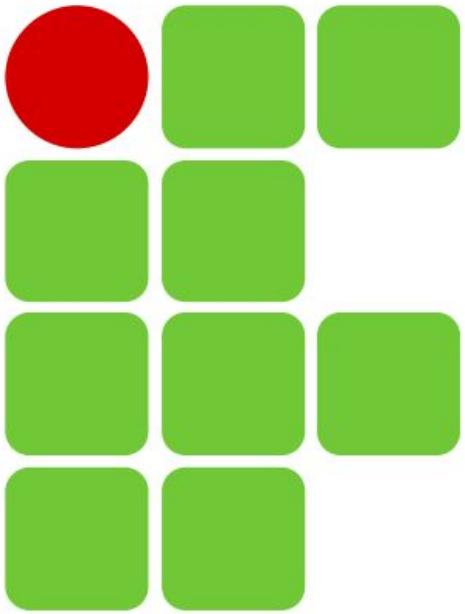


INSTITUTO FEDERAL
SANTA CATARINA
Campus São José

Transmissor Serial de Palavras de Quatro Letras

Disciplina: Dispositivos Lógicos Programáveis I
Professor: Marcos Moecke
Alunos: Alisson Boeing e Guilherme Medeiros

São José, 8 de Julho de 2019

Resumo

Desenvolvimento em FPGA utilizando linguagem de descrição de hardware VHDL de um conversor e transmissor de uma frase com quatro caracteres ASCII. As palavras específicas podem ser selecionadas em uma memória de palavras e tanto a taxa de baud quanto a paridade de envio pode ser selecionadas por chaves. Com a programação pronta, os dados foram testados tanto em placa quanto em simulações em ModelSim e se provaram positivos. O desenvolvimento de um receptor e leitor de serial foi realizado mas este não foi testado nem simulado.

Introdução

Das transmissões serial e paralelo:

Há, basicamente, duas formas de se enviar um dado com mais de um bit de informação. Os bits podem ser enviados todos de uma vez em um vetor de bits ou podem ser enviados bit a bit, para ser reconstruído em um receptor.

Na transmissão em paralelo, um dado de tamanho N é transmitido inteiramente, como um vetor de bits. Neste tipo de transmissão, um meio é necessário para cada bit. Em uma transmissão à fio de um dado de 8 bits em paralelo, são necessários 8 saídas de um transmissor e 8 entradas em um receptor, ou seja, o hardware é muito mais robusto.

Já na transmissão em paralelo de um dado de tamanho N, cada bit é transmitido por vez, tornando o hardware muito mais simples. Apenas uma saída e entrada são necessários no transmissor e receptor, entretanto há a necessidade de se computar essa informação bit a bit: Em apenas 1 bit de um dado de N bits não há informação alguma, ou seja, o dado tem que ser desmontado para a transmissão, enviado e, finalmente, reconstruído para a leitura. É neste ponto que a conversão de paralelo para serial é vital.

Da conversão de Paralelo para Serial:

Considerando um dado em paralelo, um conversor para serial deve:

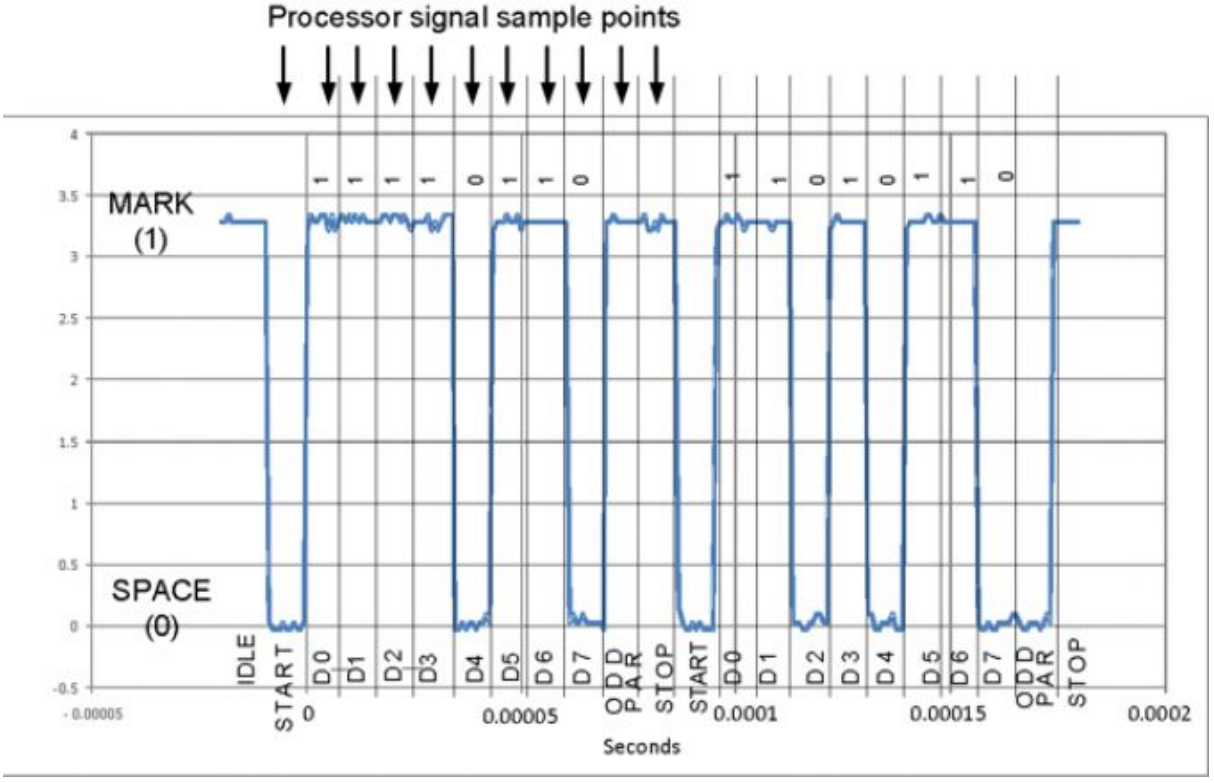
1) Receber o dado completo de uma entrada.
2) Selecionar os bits que constituem a informação.
3) Selecionar com qual sinal os dados devem sair (clock, botão, etc).
4) Enviar os dados, bit a bit, em ordem.

Da comunicação Serial Assíncrona:

Um sistema de comunicação assíncrono respeita um padrão de comunicação [2] explicado na figura 1. À um dado de 7 bits a ser transmitido serialmente, faz-se o processo:

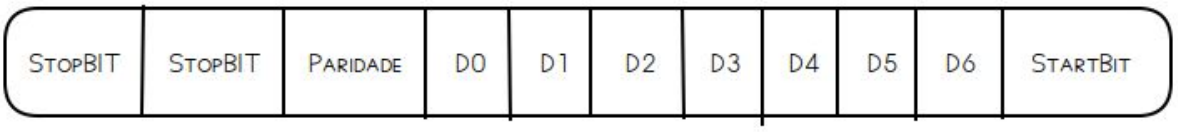
1) Cria-se um buffer com 11 bits.
2) Adiciona-se como bit menos significativo (primeiro bit a ser transmitido) um bit com valor ‘0, sendo este chamado de “*start bit”*, o começo da comunicação de um dado.
3) Adicionam-se dois bits com valor ‘1’ aos dois bits mais significativos, estes são chamados “*stopbit”* e são o fim da comunicação de um dado.
4) Adiciona-se na nona posição um bit de paridade, com intenção de manter a paridade de todo o dado como escolhida por quem o envia.
5) Adiciona-se entre a posição 2 e 8 os sete bits do dado à ser enviado.

