

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: 0.1, 2020-11-9

17.01.2021	FiniteStateMachines added
17.01.2021	Block diagrams created and updated
17.01.2021	Block descriptions added
06.11.2020	General description added
06.11.2020	Block diagram added
06.11.2020	Functional description added
06.11.2020	Document created

Contents

1 General discription	3
1.1 Block diagram	4
2 Requirements	5
3 Top Level View	7
4 Functional description	8
4.1 Debouncing all Signals	11
4.1.1 Signal Toggle	12
4.1.2 Generates Clock Rates	13
4.1.3 Sensor Handling	14
4.1.4 HeadCounter	15
4.1.5 ControllIFSM	16
4.1.6 UART to PC	17
4.1.7 Interface to S3	18
4.1.8 Debounc Input resulting in Pulse	19
4.1.9 Clock Generator	20
4.1.10 Finite State Machine for UART	21
4.1.11 Register to store bits	22
4.1.12 Binary Counter	24
4.1.13 Multiplexer for TXD	25
4.1.14 Finite State Machine for Interface to S3	25
4.1.15 Multiplexer for Interface to S3	26
4.2 State Diagrams	27
4.2.1 Control Fsm	28
4.2.2 Trigger Fsm	29
4.2.3 Uart Fsm	30
4.2.4 ifs_3 Fsm	31

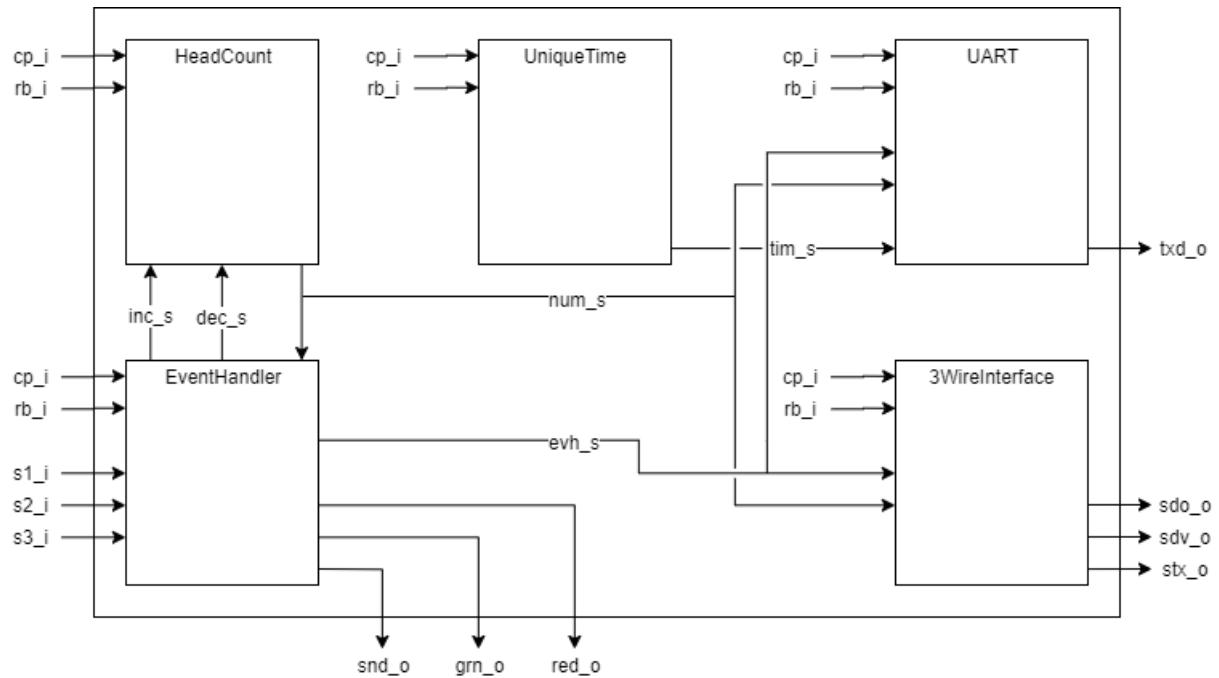
Chapter 1

General discription

IC4 is a single chip based application containing processing capabilities to detect and keep track of the amount of people in one 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 IC4 is designed on a FPGA prototype-board Max1000 with 10M16SAU169C8G device on board.

1.1 Block diagram



Chapter 2

Requirements

ID	Requirement	Importance	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	?	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 direction-tracking of possible visitors.
G05	Gen.: only one door	High	?	Only one door exists.
Sound				
S01	Sound: entered	High	Testbench	A person entered the room, play a unique sound.
S02	Sound: left	High	Testbench	A person left the room, play a unique sound.
S03	Sound: stop	High	Testbench	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	Importance	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.
UART05	UART: time	High	Testbench	The time stamp of an event should be delivered to a PC.
UART06	UART: #persons	High	Testbench	The #persons should be transmitted to a PC.
PC				
PC01	PC: language	Medium	C++-program	The information should be displayed on a PC, the language is C++.
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

Top Level View

rb_i	reset	reset, active low
cp_i	system clock	e.g. 10MHz
s1_i	light_curtain s1	log1: something passes
s2_i	light_curtain s2	log1: something passes
s3_i	light_curtain s3	log1: something passes
grn_o	LED, access garanteed	Green LED, go ahead
red_o	LED, stop, no entry	Red LED, stop, access denied
sdo_o	serial data out	drives S3 or MC
sdv_o	serial data valid	drives S3 or MC
stx_o	serial transfer active	drives S3 or MC
txd_o	serial out	to RS-232-driver, 9k6,8N2,ASCII,to PC
snd_o	sound signal	acoustic signal, to loudspeaker
inc_s	increment	increments head count when triggered
dec_s	decrement	decrements head count when triggered
num_s	number	contains the head count number
evh_s	event	contains the current event

Chapter 4

Functional description

HeadCount

Stores the current number of people in the room. It increments or decrements the number if needed.

EventHandler

It receives the signals from the light curtains and detects which event is triggered. Depending on the event it will play a sound, turn on a LED and create an output signal.

UniqueTime

This element is only counting the clock-cycles, to generate a unique timestamp.

