Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Verification Laboratory

Functional verification in practice

Dr. Lázár Jani, Dr. Péter Horváth

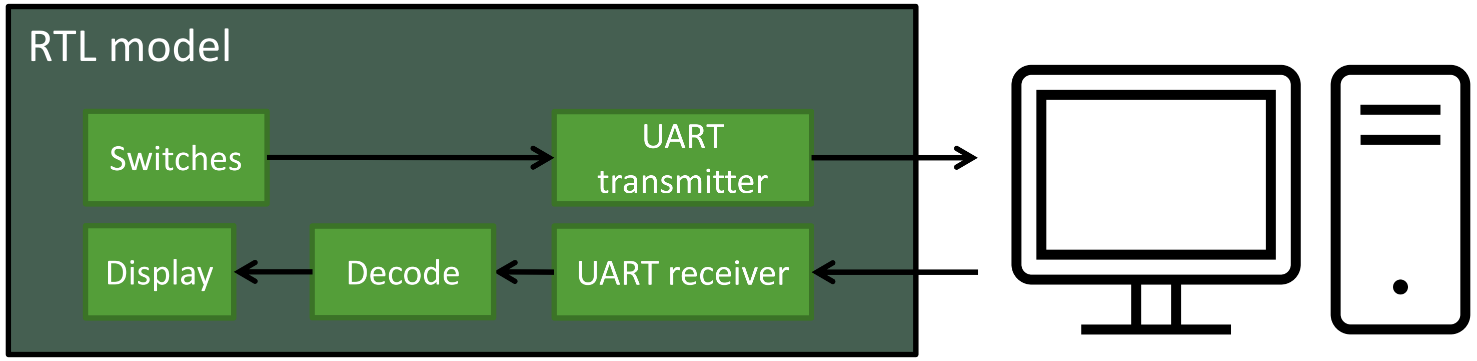
Department of Electron Devices, 2022

Design Under Verification

Requirements

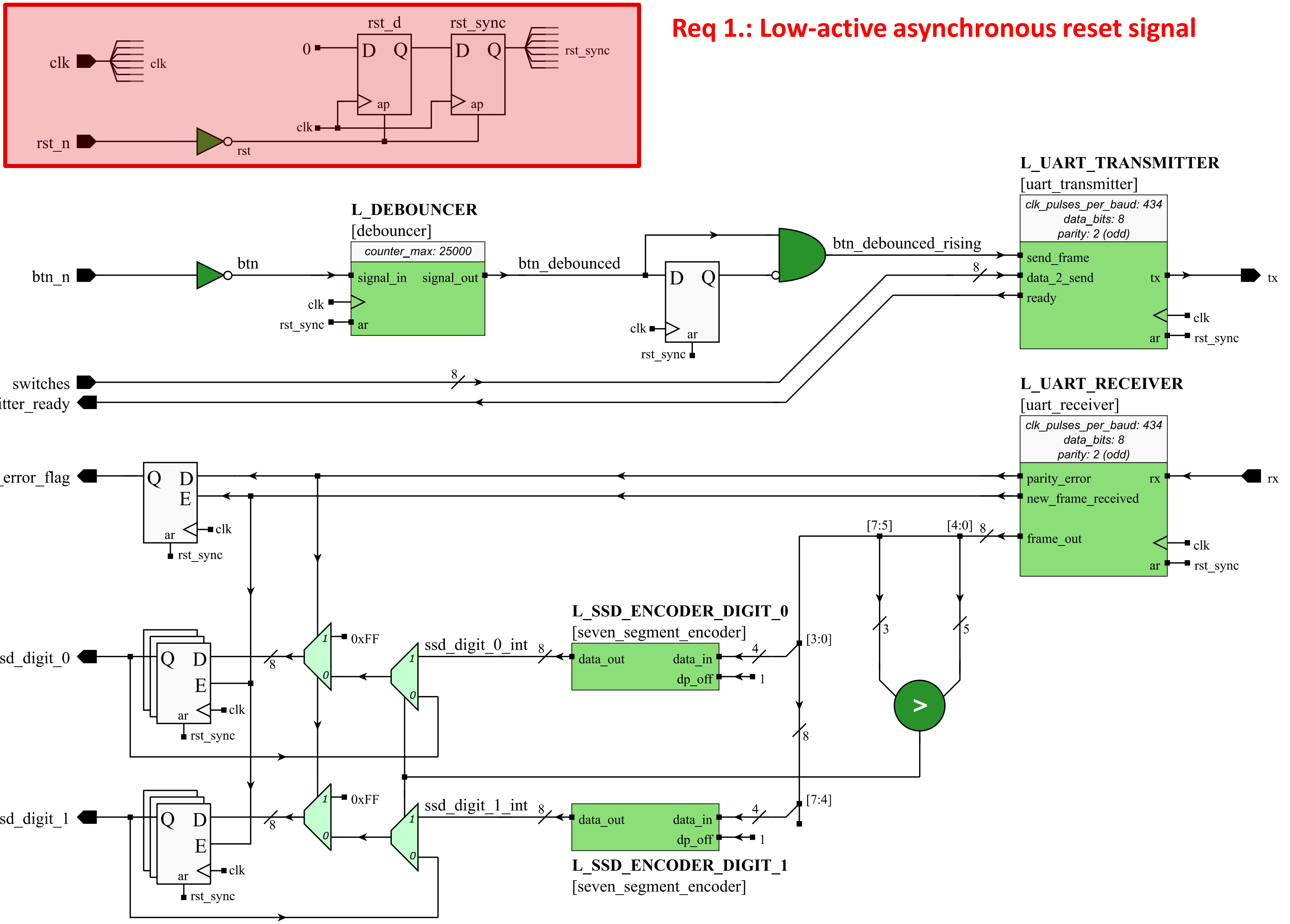
“Customer” provided the following list of requirements

