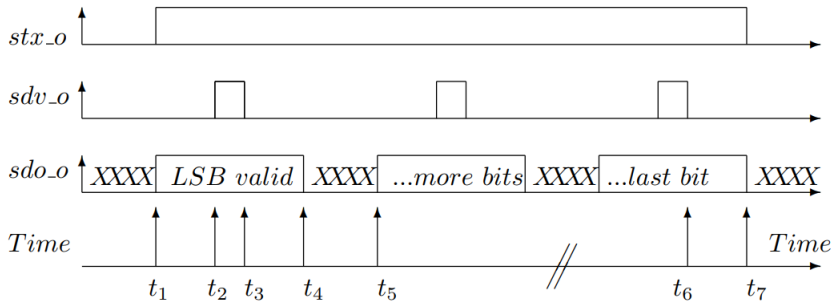

Figure 5.20: Uart FSM

The first transmitted byte is the current event ASCII code and the second byte ist than the current headcount number.

5.8.1 regsr: Register to store bits

To secure the number and the event they get loaded and stored in a register until something new is stored.

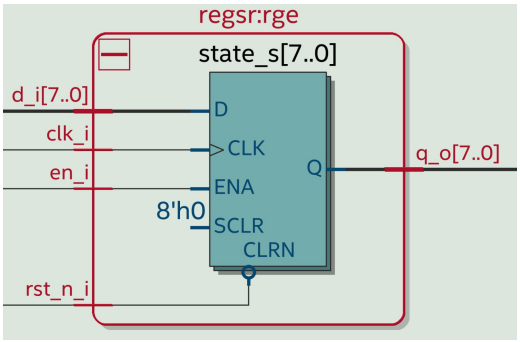

Figure 5.21: regsr_rge

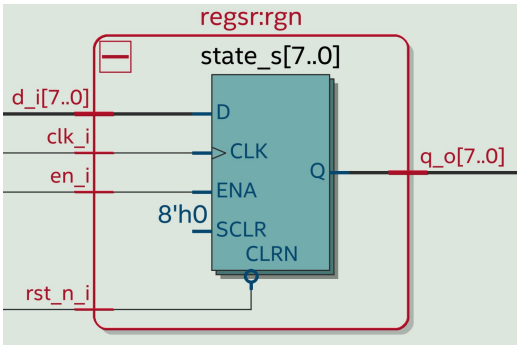

Figure 5.22: regsr_rgn

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
en_i	IN	1	Store Data
d_i	IN	[7:0]	Input Data
q_o	OUT	[7:0]	Stored Data
state_s	SIGNAL	[7:0]	Stored Data

Generic	Type	Value	Description
dta_width	integer	8	Data bit vector size

5.8.2 ctbin: Binary Counter

This binary counter counts until a pre-set value. It increments, when an enable counter signal is received. The current number and the carry can always be seen.

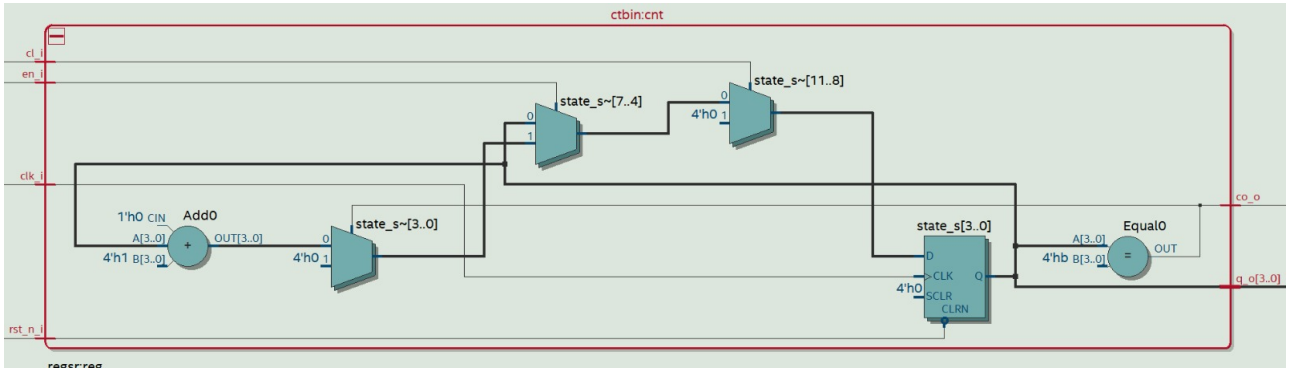

Figure 5.23: ctbin_cnt

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
en_i	IN	1	Enable Count
cl_i	IN	1	Clear Counter
co_o	OUT	1	Carry Out
q_o	OUT	[4:0]	Counter Value
state_s	SIGNAL	[4:0]	Counter Value

Generic	Type	Value	Description
cnt_width	integer	5	Counter bit vector size
cnt_max	integer	16	Trigger number

5.8.3 mxsnd: Multiplexer for Interface to S3

This multiplexer has in total 17 bits to push one by one. One of them is a initial 0 bit.