UART

The connection to RS232 is done by the UART. It takes the unique timestamp from the UART as well as the head count from the HeadCount and event type from the EventHandler, when a signal from the EventHandler is received.

3WireInterface

The IC_3 can be connected by using the 3WireInterface. When it receives a signal from the EventHandler it should pass the head count and event type to the IC_3.

The ASIC must:

- identify, if a person enters or leaves the room
- activate or deactivate a "STOP-" or a "GO-LED"
- track and count the number of people presently in the room. Events and head-counts are transferred to IC S3 using a three-wire-Interface and via a UART to a PC.
- The transmission to the PC is 9600 baud, 8 bit, no parity, 1 stop bit.

On the PC side, get the information and display it. The programming language is C++.

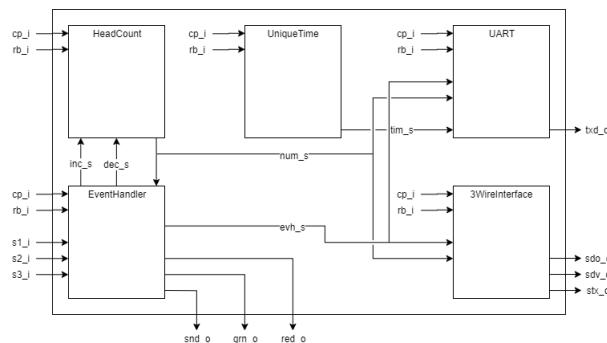


fig: Top level diagram

Table: Top level I/O

Pin	Direction	Width	Explanation
rst_n_i	IN	1	Reset, active low
clk_i	IN	1	Syscp, @ 12MHz
s1_i	IN	3	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
red_o	OUT	1	Red LED
tled_o	OUT	1	Transmission LED
txd_o	OUT	1	Transmission
sdi_o	OUT	1	S3 data value
sdv_o	OUT	1	S3 data valid
stx_o	OUT	1	S3 transmission active

Blocks of Top Level diagram and its Pin and Sifnal are described below.

4.1 Debouncing all Signals

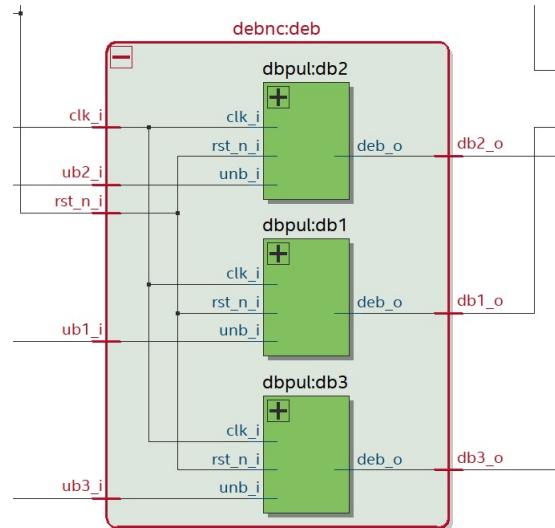


fig: debnc_deb

Pin	Direction	Width	Explanation
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

4.1.1 Signal Toggle

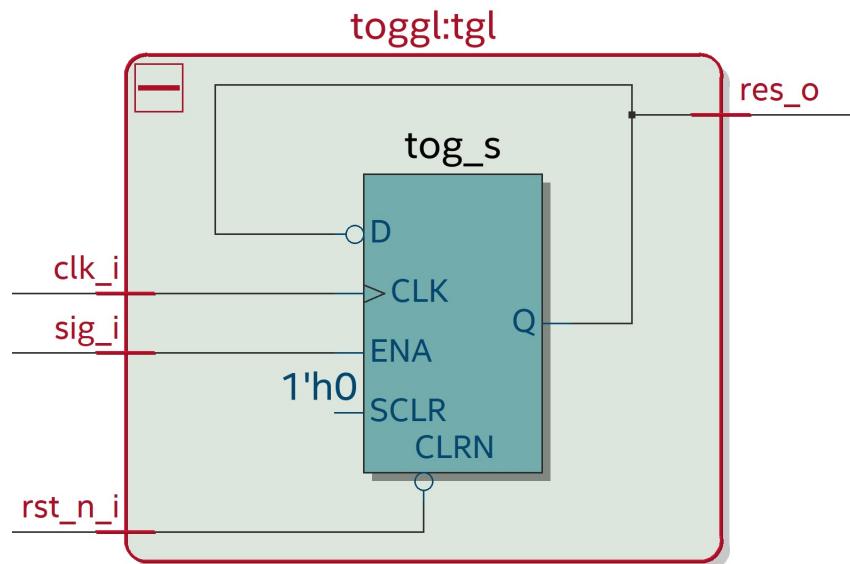


fig: toggle_tgl

Pin	Direction	Width	Explanation
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