1. Low-active asynchronous reset signal
2. 50 MHz clock frequency
3. UART transmission
   a) Data coded by the switches shall be transmitted on a press of a button
   b) UART frame shall be 8O1, baud rate 115.2 kbps
   c) The input shall be debounced: half period < 450 us, bouncing delay < 2 ms
4. UART reception
   a) UART frame shall be 8O1, baud rate 115.2 kbps
   b) Display shall be updated only if the unsigned value of the received data’s upper 3 bits is larger than the lower 5 bits’ value
   c) Display shall be dimmed on parity error



RTL model in more detail

Req 1.: Low-active asynchronous reset signal



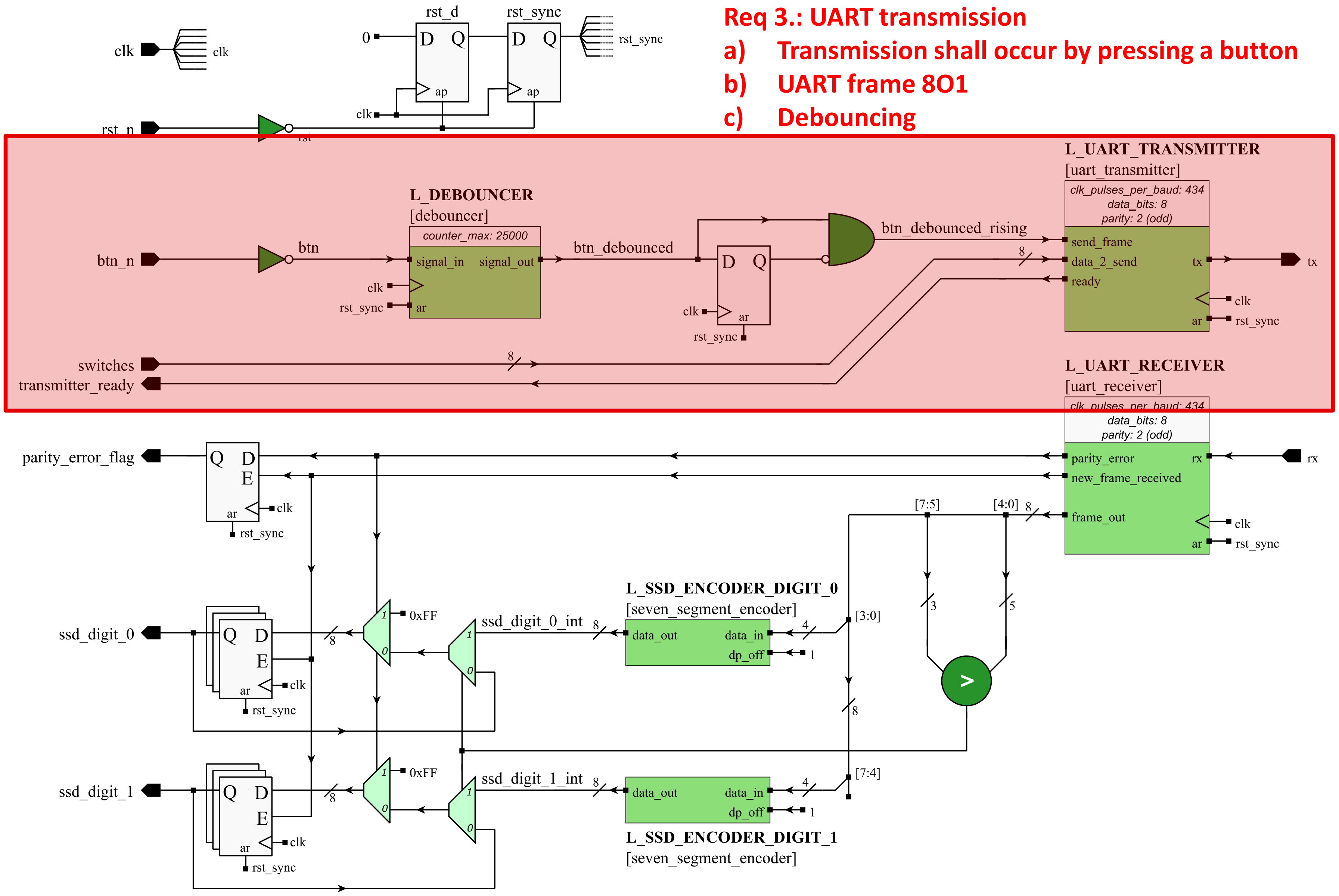
RTL model in more detail

Req 3.: UART transmission

a) Transmission shall occur by pressing a button

b) UART frame 8O1

c) Debouncing



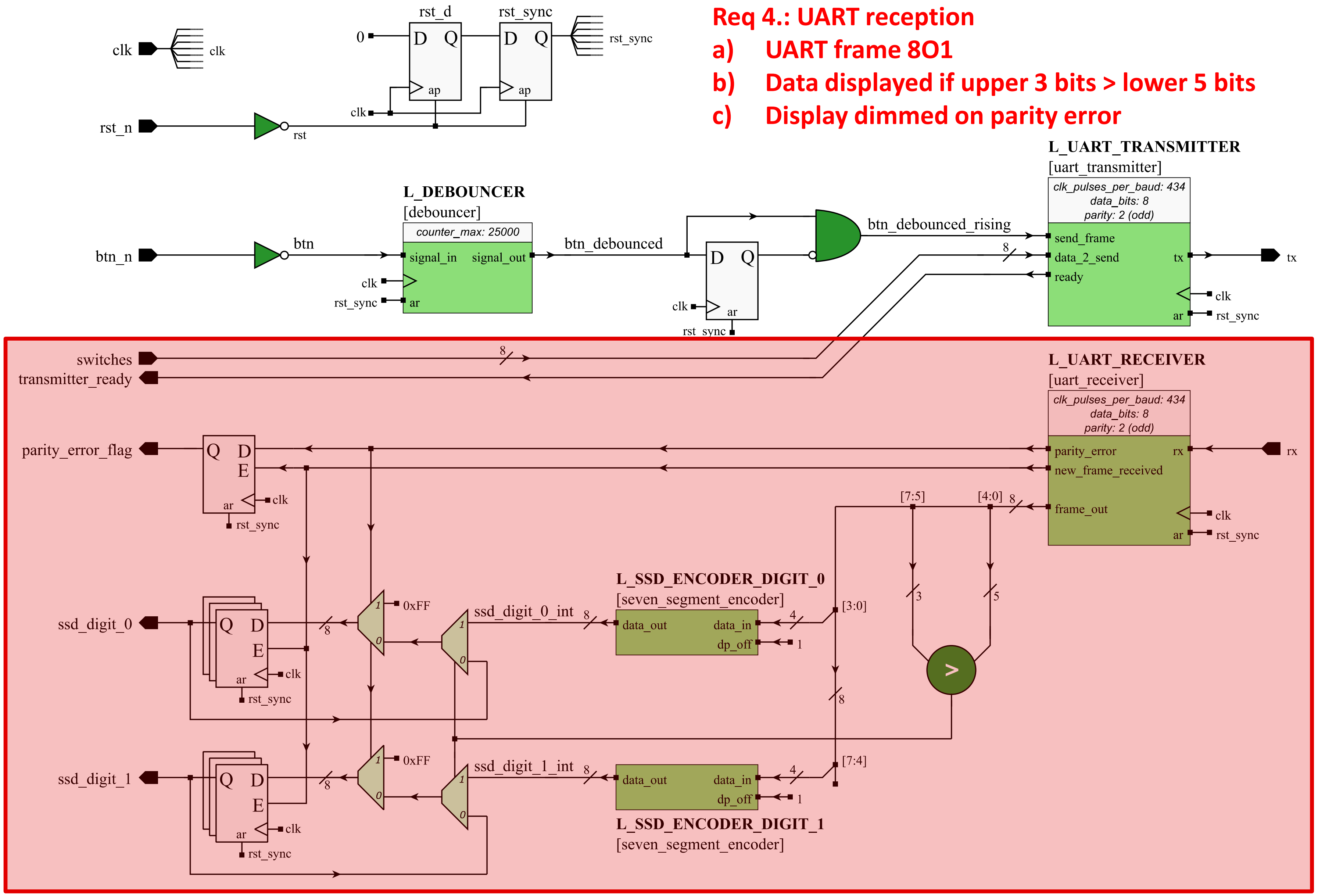
RTL model in more detail

Req 4.: UART reception

a) UART frame 8O1

b) Data displayed if upper 3 bits > lower 5 bits

c) Display dimmed on parity error



Starting QuestaSim

- OpenSUSE Tumbleweed
- Open a terminal (Ctrl+Alt+t)
  - Type the following command: /eda/run_centos
- Download the source codes and save it here: /home/x11Docker/
- Extract the archive
- Create a new project in QuestaSim
- Add existing source files to the project

First testbench

“Hand crafted” stimuli

- Hand crafted testbench
  - Stimuli implemented manually
  - Checks were implemented for a specific stimuli
- Lots of copy and paste code
  - Software developer instinct suggest something is wrong

```
-- send 0x2D
rx <= '0'; wait for 8.67 us;  -- start bit
