Systems On-Chip

Lab 2
Verilog analog modelling

Bologna Master Degree in Electrical and Computer Engineering (MEEC)
Instituto Superior Técnico
1st semester, 2nd period

Group 8
David Benjamin Krause | 104898
Duarte Miguel de Aguiar Pinto e Morais Marques | 96523
Eduardo Daniel Roque Martins | 96923

December 9th 2022

1 Introduction [1, 2]

The **battery charger** described in the previous laboratory assignment will be further worked on for this report. In this case, the main objective is to develop a new **power block** design in Verilog and validate it using the Cadence SimVision tool, since the original power block was included in the encrypted file `BATCHARGERpower_64b.vp`.

As mentioned in the previous report, the proposed battery charger topology includes four blocks: `BATCHARGERbg`, `BATCHARGERsaradc`, `BATCHARGERctrl` and `BATCHARGERpower`. The latter (for which the testbench `BATCHARGERpower_64b.vp` is used) controls the charging current according to a digital control signal from the control block - `tc=1` for the trickle current mode (current value configured by 8-bit word `itc`), `cc=1` for the constant current mode (with the 8-bit word `icc`) and `cv=1` for the constant voltage mode (with the 8-bit word `icv`). In terms of the complete charger simulations, a charging profile as in Figure 1 should be obtained. In **constant current (CC)** mode, the battery is charged until the voltage reaches $\mathrm{V_{pre-set}}$, from which the voltage is kept constant and the current decreases exponentially in **constant voltage (CV)** mode. Additionally, **Trickle charge (TC)** mode is used to charge the battery (with a small current) in case the voltage is lower than $\mathrm{V_{cutoff}}$, until it increases above it.

Files	Source
BATCHARGER_64b.v	Available from Lab 1 and used in Lab 2
BATCHARGERbg_64b.v	
BATCHARGERpower_64b.vp	
BATCHARGERsaradc_64b.v	
BATCHARGERctr.v	
BATCHARGERlipo_64b.v	
sim_rtl_all	
BATCHARGER_64b_power_tb.v	New files created/altered for Lab 2
BATCHARGERpower_64b.v	
BATCHARGERpower_64b_tb.v	
sim_rtl_power	
sim_rtl_power_our	
sim_rtl_our_all_power	

Table 1: Files included in directory `/DIGITAL/SIMULATION/BATTERYCHARGER` and used in this laboratory assignment.

In the previous laboratory assignment, the controller was designed based on the diagram shown in Figure 1, which has some modifications regarding the original configuration - in this case, there are no `wait` and `end` modes, the charging process can be reinitiated by returning to `start` and the temperature is continuously monitored.

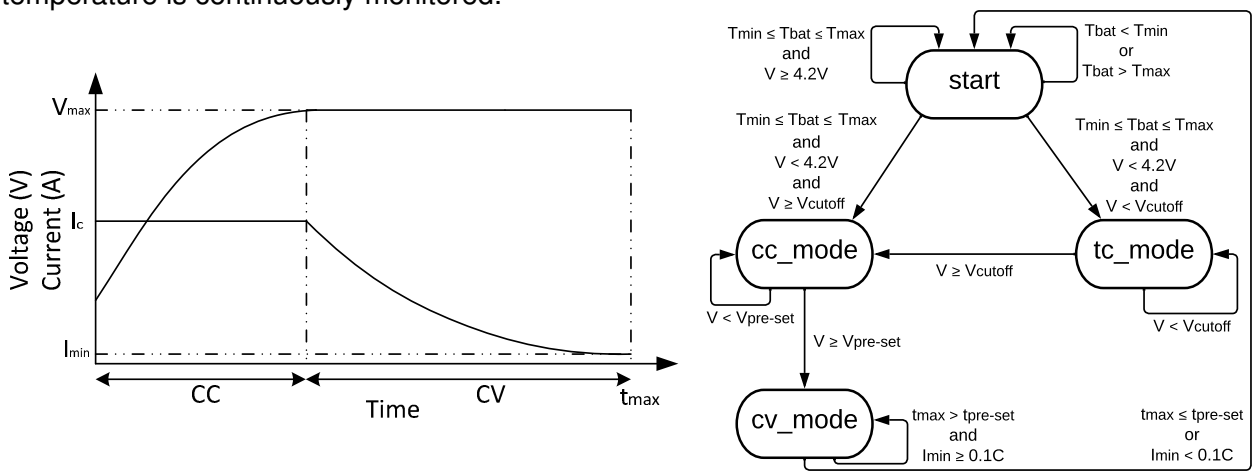

Figure 1: On the left, charging profile of CC/CV (constant current-constant voltage). On the right, diagram used to implement the new controller design in Lab 1 (based on the flow chart shown in [2]).

As seen in Figure 2, the power block is placed between the supply voltage and the battery and controls the battery charging current according to one of the three operation modes mentioned previously. The pins `ibias1u`, `en`, `dvdd`, `dgnd`, `avdd`, `agnd` and `pgnd` are not going to be directly used while designing the new power block.

From the controller module, the values of the parameters `tc`, `cc` and `cv` indicate the current mode, thus the value of the current `iforcedbat` forced at the output, given by the equations shown below. In CV mode, it is also necessary to calculate the values $\mathrm{Vtarget} = 5 \cdot (0.502 \cdot \mathrm{vcv}[7] + 0.251 \cdot \mathrm{itc}[6] + +0.1255 \cdot \mathrm{vcv}[5] + 0.0627 \cdot \mathrm{vcv}[4] + 0.0314 \cdot \mathrm{vcv}[3] + 0.0157 \cdot \mathrm{vcv}[2] + 0.0078 \cdot \mathrm{vcv}[1] + 0.0039 \cdot \mathrm{vcv}[0])$ and $\mathrm{R} = 0.4/(0.5 \cdot \mathrm{C})$. The value of C is defined by `sel[3:0]`, in which the weights of the respective bits are given by 400mAh, 200mAh, 100mAh and 50mAh and an offset of 50mAh is included. Thus,

this value can range from 50mAh to 800mAh (in steps of 50mAh). On the other hand, the binary values `itc[7:0]` and `icc[7:0]` inputted into the power block (as shown in Figure 2) can be used to directly obtain the real values of `iforcedbat` (output current to battery) in the respective modes. The pin `vsensbat`, used for sensing the battery voltage, is used together with `vcv` (which defines the constant target voltage `Vtarget`) and the resistor value `R` to obtain the real value of `iforcedbat` in CV mode, which is usually used in the last part of the charging process. The input voltage `vin` corresponds to the power supply, which needs to be at least 200mV higher than `vsensbat` to allow `iforcedbat>0`.