4.1.2 Generates Clock Rates

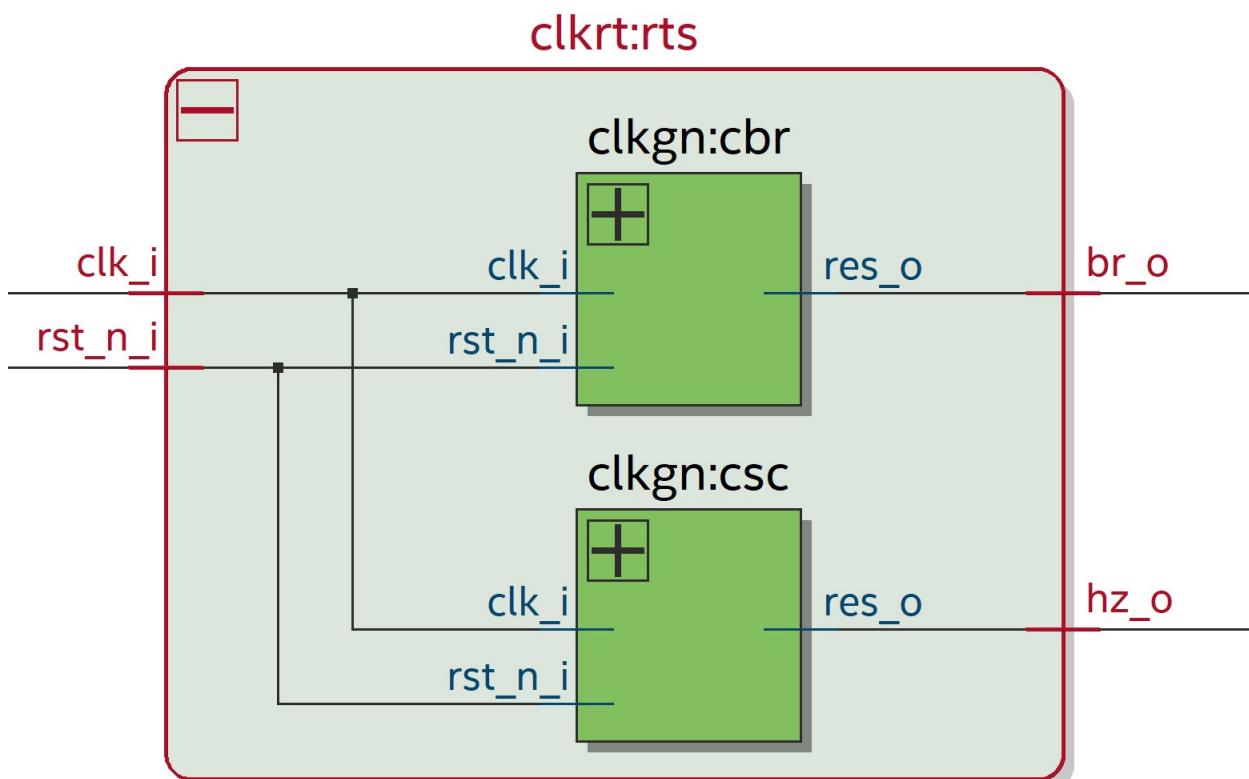


fig : clkrt_rts

Pin	Direction	Width	Explanation
<code>rst_n_i</code>	IN	1	Reset, active low
<code>clk_i</code>	IN	1	Syscp, @ 12MHz
<code>br_o</code>	OUT	1	Baud Rate @9600Hz
<code>hz_o</code>	OUT	1	Alive Pulse @1Hz

4.1.3 Sensor Handling

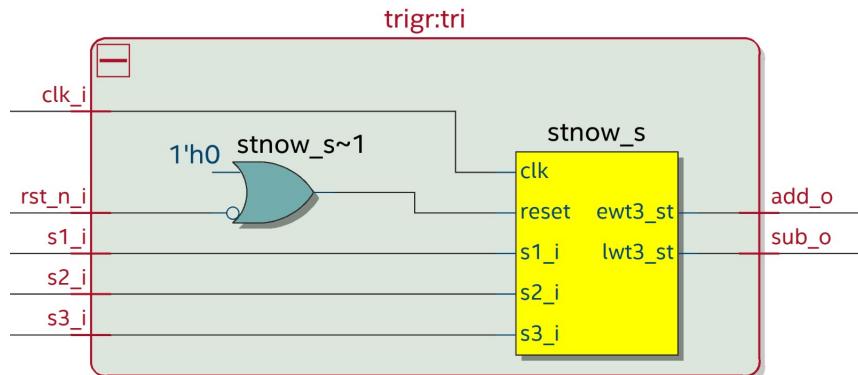


fig: trigr_tri

Pin	Direction	Width	Explanation
<code>rst_n_i</code>	IN	1	Reset, active low
<code>clk_i</code>	IN	1	Syscp, @ 12MHz
<code>s1_i</code>	IN	1	Sensor 1
<code>s2_i</code>	IN	1	Sensor 2
<code>s3_i</code>	IN	1	Sensor 3
<code>add_o</code>	OUT	1	Person entered
<code>sub_o</code>	OUT	1	Person left

4.1.4 HeadCounter

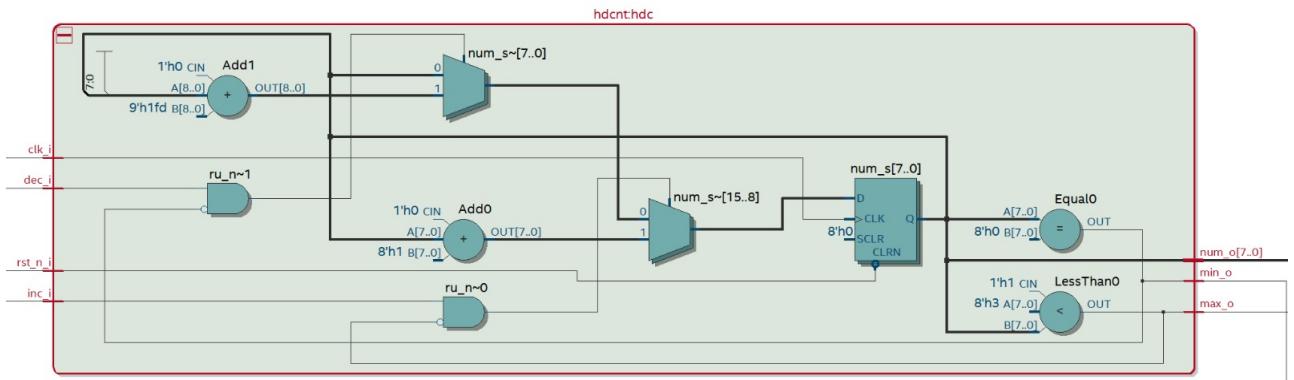


fig: hdcnt_hd

Pin	Direction	Width	Explanation
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		