O buffer agora está pronto para ser serializado e cada bit pode ser transmitido de acordo com o sinal desejado. Todo o receptor do sistema deve conhecer a modulação e frequência de transmissão dos dados se quiser compreender o conteúdo da mensagem.



(Figura 1: Exemplo de transmissão serial em um dado de 8 bits, fonte: Digilentinc)

A transmissão utilizada neste projeto assume um dado de 7 bits a ser serializado e transmitido. Um dado como este pode ser visto na figura 2.



(Figura 2: Exemplo de transmissão serial de um dado de 7 bits, fonte: Autores)

Especificações do Projeto

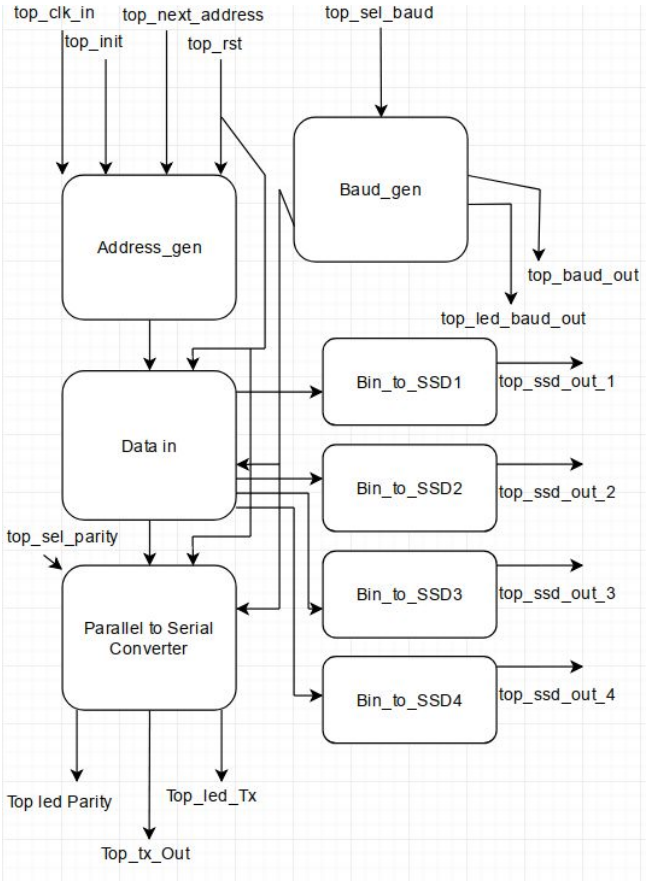
Sobre o sistema:

O projeto constitui no desenvolvimento de um dispositivo que receba uma palavra com quatro caractere ASCII (7 bits) e disponibiliza esse dado de maneira serial, tendo como saída 1 bit por vez.

No desenvolvimento do projeto utilizará programação em VHDL, linguagem de descrição de hardware, e placas FPGA do modelo DE2-115, compatíveis com a linguagem. O sistema deve ser compatível com um sistema de recepção serial, que ao ser ligado na saída do sistema, receberá os dados em serial e os disponibilizará em paralelo. Pode-se descrever o

sistema então como um conversor *parallel to serial.* O sistema em totalidade pode ser explicado pela figura 1, um diagrama da totalidade do projeto com suas saídas e entradas.

O sistema possui um uma entidade top level que liga todos os subcomponentes chamada Serial.vhd.