$$\begin{cases} \mathrm{C}\cdot(0.502\cdot\mathrm{itc}[7]+0.251\cdot\mathrm{itc}[6]+0.1255\cdot\mathrm{itc}[5]+0.0627\cdot\mathrm{itc}[4]+0.0314\cdot\mathrm{itc}[3]+0.0157\cdot\mathrm{itc}[2]+0.0078\cdot\mathrm{itc}[1]+0.0039\cdot\mathrm{itc}[0]) \text{ (TC)} \\ \mathrm{C}\cdot(0.502\cdot\mathrm{icc}[7]+0.251\cdot\mathrm{icc}[6]+0.1255\cdot\mathrm{icc}[5]+0.0627\cdot\mathrm{icc}[4]+0.0314\cdot\mathrm{icc}[3]+0.0157\cdot\mathrm{icc}[2]+0.0078\cdot\mathrm{icc}[1]+0.0039\cdot\mathrm{icc}[0]) \text{ (CC)} \\ (\mathrm{Vtarget}-\mathrm{vsensbat})/\mathrm{R} \text{ (CV)} \end{cases}$$

Finally, it is also necessary to obtain the value of $\mathrm{vbatcurr}$, obtained from `iforcedbat` by a ratio of 1000:1 (to limit the power required by the sensing method) and subsequent multiplication by $(1000 \cdot \mathrm{vref})/\mathrm{C}$, where `vref` is the input of reference voltage. Thus, the charging current is then converted into a voltage $\mathrm{vbatcurr} = (\mathrm{vref}/\mathrm{C}) \cdot \mathrm{iforcedbat}$, which can be externally converted by an ADC and provided to the controller block - which required a measure of the charging current - as a digital word.

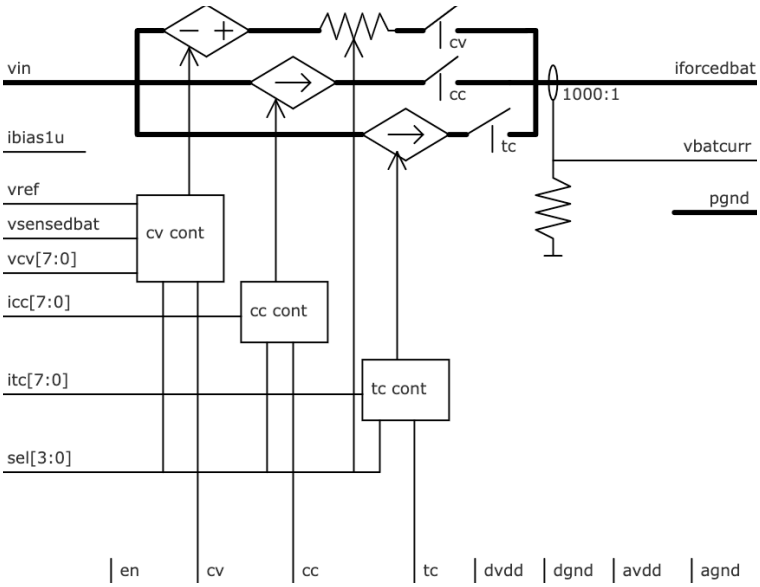

Figure 2: Charger power block diagram included in the respective datasheet [1].

2 Designed power block

Using the same header as in the encrypted file `BATCHARGERpower_64.vp`, the Verilog code with the new power block design was developed in `BATCHARGERpower_64.v`, shown in Figure 3. In this code, according to convention, variables with real values are indicated by `rl_` preceding the name of the respective binary variable. The real value of the controlled forced current at the output (`rl_iforcedbat`) is defined in the first `initial assign` command; in case the enable is `en=1`, its value is given by the expressions shown in section 1 - for each mode (TC, CC and CV), different variables were created (`rl_iforcedbat_tc`, `rl_iforcedbat_cc` and `rl_iforcedbat_cv`, respectively). The current state is given by `cv`, `cc` and `tc` (from the controller block). In case `en=0`, the current value is set to zero. In these expressions, the real value of the capacitance is obtained by properly weighting each bit in `sel[3:0]` and summing the 50mAh offset, as mentioned in section 1. In case of CV mode, `rl_iforcedbat` corresponds to charging with a constant voltage, taking into account the battery's ESR.

In order to "give" and "receive" values from the power block testbench, the proper **signal conversion** must be performed. In this case, the binary values of `vref` and `vsensbat` ("received" from the