Generic	Type	Explanation
cnt_width	integer	
max_cnt	integer	

4.1.5 ControlFSM

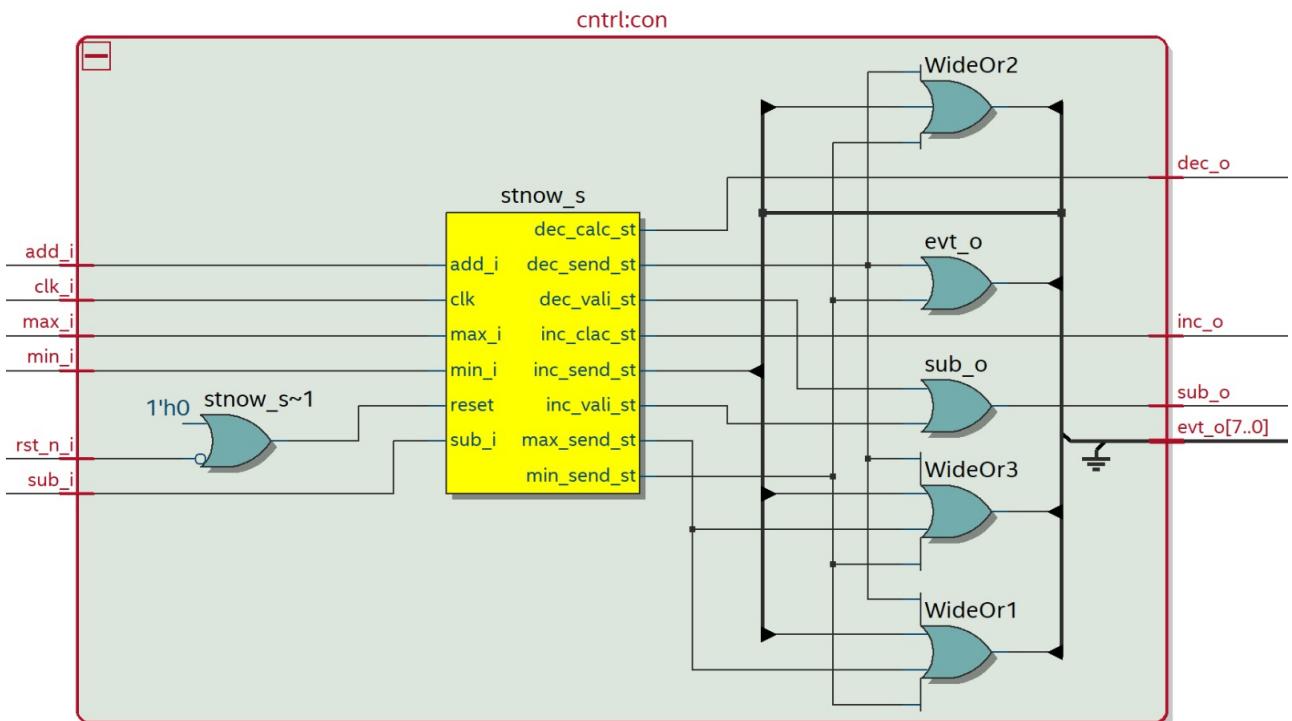


fig: cntrl_con

Pin	Direction	Width	Explanation
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		Happened event char
sub_o	OUT	1	Submitt/Send Data

4.1.6 UART to PC

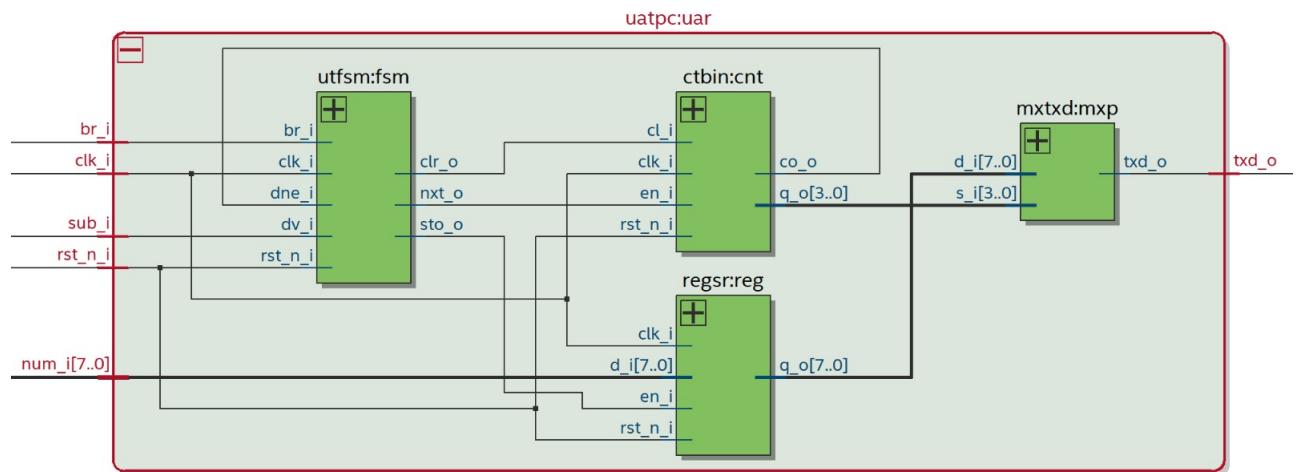


fig: uatpc_uar

Pin	Direction	Width	Explanation
<code>rst_n_i</code>	IN	1	Reset, active low
<code>clk_i</code>	IN	1	Syscp, @ 12MHz
<code>br_i</code>	IN	1	Baud rate
<code>sub_i</code>	IN	1	Submitt/Send Data
<code>num_i</code>	IN		Head count number
<code>txd_o</code>	OUT	1	Serial output