rx <= '1'; wait for 8.67 us;  -- bit 0
rx <= '0'; wait for 8.67 us;  -- bit 1
rx <= '1'; wait for 8.67 us;  -- bit 2
rx <= '1'; wait for 8.67 us;  -- bit 3
rx <= '0'; wait for 8.67 us;  -- bit 4
rx <= '1'; wait for 8.67 us;  -- bit 5
rx <= '0'; wait for 8.67 us;  -- bit 6
rx <= '0'; wait for 8.67 us;  -- bit 7
rx <= '1'; wait for 8.67 us;  -- parity bit
rx <= '1'; wait for 8.67 us;  -- stop bit
```

“Hand crafted” stimuli

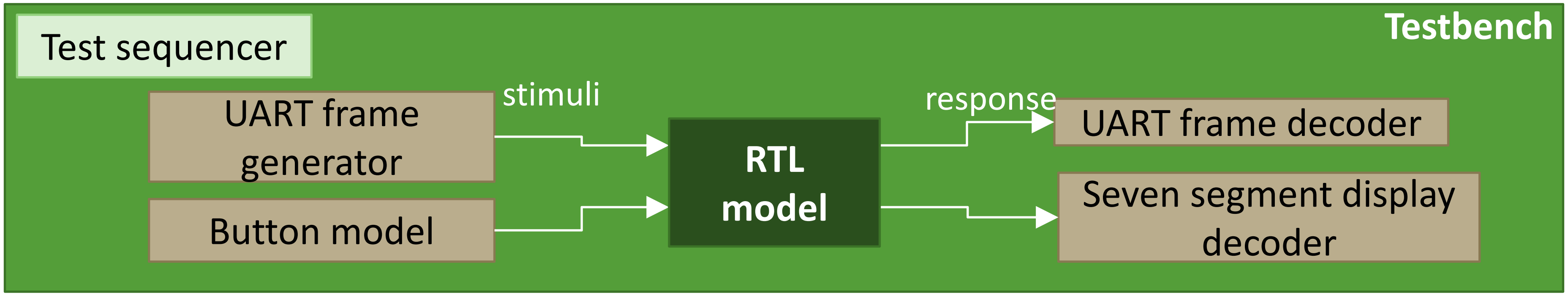
- Interpreting the output data may be difficult
  - Are the seven segment display signal values correct?
  - Are the received UART frame correct (e.g. parity bit)?

```
if tx /= '0' then report "Check # 2: TX LSB ERROR";
else report "Check # 2: TX LSB OK"; end if; -- LSB
wait for 8.67 us;
if tx /= '1' then report "Check # 2: TX 2nd bit ERROR";
else report "Check # 2: TX 2nd bit OK"; end if;
wait for 8.67 us;
if tx /= '0' then report "Check # 2: TX 3rd bit ERROR";
else report "Check # 2: TX 3rd bit OK"; end if;
wait for 8.67 us;
if tx /= '1' then report "Check # 2: TX 4th bit ERROR";
else report "Check # 2: TX 4th bit OK"; end if;
wait for 8.67 us;
if tx /= '0' then report "Check # 2: TX 5th bit ERROR";
else report "Check # 2: TX 5th bit OK"; end if;
wait for 8.67 us;
if tx /= '1' then report "Check # 2: TX 6th bit ERROR";
else report "Check # 2: TX 6th bit OK"; end if;
wait for 8.67 us;
if tx /= '1' then report "Check # 2: TX 7th bit ERROR";
else report "Check # 2: TX 7th bit OK"; end if;
wait for 8.67 us;
if tx /= '1' then report "Check # 2: TX MSB ERROR";
else report "Check # 2: TX MSB OK"; end if; -- MSB
```