testbench) are converted to bits, whereas the real variables `rl_vbatcurr` and `rl_iforcedbat` are converted to bits to be “sent” to the testbench during simulations. No action is performed with some of the variables in this header, such as `avdd` or `dvdd` - as mentioned in section 1 for the respective pins.

```
module BATCHARGERpower_64b (
        output [63:0] iforcedbat, // output current to battery
        output [63:0] vbatcurr, // scaled mirrored current of iforcedbat (1000:1) x R ; R=Vref/C  (C is battery capacity)
        input [63:0]  vsensbat, // voltage sensed
                        // (obtained at the battery as "voltage from iforcedbat integration" + ESR *iforcedbat)
        input [63:0]  vref, // voltage reference (vref = 0.5V)
        input [63:0]  vin, // input voltage; must be at least 200mV higher than vsensbat to allow iforcedbat > 0
        input [63:0]  ibias1u, // reference current     (ibias1u = 1uA)
        input [7:0]   icc, // constant current mode output current value icc=8'b1111_1111 -> iforced = 2A;
                        // ex: icc=8'b11011111 -> iforced = 1.75A (0.5C)
        input [7:0]   itc, //  trickle current mode output current value itc=8'b1111_1111 -> iforced = 2A;
                        // ex: itc=8'b00101100 -> iforced = 0.35A (0.1C)
        input [7:0]   vcv, // constant voltage target value vcv = Vtarget*255/5 = 51*Vtarget
        input         cc, // enables constant current charging mode
        input         tc, // enables trickle  current charging mode
        input         cv, // enables constant voltage charging mode
        input         en, // enables the module
        input [3:0]   sel, // battery capacity selection bits: b[3,2,1,0] weights are 400,200,100,50 mAh + offset of 50mAh
                        // covers the range from 50 up to 800 mAh
        inout [63:0]  avdd, // analog supply to other modules
        inout [63:0]  dvdd, // digital supply to other modules
        inout [63:0]  dgnd, // digital ground
        inout [63:0]  agnd, // analog ground
        inout [63:0]  pgnd // power ground
        ); // enables the module

        parameter IBIASMAX = 1.1e-6;
        parameter IBIASMIN = 0.9e-6; // current limits for ibias1u acceptance
        parameter VINMIN = 3; // minimum vin
        parameter DROPMIN = 0.2; // minimum difference between vin and vsensbat
        parameter VREFMIN = 0.45;
        parameter VREFMAX = 0.55; // voltage limits for vref acceptance

        real rl_vref; // converted value of vref to real
        real rl_imirr; // scaled mirrored current of iforcedbat (1000:1)
        real rl_iforcedbat; // iforcedbat real value
        real rl_vsensbat; // converted value of vsensbat to real
        real rl_vin; // converted value of vin to real
        real rl_ibias1u; // converted value of ibias1u to real
        real rl_resibat;
        real rl_vbatcurr;
        real rl_C; // [Ah] battery capacity
        real rl_Rch; // [Ohm] constant voltage mode resistence: zero current step if Rch = (vcv - vpreset) / icc;
                     // Ex (4.2-3.8)/0.21 = 1.9 Ohm
        real rl_icc;
        real rl_itc;
        real rl_vcv;
        wire supply_ok;
        wire ibias_ok;
        wire vin_ok;
        wire vref_ok;
        real rl_iforcedbat_cc;
        real rl_iforcedbat_cv;
        real rl_iforcedbat_tc;
        real rl_Vtarget;

        // Forced output current depending on mode and enable
        initial assign rl_iforcedbat = en ? (tc ? rl_iforcedbat_tc : (cc ? rl_iforcedbat_cc : (cv ? rl_iforcedbat_cv : 0.0))) : 0.0;

        initial assign rl_C = (sel[3]*400+sel[2]*200+sel[1]*100+sel[0]*50+50)*0.001; // calculate capacitance

        initial assign rl_iforcedbat_tc=rl_C*(0.502*itc[7]+0.251*itc[6]+0.1255*itc[5]+0.0627*itc[4]+0.0314*itc[3]+0.0157*itc[2]+0.0078*itc[1]+0.0039*itc[0]); // if tc=1

        initial assign rl_iforcedbat_cc=rl_C*(0.502*icc[7]+0.251*icc[6]+0.1255*icc[5]+0.0627*icc[4]+0.0314*icc[3]+0.0157*icc[2]+0.0078*icc[1]+0.0039*icc[0]); // if cc=1

        initial assign rl_Vtarget= 5*(0.502*vcv[7]+0.251*vcv[6]+0.1255*vcv[5]+0.0627*vcv[4]+0.0314*vcv[3]+0.0157*vcv[2]+0.0078*vcv[1]+0.0039*vcv[0]); // calculate Vtarget

        initial assign rl_Rch=0.4/(0.5*rl_C); // calculate resistor

        initial assign rl_iforcedbat_cv=(rl_Vtarget-rl_vsensbat)/rl_Rch; // if cv=1

        initial assign rl_vbatcurr=(rl_vref/rl_C)*rl_iforcedbat; // calculate rl_vbatcurr

        // Signal Conversion

        initial assign rl_vref = $bitstoreal(vref);
        initial assign rl_vsensbat = $bitstoreal (vsensbat);

        assign vbatcurr = $realtobits (rl_vbatcurr);
        assign iforcedbat = $realtobits (rl_iforcedbat);

endmodule
```

Figure 3: Verilog code of the new charger power block design, included in `BATCHARGERpower_64b.v`.

In order to validate the power block design by simulation, the script shown in Figure 4 was created. For this purpose, a new testbench (`BATCHARGERpower_64b_tb.v`) was developed and is shown in Figure 6. The simulations ran with this script provided the results shown in Figures 5 (from the console) and 7 (in the waveform window).

```
rm -Rf ../worklib/* ../worklib/.*
##
xmvlog BATCHARGERpower_64b_tb.v BATCHARGERpower_64b.v
xmelab -access +rwc BATCHARGERpower_64b_tb
xmsim -gui BATCHARGERpower_64b_tb &
```

Figure 4: Script created to separately simulate the new power block design with the new power block testbench, using `./sim_rtl_power_our` on the command line.

In the testbench, the system is initially set with `en=1`, but with all parameters `tc`, `cc` and `cv` at zero, thus the output current should remain null, as described in the Verilog code of Figure 3. This is exactly what happens, as obtained from the console (Figure 5). After a short delay of 100ns, the power block is set to CC mode and the current is adequately set to the highest value amongst the three modes, as seen in the console - as described in the charging profile of Figure 1, CC mode leads to the highest (constant) current value, later decreased in CV mode. After another 100ns delay, TC mode is entered, thus the defined `itc` leads to the lowest current value - as described in section 1, TC mode is used to charge the battery with a small current when $\mathrm{V} < \mathrm{V}_{\mathrm{cutoff}}$. Later, in CV mode, the current's value is a bit higher; in this mode, the value of `rl_vsensbat` was changed to 3.5V. It must be taken into account that, when simulating the complete battery charger, `vsensbat` should be lower than `Vtarget`, in order for `rl_iforcedbat` to not reach negative values (due to the equation that defines it in CV mode, as shown in section 1). Moreover, if `vsensbat` is too high, the condition $\mathrm{I} < 0.1\mathrm{C}$ could be verified, thus CV mode is exited, as described in Figure 1.