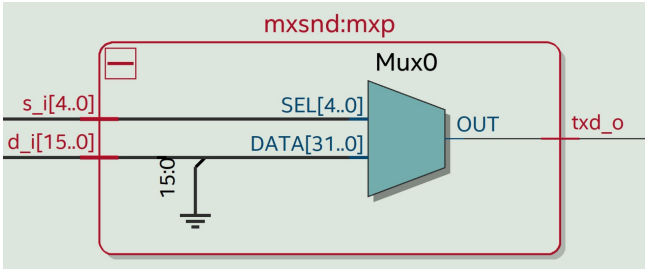

Figure 5.24: mxsnd_mxp

Signal	Direction	Width	Description
s_i	IN	[4:0]	Bit position
d_i	IN	[15:0]	Bit vector
txd_o	OUT	1	txd, Serial Output

5.8.4 iffsm: FSM for Interface to S3

This module contains the FSM for the 3-wire-interface to S3.

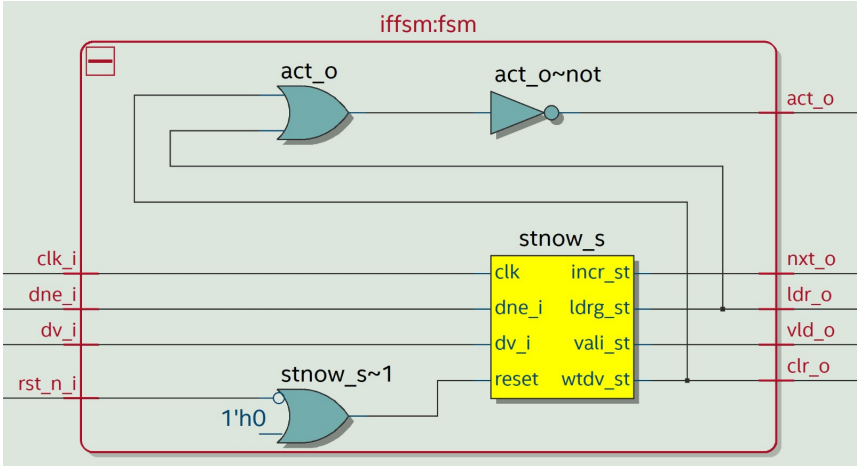

Figure 5.25: iffsm_fsm

Signal	Direction	Width	Description
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
dv_i	IN	1	Have new RTC or GPS-Data
dne_i	IN	1	Last Bit transmitted
ldr_o	OUT	1	Enable register load
act_o	OUT	1	Transmission active
vld_o	OUT	1	Data Bit valid
clr_o	OUT	1	Clear Bit-Counters
nxt_o	OUT	1	Next Bit, inc count
stnow_s	SIGNAL	-	Present state signal
stnxt_s	SIGNAL	-	Next state signal