(Figura 3 - Esquema Total do Projeto, fonte: Autores)

Sobre as entradas e saídas:

- A chave *top_init* (pinado na posição PIN_AB28) que, quando na posição ‘1’, liga todo o sistema. Esta chave possibilita o início e fim da transmissão de dados.
- O botão *top_rst* que retorna o sistema ao ponto zero (pinado na posição PIN_R24) quando na posição ‘1’.
- O botão *top_next_address* (pinado na posição PIN_M23) acessa o bloco Address Gen (descrita totalmente no item 2.3, na descrição do bloco) e, a cada vez que é pressionado um endereço diferente é selecionado para transmissão para o bloco *Data In.*
- O *top_clk_in* (pinado na posição PIN_Y2) é a entrada do sinal de *clock* em 50MHz da placa FPGA DE2-115.
- A entrada *top_sel_parity* é uma chave que seleciona a paridade do dado à ser transmitido, (pinado em PIN_AC25) quando a chave está no nível lógico ‘1’ o dado sairá com paridade par, caso o contrário, sairá com paridade ímpar.

- O *top_sel_baud* é um conjunto de duas chaves (pinadas em PIN_Y23 e PIN_Y24) que acessa o bloco *Baud_gen* alterando a divisão de clock e a saída final do sistema. Com as duas chaves, pode-se escolher entre as quatro posições possíveis.
- A saída *top_baud_out* (pinada em PIN_AB22) é a saída da velocidade da taxa de baud. Ela será usada para passar a taxa de transferência para um outro sistema receptor para sincronizar a transmissão e recepção.
- A saída *top_led_baud_out* é um conjunto de quatro leds (pinados em PIN_F15, PIN_G15, PIN_G16, PIN_H15) que mostram ao usuário qual taxa de transmissão está sendo utilizada.
- O *top_ssd_out1, top_ssd_out2, top_ssd_out3* e *top_ssd_out4,* (pinados de acordo com a tabela 1) são saídas para displays de 7 segmentos, e cada saída se refere à um caractere dos 4 que compõem as palavras da memória em *Data In*.

POS/SSD	SSD1	SSD2	SSD3	SSD4
0	PIN_G18	PIN_M25	PIN_AA25	PIN_AB19
1	PIN_F22	PIN_Y22	PIN_AA26	PIN_AA19
2	PIN_E17	PIN_W21	PIN_Y25	PIN_AG21
3	PIN_L26	PIN_W22	PIN_W26	PIN_AH21
4	PIN_L25	PIN_W25	PIN_Y26	PIN_AE19
5	PIN_J22	PIN_U23	PIN_W27	PIN_AF19
6	PIN_H22	PIN_U24	PIN_W28	PIN_AE18

(Tabela 1 - Pinagem dos SSD descrita nos termos de Posição do Led por SSD utilizado)

- A saída *top_led_tx* (pinada em PIN_E21) é um led que representa a saída de dados em serial, estando aceso quando a saída está no nível lógico ‘1’ e apagado quando a saída está no nível lógico ‘0’.
- A saída *top_led_parity* (pinada em PIN_J17) mostra ao usuário a paridade selecionada pelo *top_sel_parity*, sendo ‘1’ (led aceso) ímpar e ‘0’ (led apagado) par.

A posição na placa onde cada um destes botões, chaves e entradas estão localizados pode ser verificada no datasheet da placa FPGA Cyclone IVE Modelo DE2-115.

Sobre os blocos criados:

- O bloco *Address Gen* gera um endereço de 5 bits. Quando o botão *top_rst* é pressionado, o endereço retorna à “0000”. Cada vez que o botão *top_next_address* é pressionado, é somado ‘1’ ao endereço. Na saída deste bloco um sinal *address_pos* é enviado ao bloco *Data In.*
- O bloco *Data In* recebe a entrada *top_rst* e os sinais *address_pos e baud_data*. O bloco possui uma memória com 23 palavras associadas à uma posição. Esta posição é determinada pelo sinal de entrada *address_pos* vindo do bloco *Address Gen*. Essa posição então determina a palavra que será enviada. Este envio acontece de acordo com o sinal *baud_data*, vindo do bloco *Baud Gen*, ele determina a taxa de envio do bloco *Parallel to Serial Converter* (PSC) e, consequentemente, o tempo de envio de

cada letra do *Baud Gen* para o PSC. O funcionamento é sincronizado para quando o PSC tiver enviado todos os bits de um sinal com 11 bits (descrito no item 1 e no item 2.3 na descrição do bloco *Parallel to Serial Converter*) um novo caractere ser enviado pelo *Data In*.