After these tests, the enable variable is set to zero and it can be seen that the current's value changed to zero, as expected. After this, enable was set to 1 once again and the values of `sel[3:0]` (which defines the capacitance), `icc[7:0]`, `itc[7:0]` and `vcv[7:0]` were altered. With these new values, similar tests were performed; when `cc=tc=cv=0`, the current is null; CC mode corresponds to the highest current value and TC mode to the lowest. In CV mode, `rl_vsenbat` was changed to 4.1V, which is valid since `vcv` (which defines `Vtarget`) was also increased appropriately. After a final delay of 100ns, the simulation ends successfully. In this testbench, most variables included in `BATCHARGERpower_64b.v` are also defined here. In the end of the Verilog code, the proper signal conversions are performed, in order to transfer the values among the scripts (testbench and the power block design). For instance, `rl_vref` is here converted to its binary value of `vref`, in order to be converted in a real value in `BATCHARGERpower_64b.v` to calculate `rl_vbatcurr`. On the other hand, for instance, the binary value of `iforcedbat` (whose real value was calculated in `BATCHARGERpower_64b.v` and then converted to bits) is converted here by `$bitstoreal` in order to test the current value in different circumstances.

With the waveform window of Figure 7, the same expected results can be confirmed and the changes in the variables imposed in the testbench can be clearly noticed alongside the resulting changes in the real values of `rl_iforcedbat` and `rl_vbatcurr`. As expected from the respective equation included in section 1, the changes in the latter are analogous to the changes in the former (they are proportional for a given `vref` and `C`). Despite this, the increases in `rl_vbatcurr` in CC and CV modes are not as significant after the increase in the capacitance value (around 430 000ps).