ifs3 - Finite State Machine

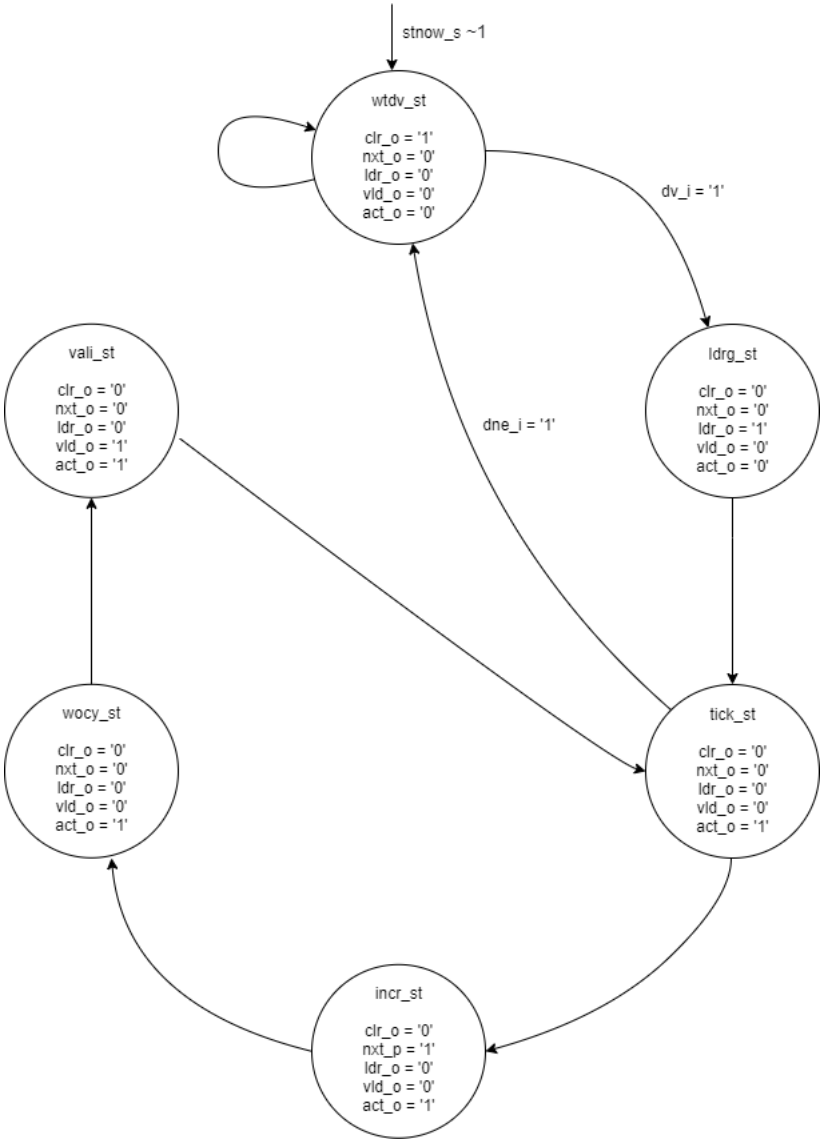

Figure 5.26: ifs3 FSM

State Name	Description
wtdv_st	Wait until data valid
ldrg_st	Load data in register
tick_st	Check if done, else go on
incr_st	Get next bit (increment counter)
wocy_st	Wait one clock cycle
vali_st	Send validation bit

Chapter 6

Testbenches

Testbenches are a great tool to test if the requirements are met. Our Project contains multiple testbenches to test the important and complex modules. The trivial modules aren't tested directly, but they are tested in testbenches of modules with higher hierarchy aswell.

Testbenches:

- TB-top
- TB-clkgn
- TB-cntrl
- TB-dbpul
- TB-hdcnt
- TB-infs3
- TB-toggl
- TB-trigr
- TB-uatpc

Chapter 7

Program Design

The pc program is written in C++. It monitors a choosen COM-Port and receives the current number of people in the room when it changes.
The program is asking for a port number (between 0 and 9) and tries to open the serial connection afterwards. When an invalid number is entered, it tries to open the serial connection at the standard port COM5.
The PC only receives the current number of people in the room, therefor the detection of the events has to be done by the program on the computer. Depending on the determined happened event the application prints out some lines in the console and makes an acoustic beep sound.

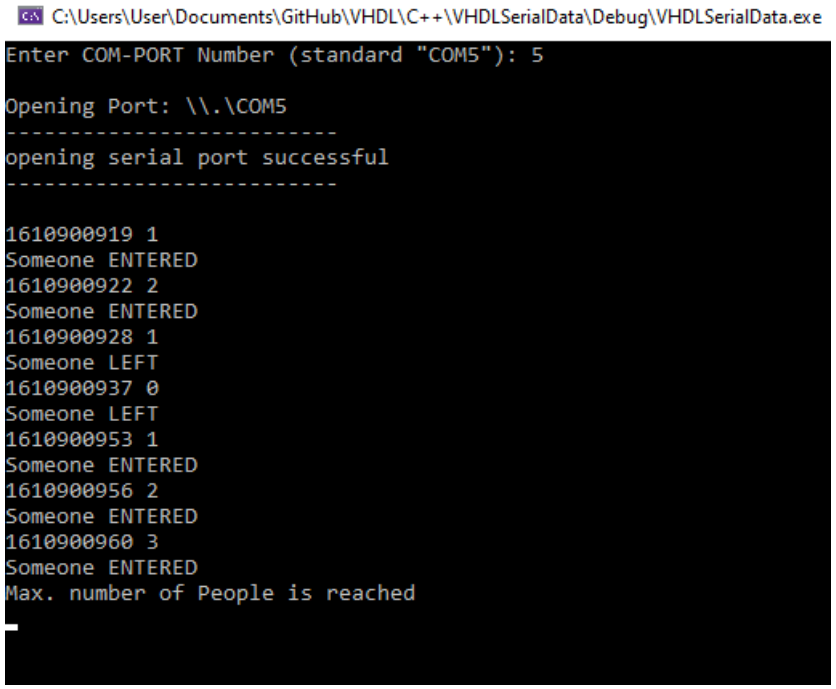

Figure 7.1: C++ Program Console

Event	Beep One	Beep Two	Description
Enter	LOW	HIGH	Someone entered the room
Leave	HIGH	LOW	Someone left the room
Stop	MEDIUM	MEDIUM	The room is full

Chapter 8

Repository - Download

The repository includes:

- VHDL source code
- VHDL testbenches
- C++ source code
- Documentation/Specification

The lastest version of this project can be downloaded directly from https://github.com/nlsy/VHDL.