4.1.7 Interface to S3

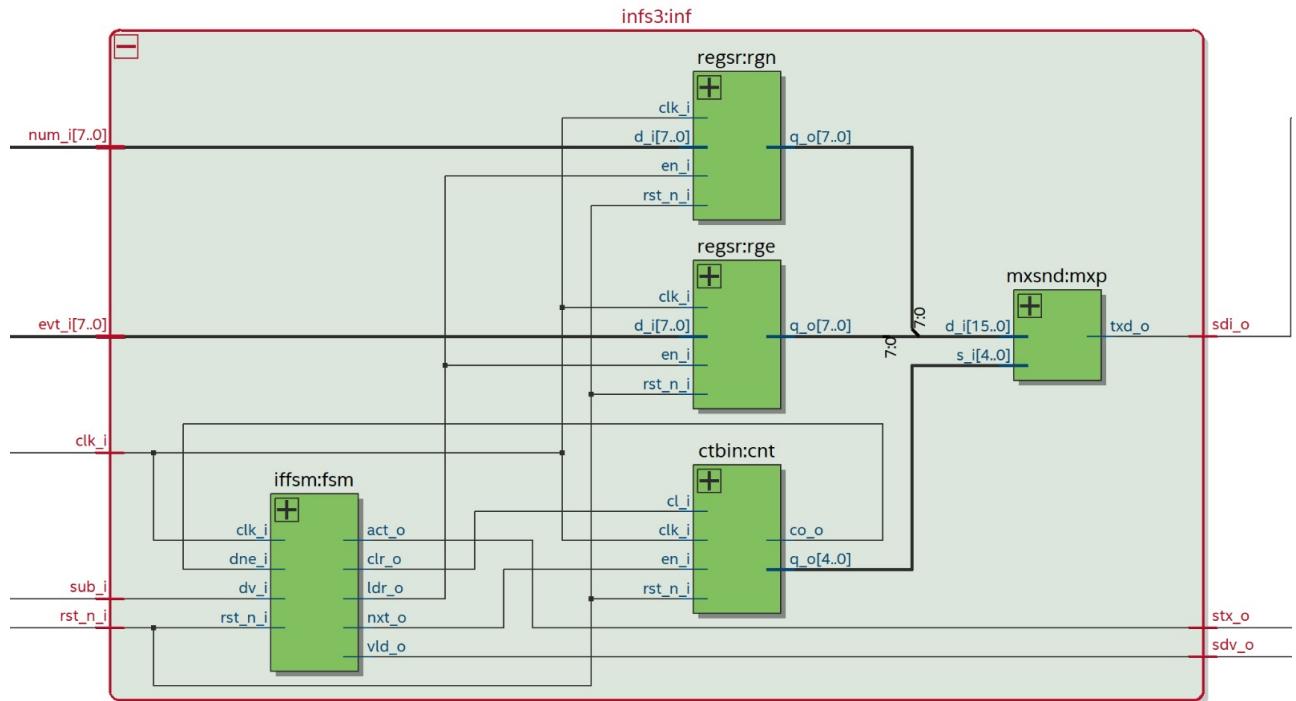


fig: inf3_inf

Pin	Direction	Width	Explanation
<code>rst_n_i</code>	IN	1	Reset, active low
<code>clk_i</code>	IN	1	Syscp, @ 12MHz
<code>sub_i</code>	IN	1	Submit/Send Data
<code>evt_i</code>	IN	1	Occured event char
<code>num_i</code>	IN	1	Head count number
<code>sdvi_o</code>	OUT	1	S3 data value
<code>sdv_o</code>	OUT	1	S3 data valid
<code>stx_o</code>	OUT	1	S3 transmission active

Lower Level Modules are described below.

4.1.8 Debounc Input resulting in Pulse

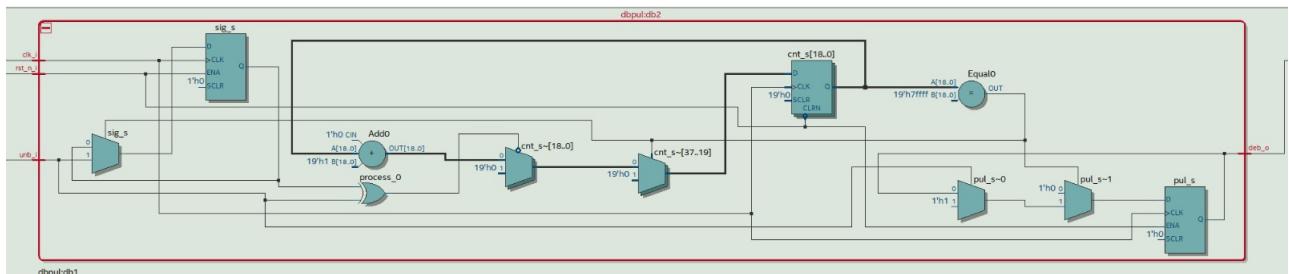


fig: dbpul_db2

Pin	Direction	Explanation
rst_n_i	IN	Reset, active low
clk_i	IN	Syscp, @ 12MHz
unb_i	IN	Unbounced Input
deb_o	OUT	Debounced Output

4.1.9 Clock Generator

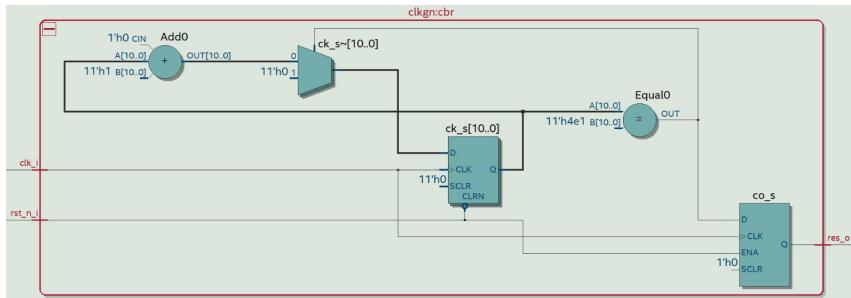


fig: clkgn_cbr

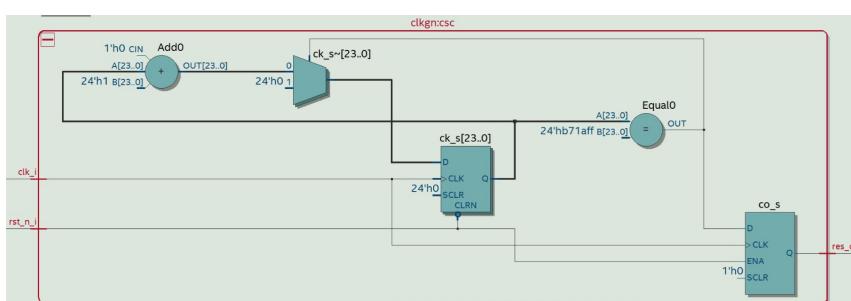


fig: clkgn_csc

Pin	Direction	Explanation
rst_n_i	IN	Reset, active low
clk_i	IN	Syscp, @ 12MHz
res_o	OUT	Resulting Ticks