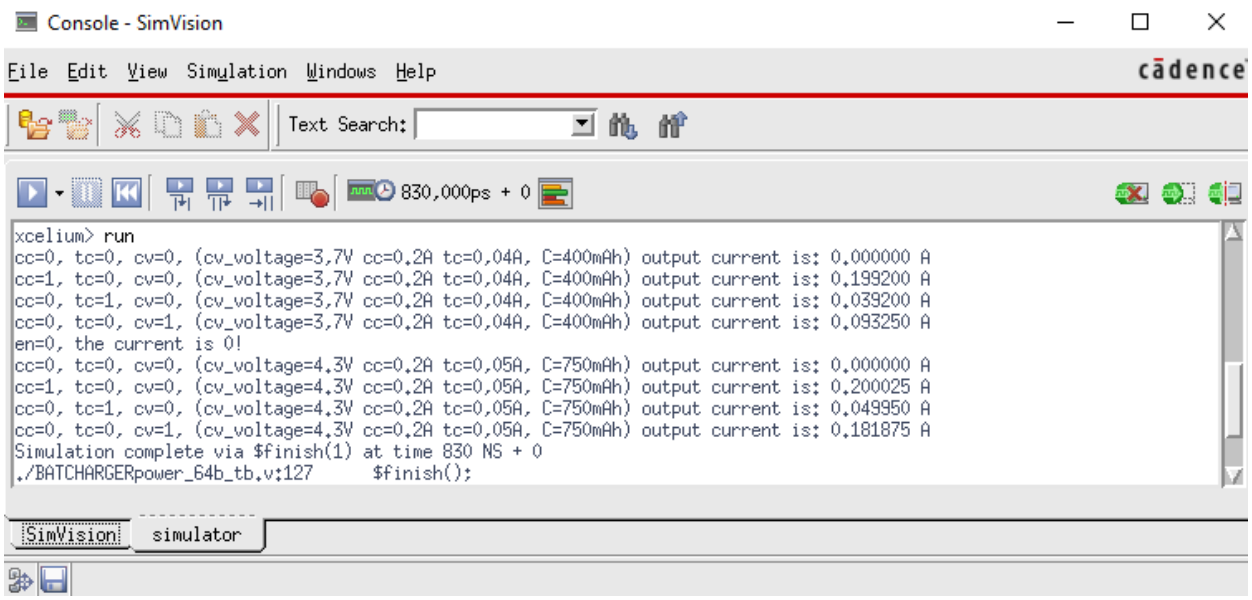

Figure 5: Picture of the messages in the SimVision console obtained by simulating the new power block design described in `BATCHARGERpower_64b.v` with the new testbench `BATCHARGERpower_64b_ourtb.v`.

```
`timescale 1ns / 1ps

module BATCHARGERpower_64b_tb;
   wire [63:0] iforcedbat; // output current to battery
   wire [63:0] vbatcurr; // ibat value scaled 1000:1 * (R=Vref/C)
   wire [63:0] vsensbat; // voltage sensed (obtained at the battery as "voltage from iforcedbat integration" + ESR * iforcedbat)
   wire [63:0] vref; // voltage reference (vref = 0.5V)
   wire [63:0] vin; // input voltage; must be at least 200mV higher than vsensbat to allow iforcedbat > 0
   wire [63:0] ibias1u; // reference current    (ibias1u = 1uA)
   reg [7:0]   icc; // constant current mode output current value icc=8'b1111_1111 -> iforced = C;
                    // ex: icc=8'b01111111 & C=0.4A -> iforced = 0.2A (0.5C)
   reg [7:0]   itc; //  trickle current mode output current value itc=8'b1111_1111 -> iforced = C;
                    // ex: itc=8'b00011001 & C=0.4A -> iforced = 0.04A (0.1C)
   reg [7:0]   vcv; // constant voltage target value vcv = Vtarget*255/5 = 51*Vtarget
   reg         cc; // enables constant current charging mode
   reg         tc; // enables trickle  current charging mode
   reg         cv; // enables constant voltage charging mode
   reg         en; // enables the module
   reg [3:0]   sel;
   wire [63:0] dvdd; // digital supply
   wire [63:0] dgnd; // digital ground
   wire [63:0] avdd; // analog supply
   wire [63:0] agnd; // analog ground
   wire [63:0] pgnd; // power ground
   real        rl_avdd;
   real        rl_dvdd;
   real        rl_agnd;
   real        rl_dgnd;
   real        rl_pgnd;
   real rl_vbatcurr; // vbatcurr real value
   real rl_vref; // converted value of vref to real
   real rl_iforcedbat; // iforcedbat real value
   real rl_vsensbat; // converted value of vsensbat to real
   real rl_vin; // converted value of vin to real
   real rl_ibias1u; // converted value of ibias1u to real
   real rl_icc;
   real rl_itc;
   real rl_vvc;

BATCHARGERpower_64b uut (
                         .iforcedbat(iforcedbat),
                         .vbatcurr(vbatcurr),
                         .vsensbat(vsensbat),
                         .vref(vref),
                         .vin(vin),
                         .ibias1u(ibias1u),
                         .icc(icc),
                         .itc(itc),
                         .vcv(vcv),
                         .cc(cc),
                         .tc(tc),
                         .cv(cv),
                         .en(en),
                         .sel(sel),
                         .avdd(avdd),
                         .dvdd(dvdd),
                         .dgnd(dgnd),
                         .agnd(agnd),
                         .pgnd(pgnd)
);

initial
  begin
     sel[3:0] = 4'b0111; // C=400mAh
     rl_vsensbat = 3.2;
     rl_vref = 0.5;
     rl_vin = 5.0;
     rl_ibias1u = 1.0e-6;
     icc[7:0]=8'b01111111; // constant current mode output current value icc=8'b01111111 -> iforced = 0.5C
     itc[7:0]=8'b00011001;
     vcv[7:0]=8'b10111100; // 8'b10111100 => 3,7V
     cc = 1'b0; // enables constant current charging mode
     tc = 1'b0; // enables trickle  current charging mode
     cv = 1'b0; // enables constant voltage charging mode
     en = 1'b1; // enables the module
     #100 $display("cc=%b, tc=%b, cv=%b, (cv_voltage=3,7V cc=0.2A tc=0,04A, C=400mAh) output current is: %f A", cc, tc, cv, rl_iforcedbat);

     cc = 1'b1; // enables constant current charging mode
     tc = 1'b0; // enables trickle  current charging mode
     cv = 1'b0; // enables constant voltage charging mode
     #100 $display("cc=%b, tc=%b, cv=%b, (cv_voltage=3,7V cc=0.2A tc=0,04A, C=400mAh) output current is: %f A", cc, tc, cv, rl_iforcedbat);

     cc = 1'b0; // enables constant current charging mode
     tc = 1'b1; // enables trickle  current charging mode
     cv = 1'b0; // enables constant voltage charging mode
     #100 $display("cc=%b, tc=%b, cv=%b, (cv_voltage=3,7V cc=0.2A tc=0,04A, C=400mAh) output current is: %f A", cc, tc, cv, rl_iforcedbat);

     rl_vsensbat = 3.5;
     cc = 1'b0; // enables constant current charging mode
     tc = 1'b0; // enables trickle  current charging mode
     cv = 1'b1; // enables constant voltage charging mode
     #100 $display("cc=%b, tc=%b, cv=%b, (cv_voltage=3,7V cc=0.2A tc=0,04A, C=400mAh) output current is: %f A", cc, tc, cv, rl_iforcedbat);

     en = 1'b0; // disables the module
     #30 if(rl_iforcedbat!=0)begin
        $display("The current should be 0 but is %f A",rl_iforcedbat);
        $finish();
     end
     $display("en=0, the current is 0!");

     sel[3:0] = 4'b1110;  // C=750mAh
     icc[7:0] = 8'b01000100;
     itc[7:0] = 8'b00010001;
     vcv[7:0] = 8'b11011011; // 8'b11011011 => 4.3V
     cc = 1'b0; // enables constant current charging mode
     tc = 1'b0; // enables trickle  current charging mode
     cv = 1'b0; // enables constant voltage charging mode
     en = 1'b1; // enables the module
     #100 $display("cc=%b, tc=%b, cv=%b, (cv_voltage=4.3V cc=0.2A tc=0,05A, C=750mAh) output current is: %f A", cc, tc, cv, rl_iforcedbat);

     cc = 1'b1; // enables constant current charging mode
     tc = 1'b0; // enables trickle  current charging mode
     cv = 1'b0; // enables constant voltage charging mode
     #100 $display("cc=%b, tc=%b, cv=%b, (cv_voltage=4.3V cc=0.2A tc=0,05A, C=750mAh) output current is: %f A", cc, tc, cv, rl_iforcedbat);

     cc = 1'b0; // enables constant current charging mode
     tc = 1'b1; // enables trickle  current charging mode
     cv = 1'b0; // enables constant voltage charging mode
     #100 $display("cc=%b, tc=%b, cv=%b, (cv_voltage=4.3V cc=0.2A tc=0,05A, C=750mAh) output current is: %f A", cc, tc, cv, rl_iforcedbat);

     cc = 1'b0; // enables constant current charging mode
     tc = 1'b0; // enables trickle  current charging mode
     cv = 1'b1; // enables constant voltage charging mode
     rl_vsensbat = 4.1;
     #100 $display("cc=%b, tc=%b, cv=%b, (cv_voltage=4.3V cc=0.2A tc=0,05A, C=750mAh) output current is: %f A", cc, tc, cv, rl_iforcedbat);

     $finish();
  end

//-- Signal conversion -----------------

   assign vref = $realtobits (rl_vref);
   assign vsensbat = $realtobits (rl_vsensbat);
   assign vin = $realtobits (rl_vin);
   assign ibias1u = $realtobits (rl_ibias1u);
   assign pgnd = $realtobits (rl_pgnd);
   assign avdd               = $realtobits (rl_avdd);
   assign dvdd               = $realtobits (rl_dvdd);
   assign agnd               = $realtobits (rl_agnd);
   assign dgnd               = $realtobits (rl_dgnd);

   initial assign rl_iforcedbat = $bitstoreal (iforcedbat);
   initial assign rl_vbatcurr = $bitstoreal (vbatcurr);

endmodule // BATCHARGERpower_64b_tb
```

Figure 6: Verilog code of the new power block testbench, included in `BATCHARGERpower_64b_ourtb.v`.