- O bloco *Baud Gen* recebe a entrada *top_clk_in* e faz uma contagem de pulsos do clock de entrada com objetivo de gerar taxas de baud mais lentas. A escolha do valor é feito através das chaves *top_sel_baud*, e foram escolhidos 3 valores comerciais para as taxas de Baud (para chaves nas posições "00", "01" e "10") além da taxa de 1 segundo para chaves na posição "11".
- O bloco *Parallel to Serial Converter* recebe caracter por caracter do bloco *Data In*, seleciona os 7 bits menos significativos do caractere (o oitavo bit não é utilizado para nenhuma das letras de nenhuma palavra da memória, sendo sempre zero), os adicionam em um buffer com 11 bits da posição 7 à posição 1. Na posição zero é adicionado um *startbit* (com valor '0'), nas posições 9 e 10 são adicionados dois *stopbits* (com valor '1') e na posição 8 é adicionado um bit para tornar a paridade do buffer igual a paridade selecionada na entrada *top_sel_parity*. Este vetor de bits é então serializado, em cada subida do sinal de clock vindo do bloco *Baud Gen*, com o período escolhido, é enviado 1 bit para a saída *top_tx_out*. O mesmo sinal é mandado ao *top_led_tx*, led de indicação de nível lógico '1' na saída.
- Cada um dos blocos *Bin_to_SSD* recebe um caractere diferente dos quatro que foram selecionados no *Data In*. Esse bloco simplesmente decodifica o display de 7 segmentos para cada um dos caracteres possíveis (sendo eles '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'b', 'C', 'd', 'E', 'F', 'H', 'I', 'J', 'L', 'P', 'U') e define todos os leds ligados para qualquer outro valor oriundo de algum problema de funcionamento, como indicativo de erro.

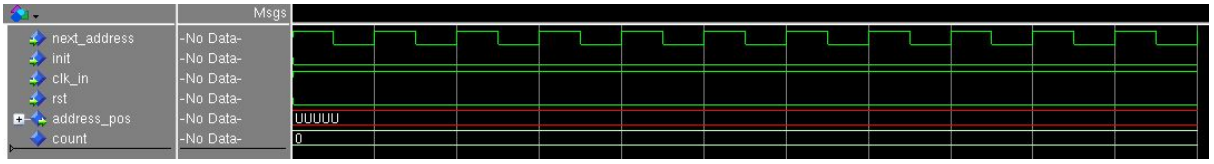
Do equipamento utilizado:

Serão utilizadas duas placas FPGA do modelo DE2-115, uma funcionará como transmissora de bits oriundos da serialização do caractere em ASCII de cada palavra da memória. Toda a informação transmitida e recebida será mostrada nos displays de 7 segmentos que a placa possui. Além disso o projeto só utiliza de chaves e botões da própria placa, explicadas no item 2.2.

Para a construção do software foi utilizado o Quartus, com programação em VHDL, versão 13.0.1, build 232 de 06/12/2013, nas instalações do Instituto Federal de Educação Tecnológica, campus São José.

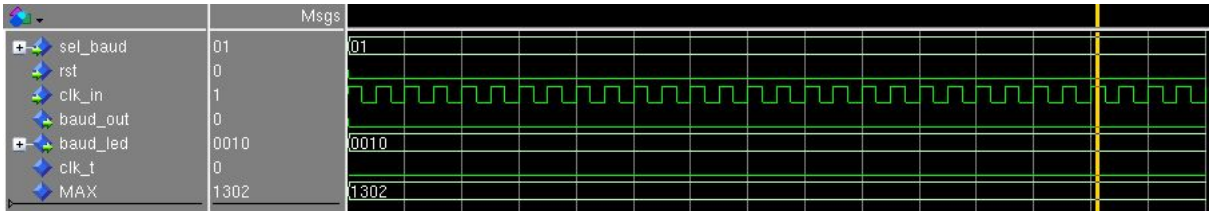
Dos testes realizados e resultados:

Em todos os blocos foi realizado uma simulação utilizando o software ModelSim, listadas abaixo como figuras 4 a 8. As simulações demonstram completo funcionamento do sistema desenvolvido, que utilizou 274 elementos lógicos e 43 pinos da placa FPGA DE2-115.