IMPROVING THE TESTBENCH

Improving the verification environment

- Using the knowledge from 'Systematic Functional Verification' lecture
- Implement reusable verification components
  - UART frame generator and decoder
  - Seven segment display decoder
  - Bouncing button behavior model
- Update the testbench to use the reusable components



UART frame generator and decoder

- Generator requirements to improve reusability
  - Variable number of data bits
  - Odd/even or no parity bit
  - Length of stop bit (not implemented)
  - Configurable baud delay
  - Error injection
- Decoder requirements
  - Variable number of data bits
  - Odd/even or no parity bit
  - Configurable baud delay

Seven segment display decoder

- Decode the display's input
  - '0' means the segment is enabled
  - HEX characters only

```
function seven_segment_decoder(
    data: std_logic_vector(7 downto 0))
  return std_logic_vector is
    variable result: std_logic_vector(3 downto 0);
  begin

    case data is
      when B"11000000" => result := X"0";
      when B"11111001" => result := X"1";
      when B"10100100" => result := X"2";
      when B"10110000" => result := X"3";
      when B"10011001" => result := X"4";
      when B"10010010" => result := X"5";
      when B"10000010" => result := X"6";
      when B"11111000" => result := X"7";
      when B"10000000" => result := X"8";
      when B"10010000" => result := X"9";
      when B"10001000" => result := X"A";
      when B"10000011" => result := X"B";
      when B"11000110" => result := X"C";
      when B"10100001" => result := X"D";
      when B"10000110" => result := X"E";
      when B"10001110" => result := X"F";
      when others => result := (others => 'X');
    end case;

    return result;
  end function;
```

Bouncing button behavior model

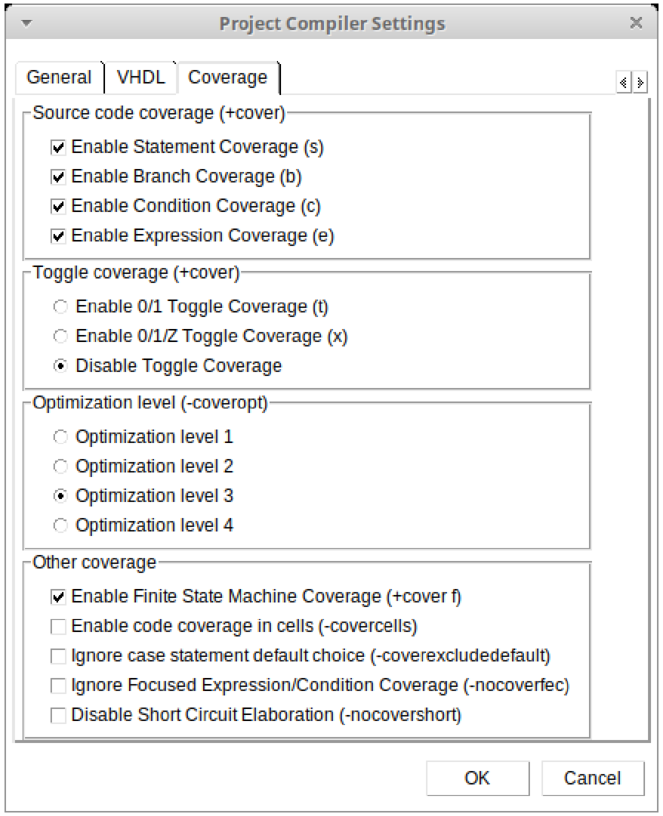
- Configurable timing
  - Period length
  - Number of bouncing

```
procedure bouncing_button(
    period:                  in  time;
    number:                  in  positive;
    signal button:           out std_logic
  ) is

  begin

    for i in 0 to number-1 loop
      button <= not button;
      wait for period;
    end loop;

  end procedure;
```

SIMULATION WITH CODE COVERAGE

Compile the source code

- Compile settings needs to be modified to enable coverage data collection
  - Select the design source files (everything but the testbench and verification package)
  - Right click -> Compile -> Compile properties, Coverage tab



Source code coverage (+cover)
- [✓] Enable Statement Coverage (s)
- [✓] Enable Branch Coverage (b)
- [✓] Enable Condition Coverage (c)
- [✓] Enable Expression Coverage (e)