Generic	Type	Explanation
cnt_width	integer	
div_cnt	integer	

4.1.10 Finite State Machine for UART

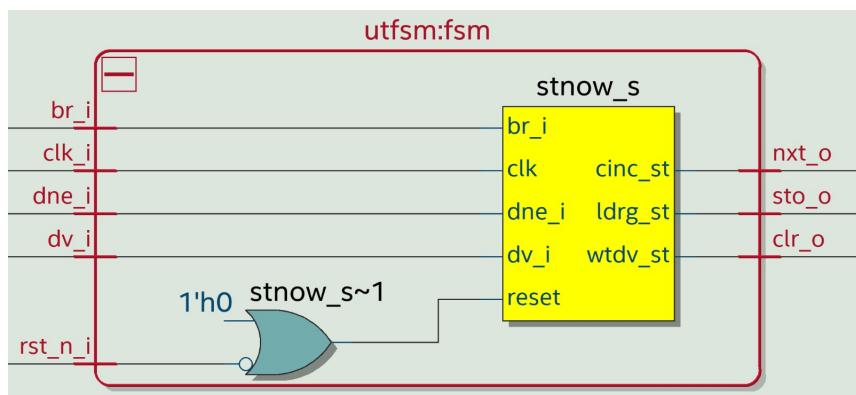


fig: `utfsm_fsm`

Pin	Direction	Explanation
<code>rst_n_i</code>	IN	Reset, active low
<code>clk_i</code>	IN	Syscp, @ 12MHz
<code>dv_i</code>	IN	Have new RTC or GPS-Data
<code>br_i</code>	IN	Baud-Rate to ena Counter
<code>dne_i</code>	IN	Last Bit transmitted
<code>sto_o</code>	OUT	enable register load
<code>clr_o</code>	OUT	clear Bit-Counters
<code>nxt_o</code>	OUT	next Bit, inc count

4.1.11 Register to store bits

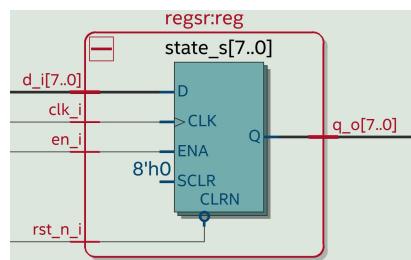


fig: regsr_reg

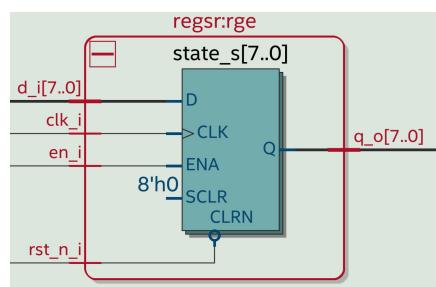


fig: regsr_rge

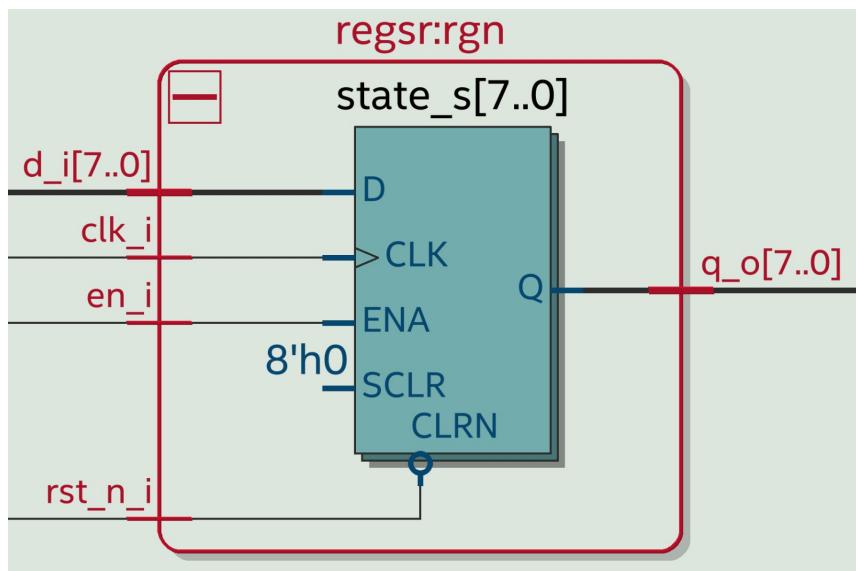


fig: `regsr_rgn`

`dta_width : integer`

Pin	Direction	Explanation
<code>rst_n_i</code>	IN	Reset, active low
<code>clk_i</code>	IN	Syscp, @ 12MHz
<code>en_i</code>	IN	Store Data
<code>d_i</code>	IN	Input Data
<code>q_o</code>	OUT	Stored Data

4.1.12 Binary Counter

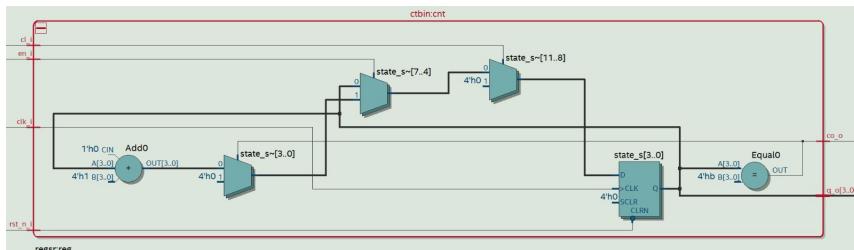


fig:ctbin_cnt

Generic	Type	Explanation
cnt_width	integer	
cnt_max	integer	

Pin	Direction	Explanation
rst_n_i	IN	Reset, active low
clk_i	IN	Syscp, @ 12MHz
en_i	IN	Enable Count
cl_i	IN	Clear Counter
co_o	OUT	Carry Out
q_o	OUT	Counter Value

