Spw Codec Error Injection System
User manual

Revision: 1.00
Date: Set/12/2018
Responsable: Cassio Berni

1. Introduction

This work has as result a modified version of Spacewire Light codec (v20130504), with an error injection system. The motivation for that is a Lesia requirement for Simucam, which stands for the Simucam capability to generate exchange level errors, according to ECSS-E-ST-50-12C (july – 2008) spacewire normative specification.

The purpose of this work is to provide a simple user interface for error injection, with minimum modification on original codec code.

The list of errors that the system is capable to generate is:

- Disconnection;
- Parity;
- Escape;
- Credit;
- Char sequence.

The error injection system is applied to the spwstream interface version, with generic tx and rx modules implementation, as used by Simucam.

2. Files/folder structure

The main project folder is "spwerr", with the following subfolders:

- Doc
  - Concepcoes
    - Sistema -> system conception doc, and this manual;
      - Aux_docs -> general documents;
    - Sw -> original spacewire light codec design.
  - Proposta -> commercial terms.
  - Relatorio_atividades -> engineering hours report.
- Sw
  - Spw_codec -> Sigasi Studio project files
    - Doc -> NSEE Simucam vhdl Interface with codec;
    - Ip -> modified codec vhdl code;
    - Sim -> modified codec Modelsim simulation files;
    - Sim_ori -> original codec Modelsim simulation files;
    - Tb -> vhdl testbench files.

3. Concepts

Regarding the exchange level of spacewire protocol, the main vhdl module of codec is spwlink. For supporting it with error injection capabilities, it was created a new module: spwerr, which was instantiated by spwlink as it´s subcomponent. So, spwerr is an internal module of spwlink.

It´s important to note that the error injection system doesn´t depend on spwstream module, which is the codec top level entity. And, because spwerr acts directly on spwlink and spwxmit modules, it has priority to send errors over normal data send requests by spwstream module.

The main idea was to make code changes only inside spwlink, but it was not possible due to some errors nature. So, the codec changed files were:

- Spwpkg -> new types inserted, and spwerr declaration;
- Spwlink -> main changed unit;
- Spwxmit -> some errors demands a direct intervention on tx

module;

- Spwstream & streamtest -> user interface ports insertion.

The error injection user interface is provided it by spwerr module. It generates signals to spwlink and spwxmit modules, as shown in figure 1 below.

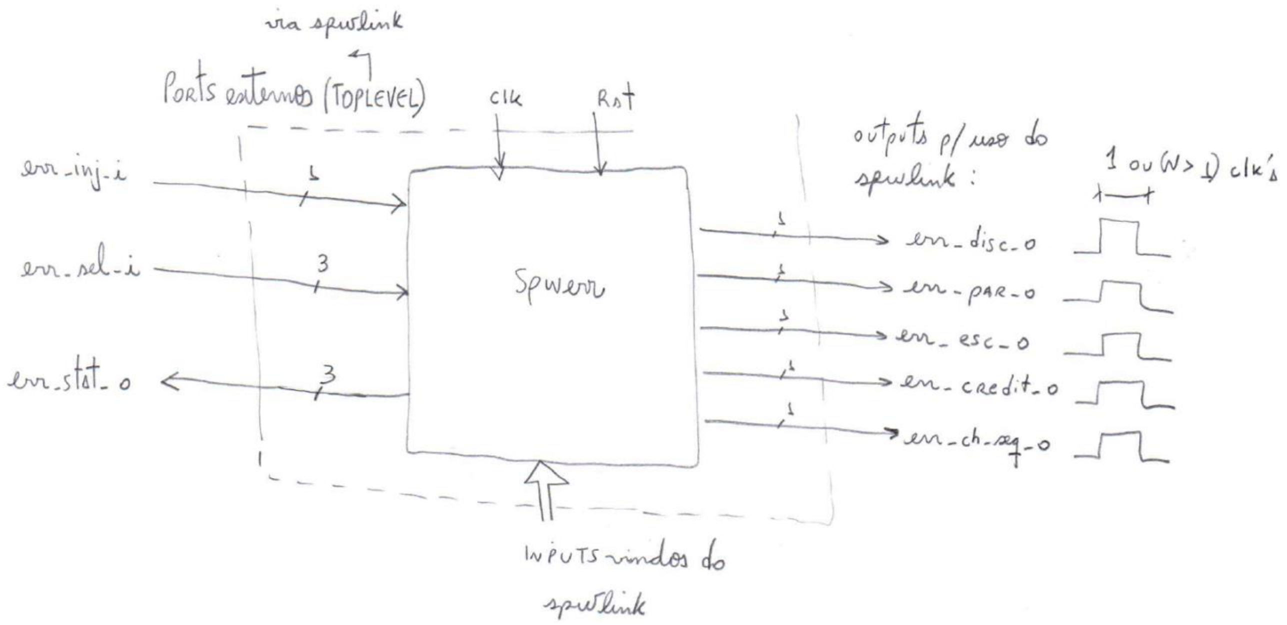

Figure 1 - Spwerr block diagram

The error injection user interface is formed by:

- Err_inj_i -> a rising edge starts error injection, according to error selection (err_sel_i), if certain conditions are satisfied. Err_inj_i is only treated if err_stat_o = stby; only one request is accepted at a time, i.e., if an error injection request is accepted, new ones won´t be until the actual one is finished;

- Err_sel_i -> Error selection. The options are:
  - “000” = Disconnection: emulates a link disable;
  - “001” = Parity: it sends a parity error NULL;
  - “010” = Escape – ESC + EOP – not implemented;
  - “011” = Escape – ESC + EEP – not implemented;
  - “100” = Escape – ESC + ESC;
  - “101” = Credit: It sends 8 x FCT, without local rx_credit increment;

    - "110" = Char sequence: it sends an EOP N-char in started/connection local linl states, without local tx_credit decrement;
    - "111" = Reserved.

- Err_stat_o -> Status output:
    - "000" = Stby. Initial/reset condition. After an error injection request (accepted or not), Spwerr returns to stby only after err_inj_i drops to '0';
    - "001" = Accepted. This indicates that error injection request was accepted (conditions ok) and it´s in progress. Spwerr goes to "ended_ok" through a timer condition;
    - "010" = Invalid selection (esc+eop, esc+eep, reserved);
    - "011" = Inconsistent request. This condition is achieved if user requests disconnection, parity, escape or credit error injection with the local link outside run state. And, similarly, if a char sequence error injection is requested with local link outside started or connecting states;
    - "100" = Ended_ok. Successful error injection concluded;
    - "101" to "111" = reserved.

Each error selected has an stdlogic output port to spwlink module. For each output, the pulse duration is configurable, as it will be seen in item 4.

The spwerr FSM is shown in figure 2.