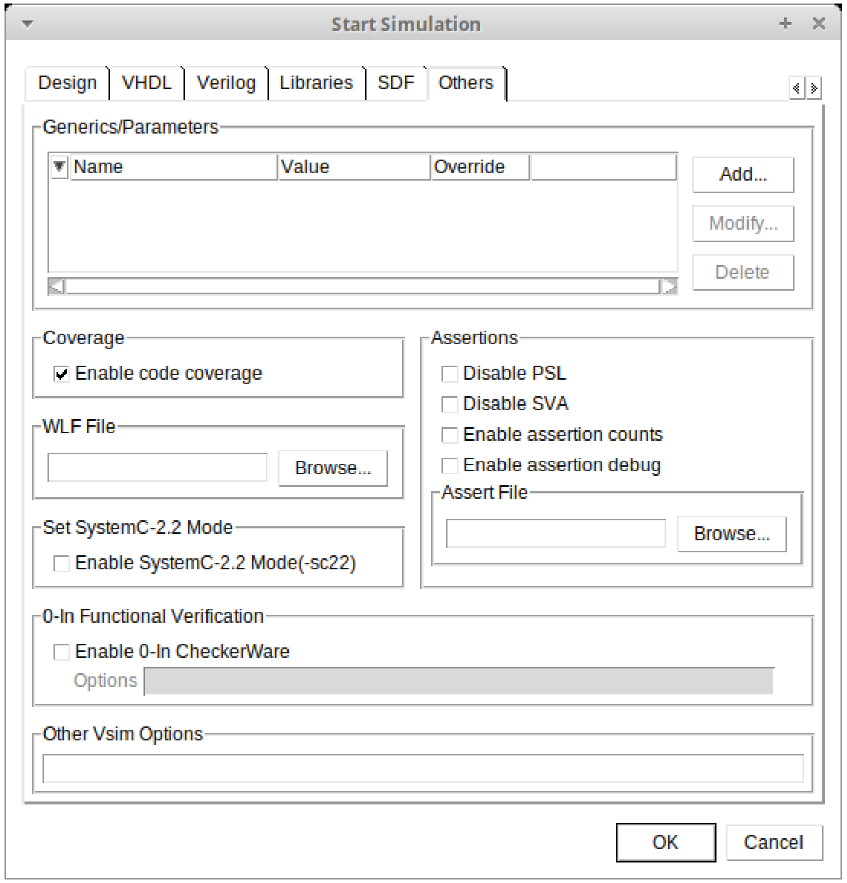
Toggle coverage (+cover)
- ○ Enable 0/1 Toggle Coverage (t)
- ○ Enable 0/1/Z Toggle Coverage (x)
- ● Disable Toggle Coverage

Optimization level (-coveropt)
- ○ Optimization level 1
- ○ Optimization level 2
- ● Optimization level 3
- ○ Optimization level 4

Other coverage
- [✓] Enable Finite State Machine Coverage (+cover f)
- [ ] Enable code coverage in cells (-covercells)
- [ ] Ignore case statement default choice (-coverexcludedefault)
- [ ] Ignore Focused Expression/Condition Coverage (-nocoverfec)
- [ ] Disable Short Circuit Elaboration (-nocovershort)

Starting the simulation

- Start simulation
  - Select the testbench as before
  - On Others tab, coverage collection also should be enabled



Generics/Parameters

Name	Value	Override

Coverage
[✓] Enable code coverage

WLF File

Set SystemC-2.2 Mode
[ ] Enable SystemC-2.2 Mode(-sc22)

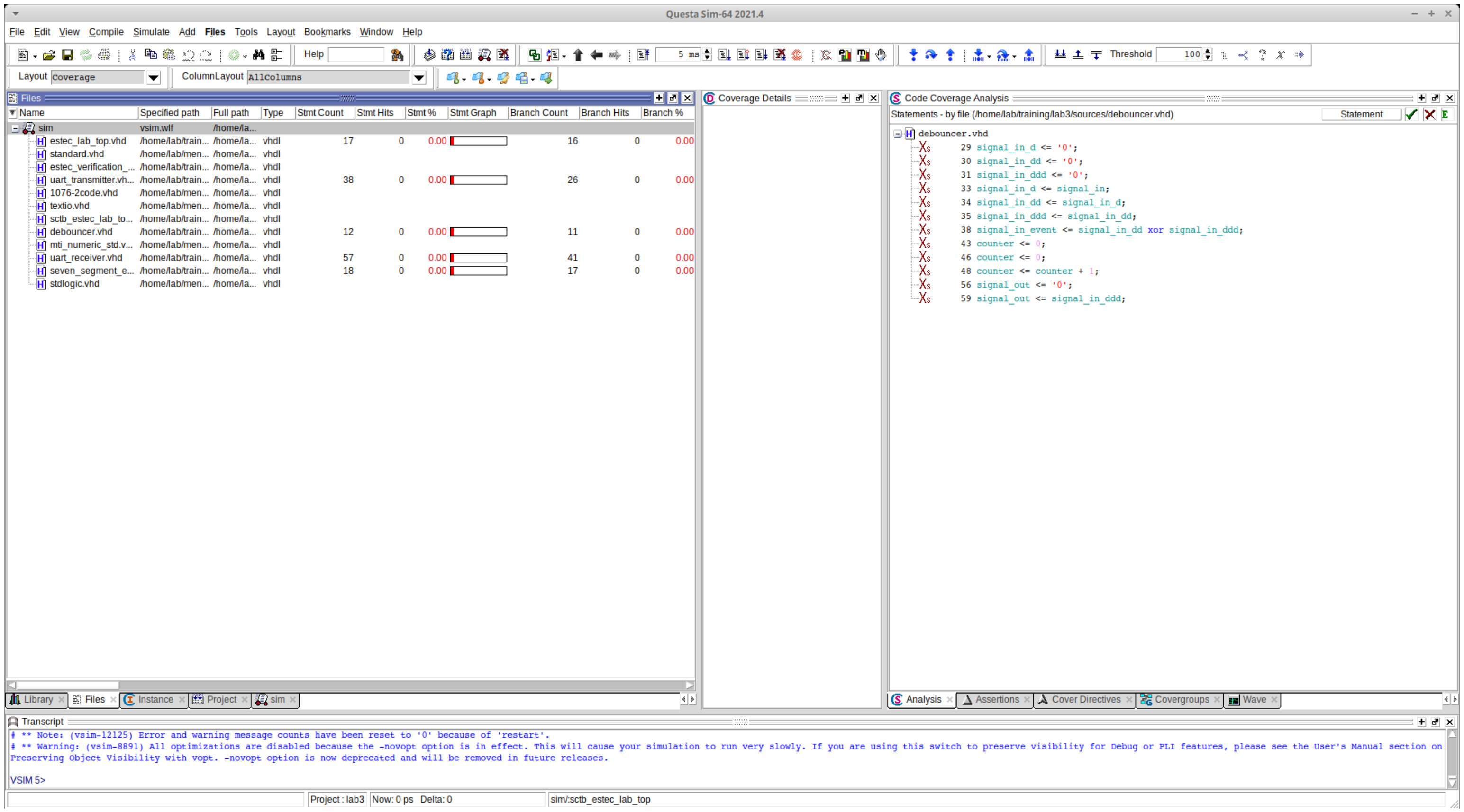
Assertions
[ ] Disable PSL
[ ] Disable SVA
[ ] Enable assertion counts
[ ] Enable assertion debug
Assert File