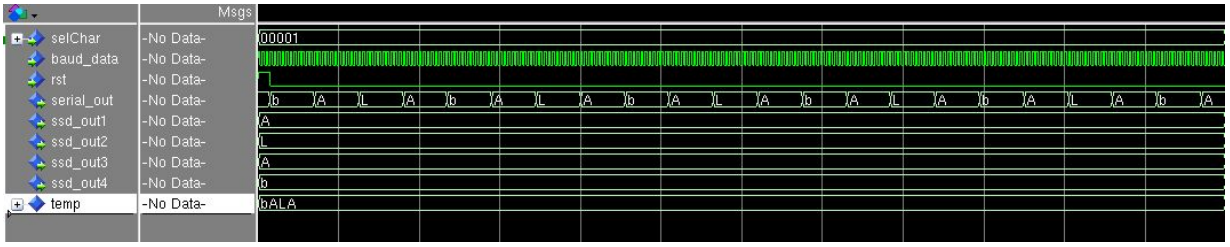


(Figura 4 - Simulação em ModelSim do bloco Address Gen, fonte: autores)

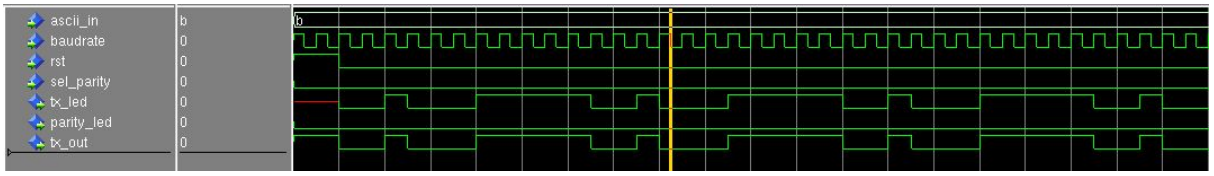
(Figura 5 - Simulação em ModelSim do bloco Bin to SSD, fonte: autores)



(Figura 6 - Simulação em ModelSim do bloco Baud Gen, fonte: autores)

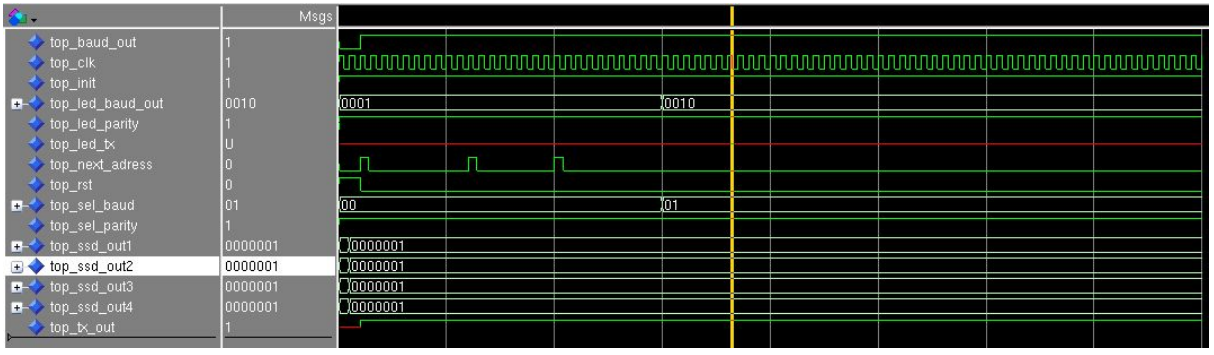


(Figura 7 - Simulação em ModelSim do bloco Data In, fonte: autores)