4.1.13 Multiplexer for TXD

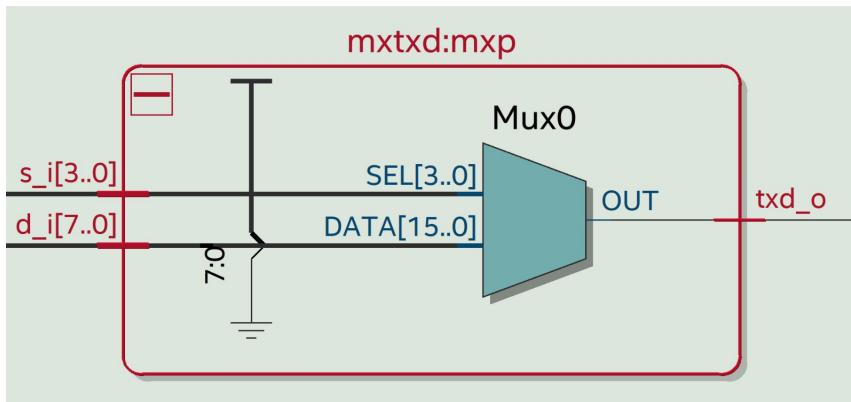


fig:mxtxd_mxp

Pin	Direction	Explanation
s_i	IN	Bit position
d_i	IN	Bit vector
txd_o	OUT	Txd, Serial Output

4.1.14 Finite State Machine for Interface to S3

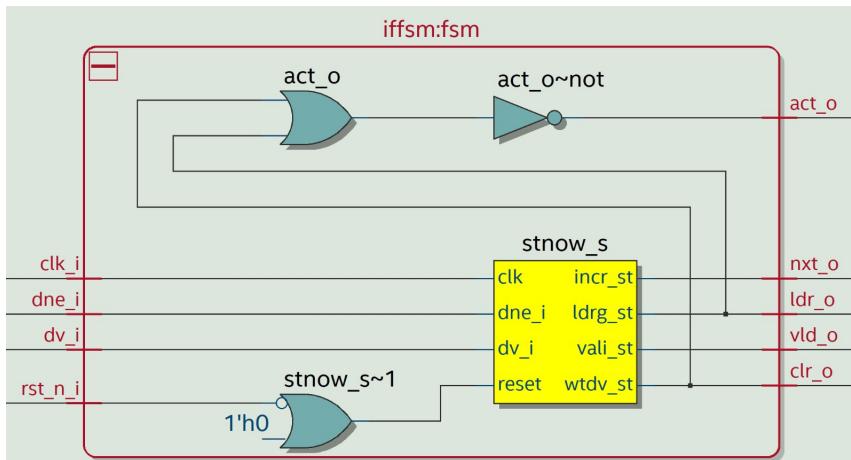


fig:iffsm_fsm

rst_n_i	IN	Reset, active low
clk_i	IN	Syscp, @ 12MHz
dv_i	IN	Have new RTC or GPS-Data
dne_i	IN	Last Bit transmitted
ldr_o	OUT	Enable register load
act_o	OUT	Transmission active
vld_o	OUT	Data Bit valid
clr_o	OUT	Clear Bit-Counters
nxt_o	OUT	Next Bit, inc count

4.1.15 Multiplexer for Interface to S3

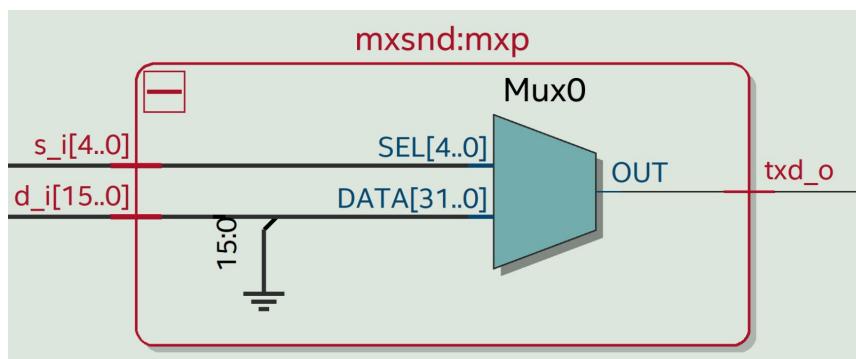


fig:mxsnd_mxp

s_i	IN	Bit position
d_i	IN	Bit vector
txd_o	OUT	txd, Serial Output

4.2 State Diagrams

The State Transition and the Output states of Uart, Control, Trigger and ifs_3 are given below