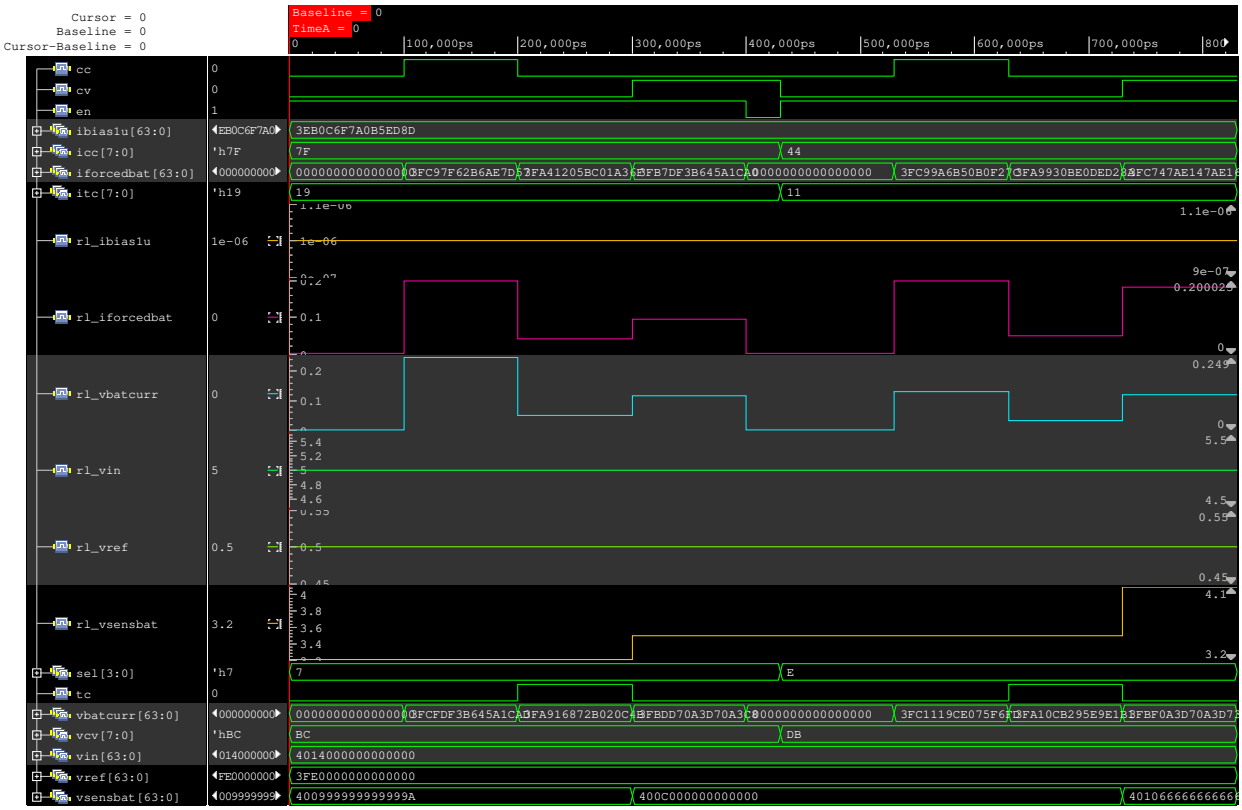

Figure 7: Waveform window obtained by simulating the new power block design described in `BATCHARGERpower_64b.v` with the new testbench `BATCHARGERpower_64b_ourtb.v`, for a total simulation time of 830 000ps.

3 Complete charger with new power block design

To validate the **complete charger** with the new power block model, the testbench `BATCHARGER_64b_power_tb.v` (shown in Figure 11) was created. To verify that the charger operates properly and does not perform the charging process in case of a faulty condition, tests were added to the complete charger testbench initially provided in the subject's web page.

By using the script shown in Figure 8, the results shown in Figures 9 and 10 were obtained. These simulations are run using the new controller design created in Lab 1, having the design of the other blocks not been altered.

```
rm -Rf ../worklib/* ../worklib/.*
##
xmvlog BATCHARGER_64b_power_tb.v BATCHARGER_64b.v BATCHARGERbg_64b.v BATCHARGERctr.v BATCHARGERlipo_64b.v BATCHARGERpower_64b.v BATCHARGERsaradc_64b.v
xmelab -access +rwc BATCHARGER_64b_power_tb
xmsim -gui BATCHARGER_64b_power_tb &
```

Figure 8: Script created to simulate the complete charger with the new power block design, using `./sim_rtl_our_all_power` on the command line.

According to the code implemented in the controller block, the monitoring of the battery temperature is carried permanently during the charging process. In case the battery temperature does not sit inside its normal range, the system should remain in `start` (as shown in Figure 1), thus the charging process does not begin or is stopped. Therefore, in this testbench, after setting `en=1` and defining the capacitance value (along with `rl_vin` and `rl_pgnd`), the initial values of the battery voltage and current are shown in the console. After this, the temperature is increased above the normal range, by defining `vtbat_tb` to 0.4V - this voltage (which should vary between 0V and 0.5V) represents the battery temperature from -40ºC and 125ºC. For this purpose, it is important to notice the new variables `vtbat_tb` and `rl_vtbat_tb` were defined. This was made in order to avoid temperature alterations performed in `BATCHARGERlipo_64b_tb.v`, which receives the value of the previously existing `vtbat`.

As shown in the console window, the battery voltage doesn't increase from its initial value after

0.01s and the current remains null. This means that the commands included in the controller block were successfully applied and the charging process did not occur. After resetting the temperature back to a value within its normal range (20ºC), the charging process was initiated. After 0.01s, the battery voltage increased from $\approx 2.948V$ to $\approx 4.161V$, while the battery current increased to $\approx 0.0195A$.