0-In Functional Verification
[ ] Enable 0-In CheckerWare
Options

Other Vsim Options

Starting the simulation

- After starting the simulation, the window should look similar to this
  - There are some new tabs dedicated to code coverage analysis


Questa Sim-64 2021.4
File Edit View Compile Simulate Add Files Tools Layout Bookmarks Window Help
Help
5 ms
Threshold 100
Layout Coverage
ColumnLayout AllColumns
Files
Coverage Details
Code Coverage Analysis
Statement
Library Files Instance Project sim
Analysis Assertions Cover Directives Covergroups Wave
Transcript

Name	Specified path	Full path	Type	Stmt Count	Stmt Hits	Stmt %	Stmt Graph	Branch Count	Branch Hits	Branch %
sim	vsim.wlf	/home/la...								
estec_lab_top.vhd	/home/lab/train...	/home/la...	vhdl	17	0	0.00		16	0	0.00
standard.vhd	/home/lab/men...	/home/la...	vhdl							
estec_verification_...	/home/lab/train...	/home/la...	vhdl							
uart_transmitter.vh...	/home/lab/train...	/home/la...	vhdl	38	0	0.00		26	0	0.00
1076-2code.vhd	/home/lab/men...	/home/la...	vhdl							
textio.vhd	/home/lab/men...	/home/la...	vhdl							
sctb_estec_lab_to...	/home/lab/train...	/home/la...	vhdl							
debouncer.vhd	/home/lab/train...	/home/la...	vhdl	12	0	0.00		11	0	0.00
mti_numeric_std.v...	/home/lab/men...	/home/la...	vhdl							
uart_receiver.vhd	/home/lab/train...	/home/la...	vhdl	57	0	0.00		41	0	0.00
seven_segment_e...	/home/lab/train...	/home/la...	vhdl	18	0	0.00		17	0	0.00
stdlogic.vhd	/home/lab/men...	/home/la...	vhdl							

Statements - by file (/home/lab/training/lab3/sources/debouncer.vhd)

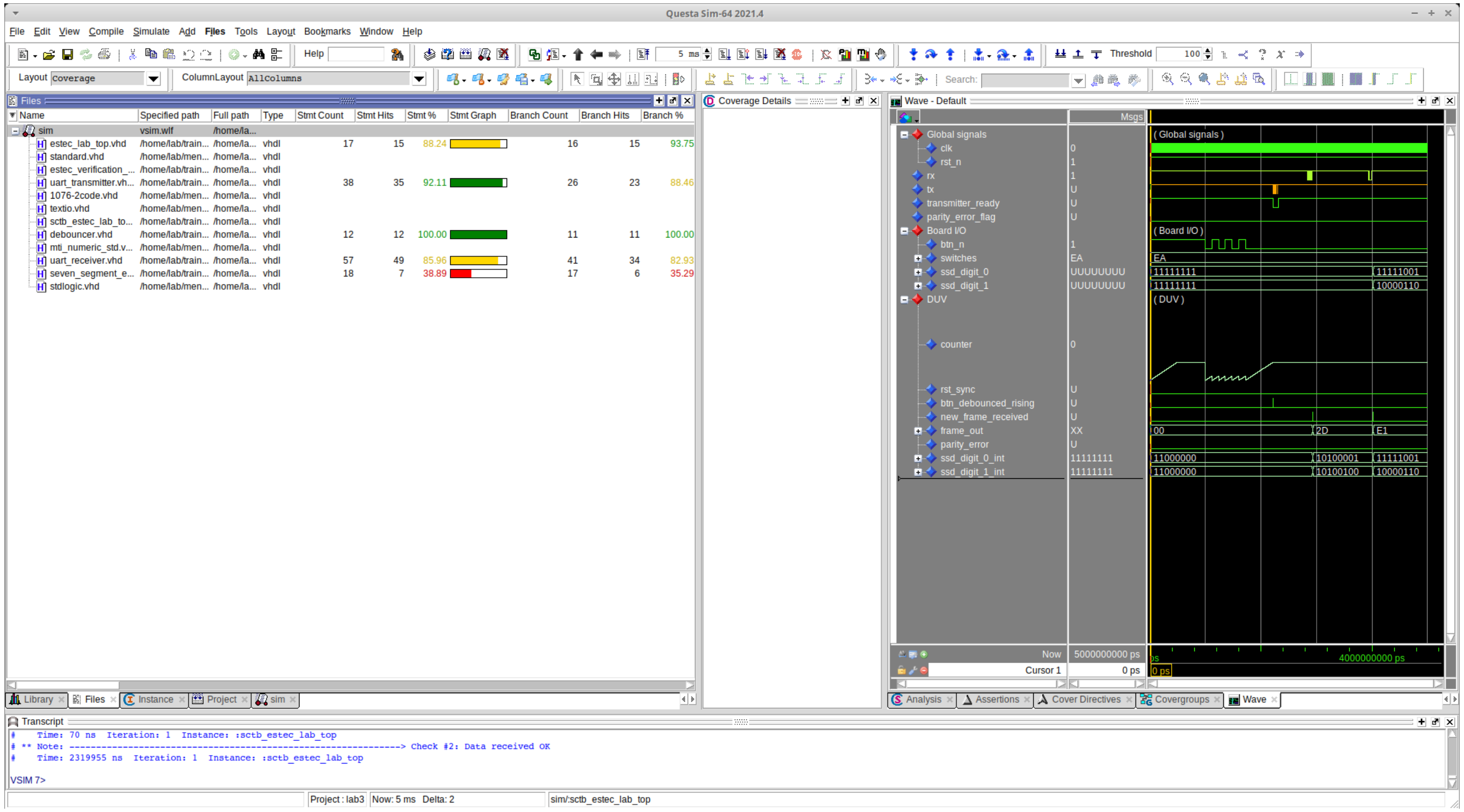
```
debouncer.vhd
  Xs    29 signal_in_d <= '0';
  Xs    30 signal_in_dd <= '0';
  Xs    31 signal_in_ddd <= '0';
  Xs    33 signal_in_d <= signal_in;
  Xs    34 signal_in_dd <= signal_in_d;
  Xs    35 signal_in_ddd <= signal_in_dd;
  Xs    38 signal_in_event <= signal_in_dd xor signal_in_ddd;
  Xs    43 counter <= 0;
  Xs    46 counter <= 0;
  Xs    48 counter <= counter + 1;
  Xs    56 signal_out <= '0';
  Xs    59 signal_out <= signal_in_ddd;
```

```
# ** Note: (vsim-12125) Error and warning message counts have been reset to '0' because of 'restart'.
# ** Warning: (vsim-8891) All optimizations are disabled because the -novopt option is in effect. This will cause your simulation to run very slowly. If you are using this switch to preserve visibility for Debug or PLI features, please see the User's Manual section on Preserving Object Visibility with vopt. -novopt option is now deprecated and will be removed in future releases.

VSIM 5>
```