State Name	Explanation
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

4.2.1 Control Fsm

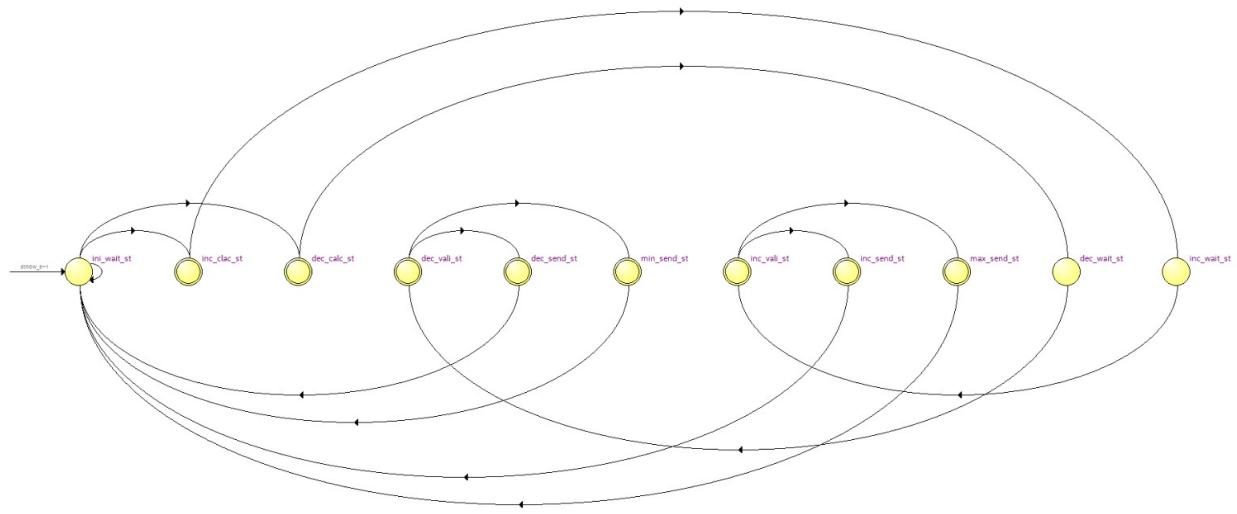


fig: Control FSM

Source State	Destination State	Condition
ewt1_st	ewt2_st	$(!s3.i).(!s1.i).(s2.i)$
ewt1_st	ewt1_st	$(!s3.i).(!s1.i).(!s3.i) + (!s3.i).(s1.i) + (s3.i)$
ewt2_st	ewt3_st	$(s3.i).(!s1.i).(!s2.i)$
ewt2_st	ewt2_st	$(!s3.i) + (s3.i).(!s1.i) + (s3.i).(s1.i)$
ewt3_st	init_st	
init_st	ewt1_st	$(!s3.i).(s1.i).(!s2.i)$
init_st	lwt1_st	$(s3.i).(!s1.i).(!s2.i)$
init_st	init_st	$(!s3.i).(!s1.i) + (!s3.i).(s1.i).(s2.i) + (s3.i).(!s3.i).(s2.i) + (s3.i).(s1.i)$
lwt1_st	lwt2_st	$(!s3.i).(!s1.i).(!s2.i)$
lwt1_st	lwt1_st	$(!s3.i).(!s1.i).(!s2.i) + (!s3.i).(s1.i) + (s3.i)$
lwt2_st	lwt3_st	$(!s3.i).(s1.i).(!s2.i)$
lwt2_st	lwt2_st	$(!s3.i).(!s1.i) + (!s3.i).(s1.i).(s2.i) + (s3.i)$
lwt3_st	init_st	

4.2.2 Trigger Fsm

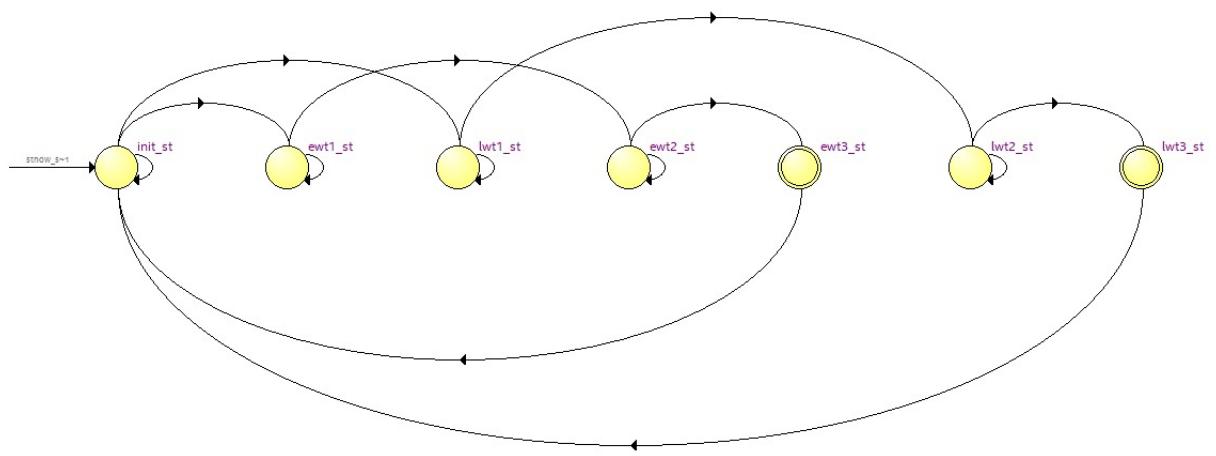


fig: Trigger FSM

State Name	Description
<code>init_st</code>	wait for either sensor 1 or sensor 2 triggered
<code>ewt1_st</code>	someone entering, sensor 1 triggered wait for sensor 2
<code>lwt1_st</code>	someone leaving, sensor 3 triggered wait for sensor 2
<code>ewt2_st</code>	someone entering, sensor 2 triggered wait for sensor 3
<code>lwt2_st</code>	someone leaving, sensor 2 triggered wait for sensor 1
<code>ewt3_st</code>	someone entering, sensor 3 triggered - goes back to init
<code>lwt3_st</code>	someone leaving, sensor 1 triggered - goes back to init

Source State	Destination State	Condition

4.2.3 Uart Fsm

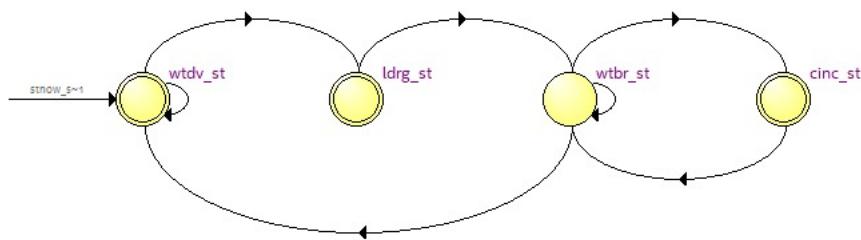


fig: Uart FSM

State Name	Description
<code>wtdv_st</code>	wait until data valid
<code>ldrg_st</code>	load data in register
<code>wtbr_st</code>	wait till baud rate is 1 or goto wtdv when done transmitting
<code>cinc_st</code>	get next bit (increment counter)

Source State	Destination State	Condition

4.2.4 ifs_3 Fsm

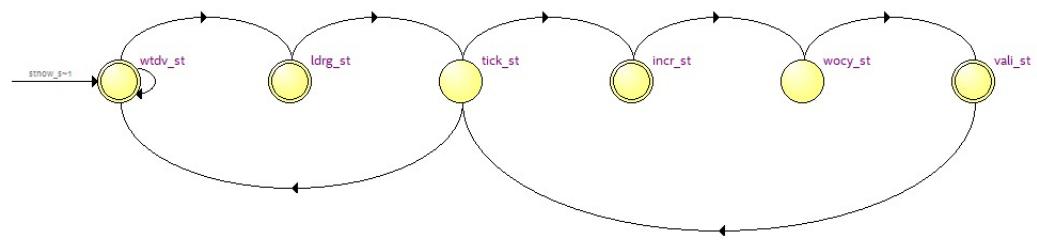


fig: ifs_3 FSM

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)
wocry_st	wait one clock cycle
vali_st	send validation bit

Source State	Destination State	Condition