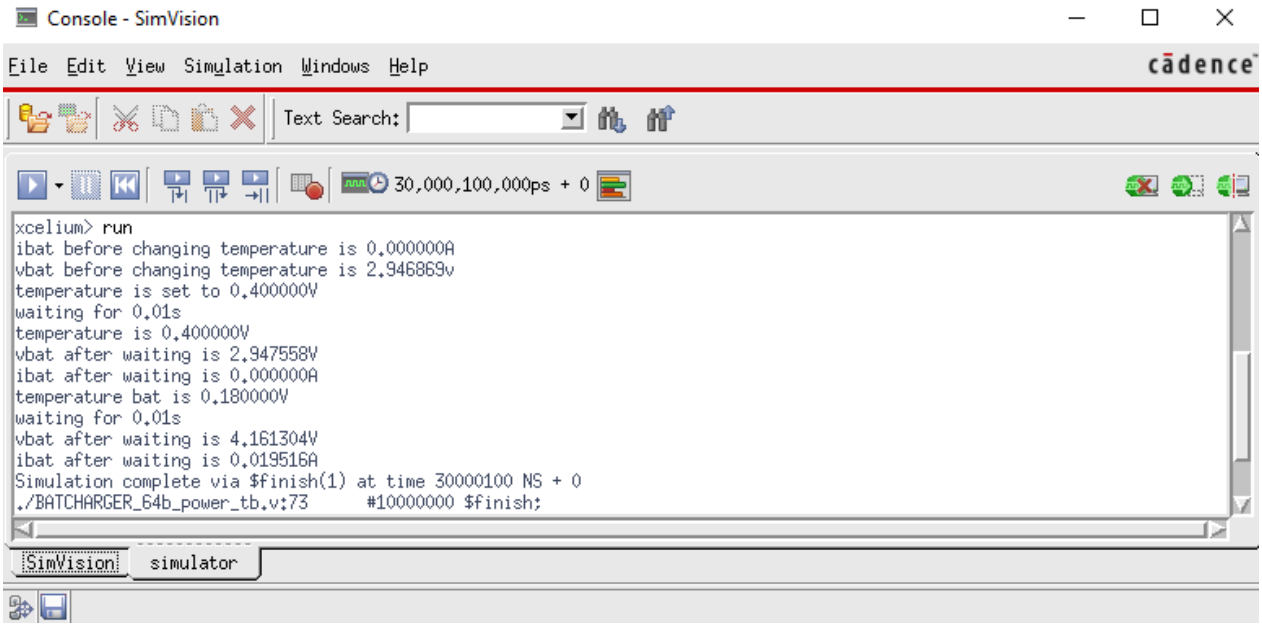

Figure 9: Picture of the messages in the SimVision console obtained by simulating the complete charger with the new testbench `BATCHARGER_64b_power_tb.v`.