Project : lab3 Now: 0 ps Delta: 0 sim/:sctb_estec_lab_top

Running the simulation

- Load the wave.do as before to add signals to the waveform
  - File -> Load -> Macro File, select wave.do
- Run the simulation for 5 ms!



Questa Sim-64 2021.4

Files

Name	Specified path	Full path	Type	Stmt Count	Stmt Hits	Stmt %	Stmt Graph	Branch Count	Branch Hits	Branch %
sim	vsim.wlf	/home/la...								
estec_lab_top.vhd	/home/lab/train...	/home/la...	vhdl	17	15	88.24		16	15	93.75
standard.vhd	/home/lab/men...	/home/la...	vhdl							
estec_verification_...	/home/lab/train...	/home/la...	vhdl							
uart_transmitter.vh...	/home/lab/train...	/home/la...	vhdl	38	35	92.11		26	23	88.46
1076-2code.vhd	/home/lab/men...	/home/la...	vhdl							
textio.vhd	/home/lab/men...	/home/la...	vhdl							
sctb_estec_lab_to...	/home/lab/train...	/home/la...	vhdl							
debouncer.vhd	/home/lab/train...	/home/la...	vhdl	12	12	100.00		11	11	100.00
mti_numeric_std.v...	/home/lab/men...	/home/la...	vhdl							
uart_receiver.vhd	/home/lab/train...	/home/la...	vhdl	57	49	85.96		41	34	82.93
seven_segment_e...	/home/lab/train...	/home/la...	vhdl	18	7	38.89		17	6	35.29
stdlogic.vhd	/home/lab/men...	/home/la...	vhdl							

Wave - Default

	Msgs
Global signals	
clk	0
rst_n	1
rx	1
tx	U
transmitter_ready	U
parity_error_flag	U
Board I/O	
btn_n	1
switches	EA
ssd_digit_0	UUUUUUUU
ssd_digit_1	UUUUUUUU
DUV	
counter	0
rst_sync	U
btn_debounced_rising	U
new_frame_received	U
frame_out	XX
parity_error	U
ssd_digit_0_int	11111111
ssd_digit_1_int	11111111