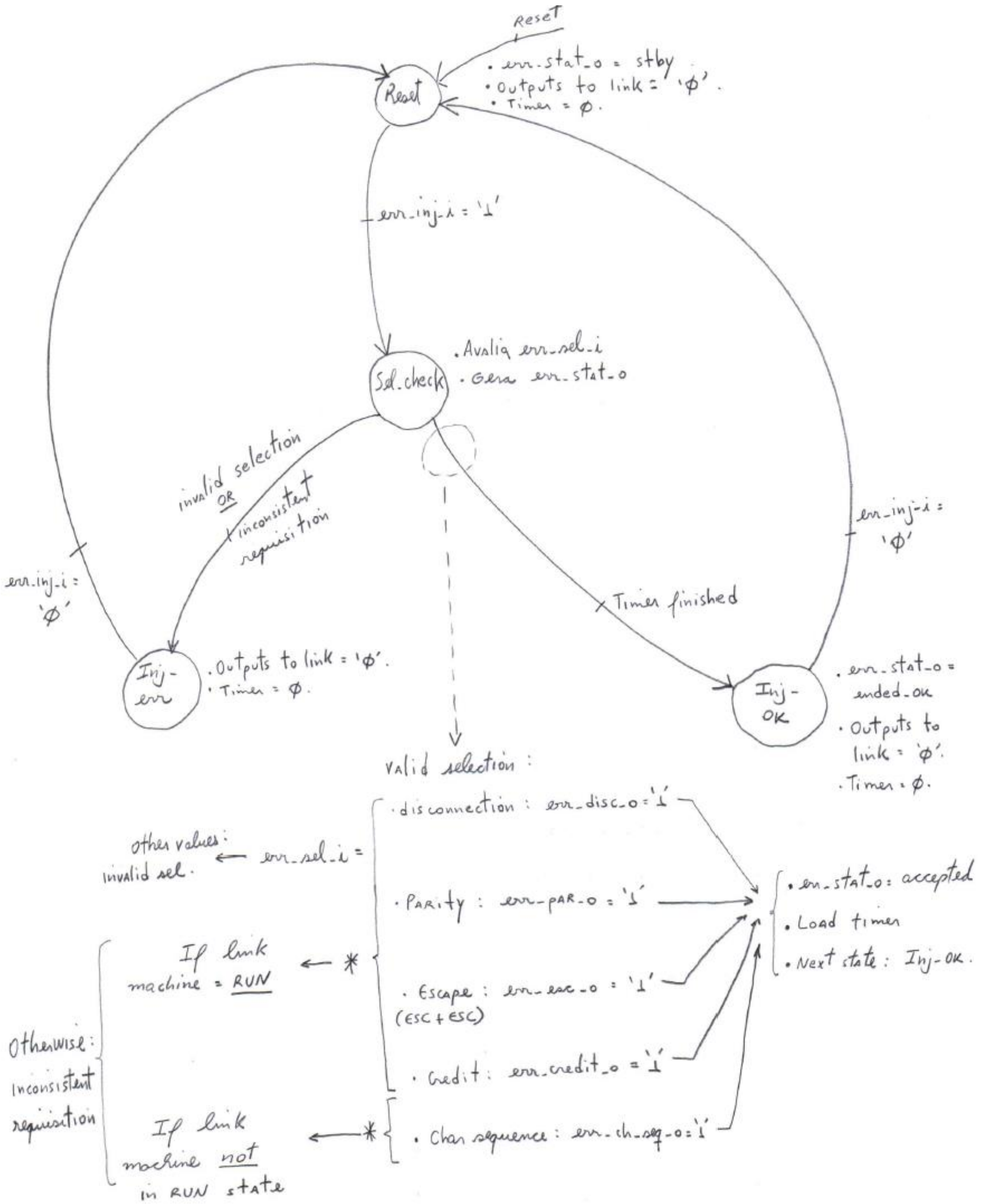

Figure 2 - Spwerr fsm diagram

4.Parameters

In spwerr module, the main parameters refers to ouput pulse duration for each error injection selection. These intervals are crucial, in order to guarantee sufficient time for tx to send the correspondent error packet. If the period is too short, the system may not be able to detect the request, due to previous char been transmited. If it´s too long, multiple error packets could be sent, which is not a problem, but only unnecessary.

It can be noted that after an error char has been sent, remote link will be reseted, which will make the same for local link. Therefore, is not recommended that error output pulses exceeds 6,4 us, for example, which is the interval to link get into error_wait state, after an error_reset condition.

Below is the parameters code in spwerr.vhd. Please note the dependency with Simucam spw codec main clock (actual: 200 MHz) and tx clock (actual: 100 MHz). Changes in this clock values could demand spwerr parameters review as well.

*Outputs to link pulse duration, **in clk cycles** (err inj. accepted -> err inj. ended). The calibration of this times is important. Note that (a + b) values below are refered to: a = actual tx char time, and b = worst case previous tx time.*

The 2x factor is for RUN state errors: clk = 200 MHz, tx_clk = 100 MHz.

The 20x factor is for initialization states: clk = 200 MHz; tx_clk = 10 MHz.

constant **DISC_PULS_TIME** : integer := 1; -- Works fine for 1 clk.

constant **PAR_PULS_TIME** : integer := 2 * (8 + 14); -- null with wrong parity + possible time code

constant **ESC_PULS_TIME** : integer := 2 * (8 + 14); -- (esc+esc) + possible time code

constant **CREDIT_PULS_TIME** : integer := 2 * (32 + 14); -- (8 x fct) + possible time code

constant **CH_SEQ_PULS_TIME** : integer := 20 * (4 + 8); -- eop + possible null (fct is only 4 bits)

Specific transition state duration, in clk cycles (for invalid err_sel and inconsistent request)

```
constant INVALID_SEL_TIME      : integer := 1;
constant INCONSISTENT_REQ_TIME : integer := 1;
```

5. Restrictions / non-implemented features

Disconnection, parity, escape, and credit errors are only generated with local link in run state. Char sequence error are generated with local link is started or connecting states, since they are the only initialization states with tx module enabled.

In the escape error injection, there are three possible frames capable to generate error: esc+esc, esc+eop, and esc+eep. Only esc+esc was implemented in this design version.

6. Simulation

The modelsim simulations were made with spwlink_tb.vhd file. Each error injection test uses a similar piece of code used by original spwlink_tb.vhd simulation, known as "test xx" ("xx" = 1, 2, … N). The wave signals can be opened in modelsim by loading wave_a.do macro file.