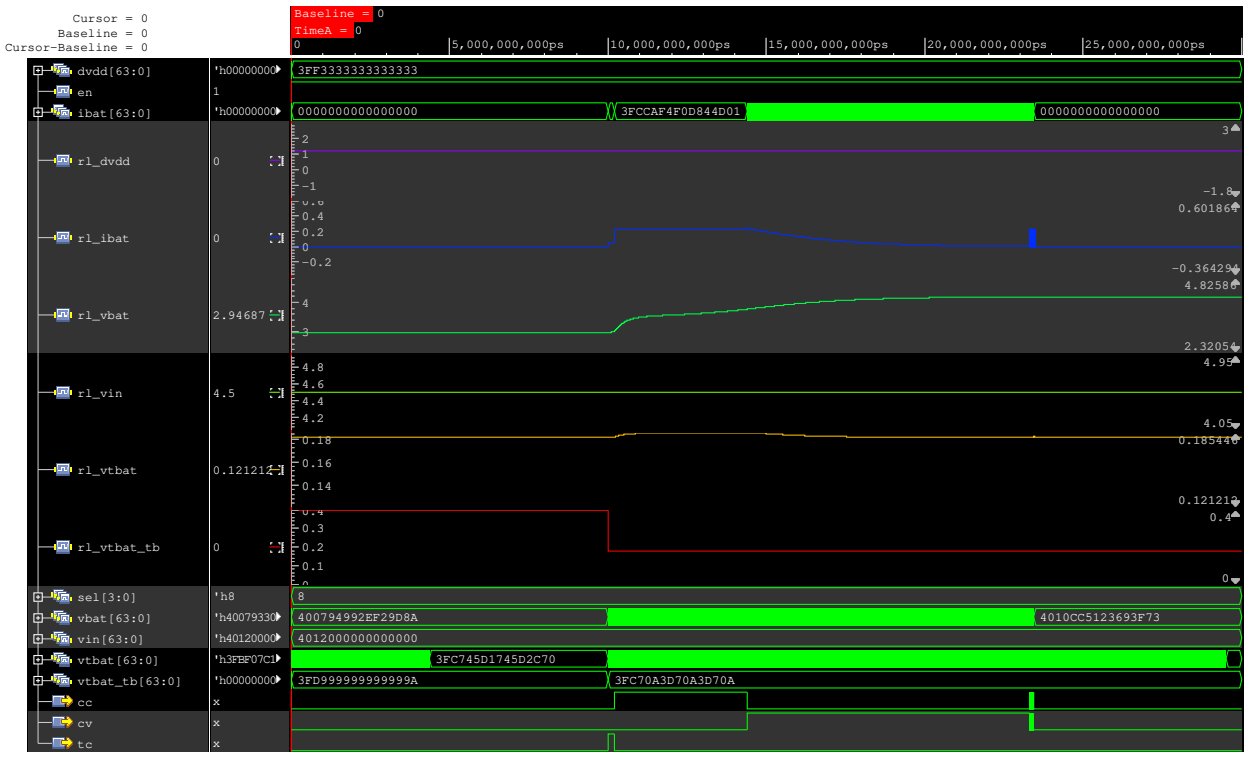

Figure 10: Waveform windows obtained by simulating the complete charger with the new testbench `BATCHARGER_64b_power_tb.v`, for a total simulation time of 30 000 100 000ps.

```
`timescale 1 ns/10 ps

module BATCHARGER_64b_power_tb;

   wire [63:0] vin; // input voltage; must be at least 200mV higher than vsensbat to allow iforcedbat > 0
   wire [63:0] vbat; // battery voltage (V)
   wire [63:0] ibat; // battery current (A)
   wire [63:0] vtbat; // Battery temperature
   wire [63:0] dvdd; // digital supply
   wire [63:0] dgnd; // digital ground
   wire [63:0] pgnd; // power ground
   wire [63:0] vtbat_tb; // avoid that vtbat gets controlled by lipo -> new variable for temperature

   reg         en; // enables the module
   reg [3:0]   sel; // battery capacity selection bits:
                    // b[3,2,1,0] weights are 400,200,100,50 mAh + offset of 50mAh covers the range from 50 up to 800 mAh

   real        rl_dvdd, rl_dgnd, rl_pgnd;
   real        rl_ibat, rl_vbat, rl_vtbat;
   real        rl_vin; // converted value of vin to real
   real        rl_vtbat_tb;

BATCHARGER_64b uut(
                    .iforcedbat(ibat), // output current to battery
                    .vsensbat(vbat), // voltage sensed (obtained at the battery as "voltage from iforcedbat integration" + ESR * iforcedbat)
                    .vin(vin), // input voltage; must be at least 200mV higher than vsensbat to allow iforcedbat > 0
                    .vbattemp(vtbat_tb), // voltage that represents the battery temperature -40ºC to 125ºC -> 0 to 0.5V
                    .en(en), // block enable control
                    .sel(sel), // battery capacity selection bits:
                               // b[3,2,1,0] weights are 400,200,100,50 mAh + offset of 50mAh covers the range from 50 up to 800 mAh
                    .dvdd(dvdd), // digital supply
                    .dgnd(dgnd), // digital ground
                    .pgnd(pgnd) // power ground
);

BATCHARGERlipo lipobattery(
                    .vbat(vbat), // battery voltage (V)
                    .ibat(ibat), // battery current (A)
                    .vtbat(vtbat) // battery temperature
                    );

initial
  begin
     rl_vin = 4.5;
     rl_pgnd = 0.0;
     sel[3:0] = 4'b1000;  // 450mAh selection
     en = 1'b1;

     #100;

     $display("ibat before changing temperature is %fA", rl_ibat);
     $display("vbat before changing temperature is %fv", rl_vbat);

     rl_vtbat_tb = 0.4; // temperature above normal range
     $display("temperature is set to %fV",rl_vtbat_tb);

     $display("waiting for 0.01s");
     #10000000; // wait some time

     $display("temperature is %fV", rl_vtbat_tb);
     $display("vbat after waiting is %fV", rl_vbat);
     $display("ibat after waiting is %fA", rl_ibat);

     rl_vtbat_tb = 0.18; // now set to around 20ºC
     $display("temperature bat is %fV", rl_vtbat_tb);

     $display("waiting for 0.01s");
     #10000000; // wait some time

     $display("vbat after waiting is %fV", rl_vbat);
     $display("ibat after waiting is %fA", rl_ibat);

     #10000000 $finish;

  end

//-- Signals conversion ------------------------------------------------
   initial assign rl_vbat = $bitstoreal(vbat);
   initial assign rl_vtbat = $bitstoreal(vtbat);
   initial assign rl_ibat = $bitstoreal(ibat);
   initial assign rl_dvdd = $bitstoreal(dvdd);
   initial assign rl_dgnd = $bitstoreal(dgnd);

   assign vin = $realtobits(rl_vin);
   assign pgnd = $realtobits(rl_pgnd);
   assign vtbat_tb = $realtobits(rl_vtbat_tb);

endmodule
```

Figure 11: Verilog code of the new complete charger testbench, included in `BATCHARGER_64b_power_tb.v`.

By looking at the waveform window shown above (in which the parameters `cc`, `cv` and `tc` from the controller block were included), the effect of the initial change in temperature can be seen, as the battery current and voltage remain constant. Once `rl_vtbat_tb` changes around 10 000 000ns, the current `rl_ibat` starts to increase, as well as the voltage `rl_vbat`. In this initial phase, the charger is in TC mode, since `tc=1`. After about 200 000ns, the system changes to CC mode (due to high voltage, which surpasses $\mathrm{V_{cutoff}}$), in which the current remains constant at 0.2241A and the voltage increases. Additionally, `rl_vtbat` (temperature) also increases, due to self-heating in the charging process. Finally, once the voltage surpasses $\mathrm{V_{pre-set}}$ (around 14 300 000ns), CV mode

is entered, `rl_ibat` decreases and the change in `rl_vbat` becomes less significant, as expected from the graph in Figure 1 - moreover, the battery temperature change also becomes (much) less significant. Starting at around 23 500 000ns, when the battery voltage stabilizes at around 4.2V and the current at 0A, the system is not in either mode and no charging is being performed, as intended. This happens after a short final and successive alternation between CV and CC modes, due to the loop implemented in the controller design (as explained in the previous report).

4 Conclusion

In this laboratory assignment, all the proposed objectives have been achieved. Having properly analysed the functionality of the power block, as described in its datasheet, a new design was implemented and simulated using SimVision. By doing this, the respective Verilog code was validated using a new testbench, in which changes were forced on several parameters in the design. These changes led to the intended results in the simulation.

Finally, a new testbench was created to test the complete battery charger with this new power block design. Once again, this validated the desired functionality, having a temperature outside the normal range blocked the charging process to occur. When the temperature was re-established within its normal range, the desired charging profile was obtained.

References

[1] Documents available in the subject's Fenix webpage.

[2] Weixiang Shen, Thanh Tu Vo and Ajay Kapoor. "Charging algorithms of lithium-ion batteries: an overview". In: *IEEE Conference on Industrial Electronics and Applications (ICIEA)* (July 2012), pp. 1567–1572.