Now 5000000000 ps
Cursor 1 0 ps

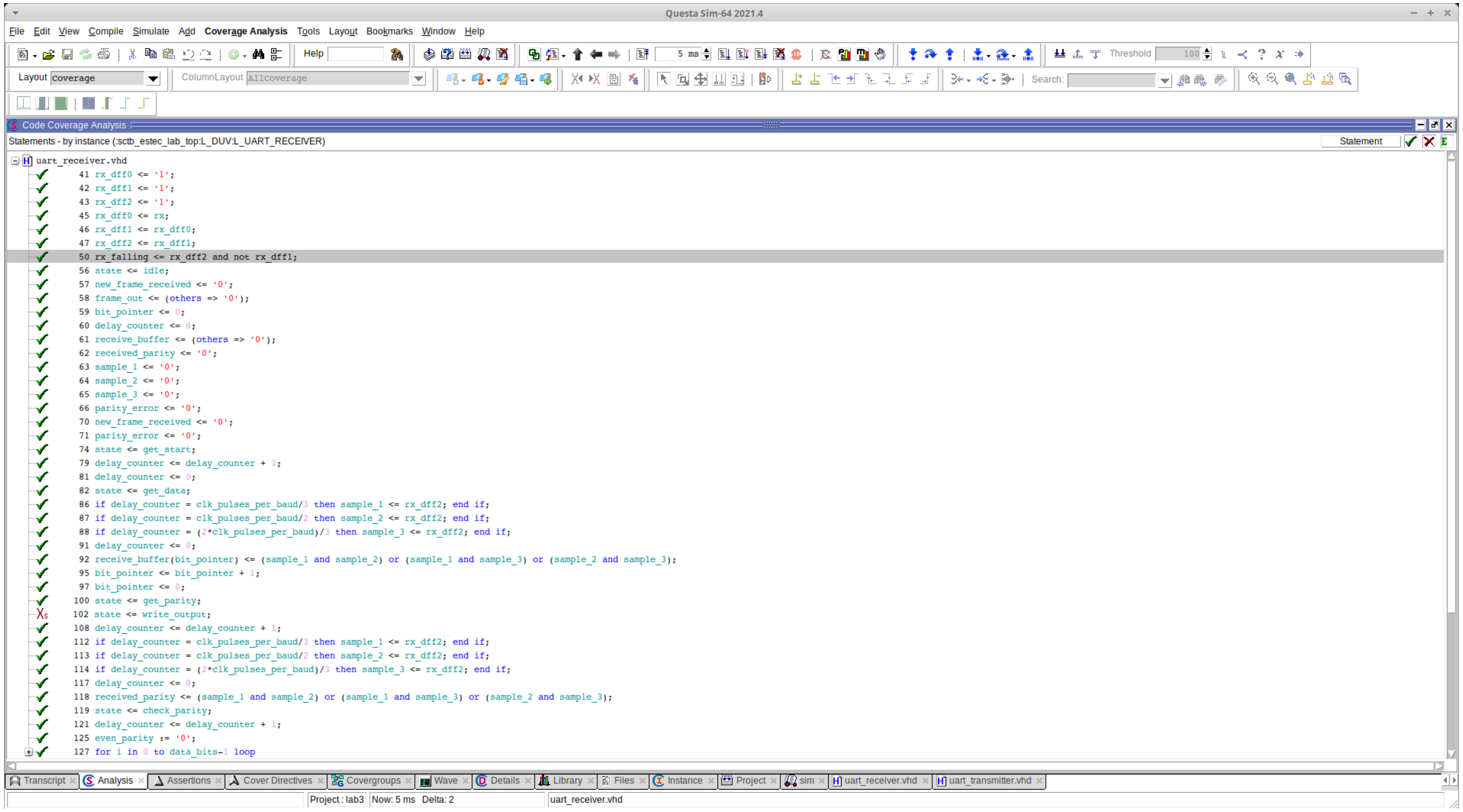
Transcript

```
#    Time: 70 ns  Iteration: 1  Instance: :sctb_estec_lab_top
# ** Note: ------------------------------------------------------------> Check #2: Data received OK
#    Time: 2319955 ns  Iteration: 1  Instance: :sctb_estec_lab_top

VSIM 7>
```

Project : lab3 | Now: 5 ms Delta: 2 | sim/:sctb_estec_lab_top

Evaluating the result

- Check the coverage information in the Files tab
  - By selecting one file, we can examine the coverage in detail in the Analysis tab


Questa Sim-64 2021.4
File Edit View Compile Simulate Add Coverage Analysis Tools Layout Bookmarks Window Help
Layout coverage
ColumnLayout AllCoverage
Threshold 100
Search:
Code Coverage Analysis
Statement
Transcript Analysis Assertions Cover Directives Covergroups Wave Details Library Files Instance Project sim uart_receiver.vhd uart_transmitter.vhd

Statements - by instance (:sctb_estec_lab_top:L_DUV:L_UART_RECEIVER)

```
uart_receiver.vhd
✓   41 rx_dff0 <= '1';
✓   42 rx_dff1 <= '1';
✓   43 rx_dff2 <= '1';
✓   45 rx_dff0 <= rx;
✓   46 rx_dff1 <= rx_dff0;
✓   47 rx_dff2 <= rx_dff1;
✓   50 rx_falling <= rx_dff2 and not rx_dff1;
✓   56 state <= idle;
✓   57 new_frame_received <= '0';
✓   58 frame_out <= (others => '0');
✓   59 bit_pointer <= 0;
✓   60 delay_counter <= 0;
✓   61 receive_buffer <= (others => '0');
✓   62 received_parity <= '0';
✓   63 sample_1 <= '0';
✓   64 sample_2 <= '0';
✓   65 sample_3 <= '0';
✓   66 parity_error <= '0';
✓   70 new_frame_received <= '0';
✓   71 parity_error <= '0';
✓   74 state <= get_start;
✓   79 delay_counter <= delay_counter + 1;
✓   81 delay_counter <= 0;
✓   82 state <= get_data;
✓   86 if delay_counter = clk_pulses_per_baud/3 then sample_1 <= rx_dff2; end if;
✓   87 if delay_counter = clk_pulses_per_baud/2 then sample_2 <= rx_dff2; end if;
✓   88 if delay_counter = (2*clk_pulses_per_baud)/3 then sample_3 <= rx_dff2; end if;
✓   91 delay_counter <= 0;
✓   92 receive_buffer(bit_pointer) <= (sample_1 and sample_2) or (sample_1 and sample_3) or (sample_2 and sample_3);
✓   95 bit_pointer <= bit_pointer + 1;
✓   97 bit_pointer <= 0;
✓  100 state <= get_parity;
Xs 102 state <= write_output;
✓  108 delay_counter <= delay_counter + 1;
✓  112 if delay_counter = clk_pulses_per_baud/3 then sample_1 <= rx_dff2; end if;
✓  113 if delay_counter = clk_pulses_per_baud/2 then sample_2 <= rx_dff2; end if;
✓  114 if delay_counter = (2*clk_pulses_per_baud)/3 then sample_3 <= rx_dff2; end if;
✓  117 delay_counter <= 0;
✓  118 received_parity <= (sample_1 and sample_2) or (sample_1 and sample_3) or (sample_2 and sample_3);
✓  119 state <= check_parity;
✓  121 delay_counter <= delay_counter + 1;
✓  125 even_parity := '0';
✓  127 for i in 0 to data_bits-1 loop
```

Project : lab3 Now: 5 ms Delta: 2 uart_receiver.vhd

Evaluating the result

- What parts of the design were not tested?
  - Open the files, code coverage information is next to the line number
- Is it possible to add more test case to achieve 100% coverage?
  - No, but why?
  - How could we still test those parts?
- We can still improve the coverage, check what functionality was not tested

End of topic

Key concepts

- Complex testbenches utilizes higher abstraction level implementation of the test environment
- Verification components can help improving the productivity
- Verification components can be re-used across projects
- Code coverage can help uncover untested parts of the design, but should not be used as sole metric of the verification process