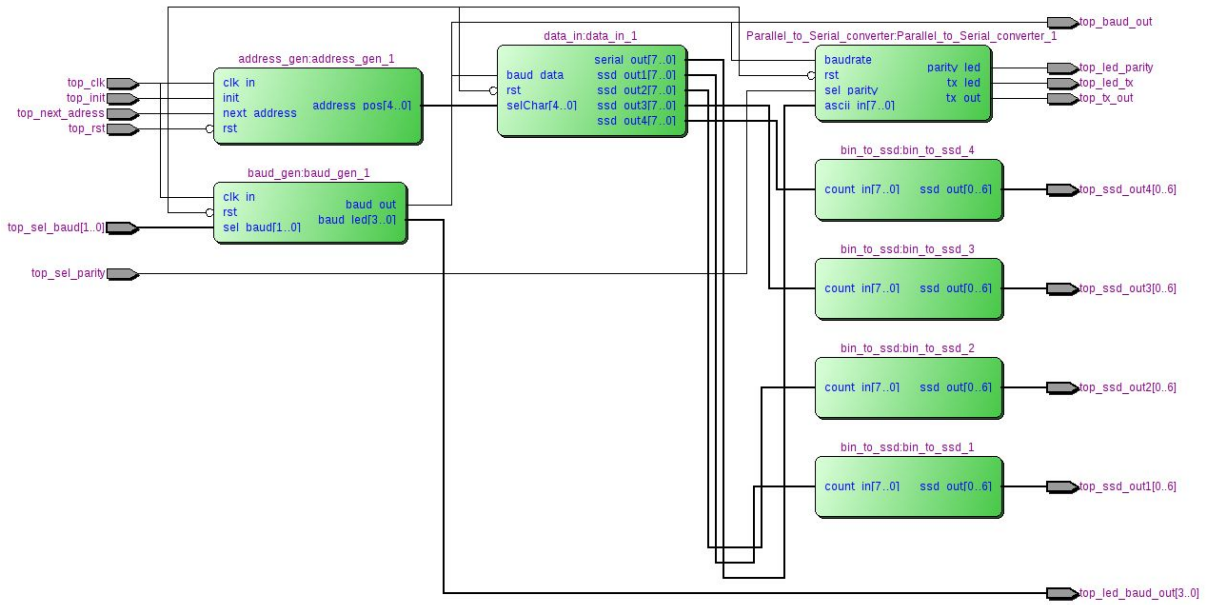


(Figura 8 - Simulação em ModelSim do bloco Parallel to Serial Converter, fonte: autores)

Do top-level, o bloco que liga todos os funcionamentos, foi realizado a simulação em .vht.e



(Figura 9 - Simulação .VHT de todo o sistema)



(Figura 10 - Diagrama RTL gerado pelo Quartus, fonte: Autores).

Conclusão

No desenvolvimento deste trabalho foi estudado a transmissão assíncrona serial com o objetivo de desenvolver um conversor/transmissor de dados ASCII. A implementação foi realizada utilizando VHDL, que se mostrou bastante eficaz para o desenvolvimento de projetos deste cunho, principalmente devido a maneira com que a linguagem permite o acesso a bits específicos de dados dentro do sistema e completa manipulação dos mesmos.

O projeto final utilizou 274 elementos lógicos da FPGA DE2-115, menos de 0,23% do potencial da placa, o que mostra que o mesmo poderia ter sido feito em placas mais enxutas e baratas. Os testes em placas menores não foram realizados, entretanto, devido a falta de disponibilidade no IFSC-SJ.

Este relatório detalha o desenvolvimento e implementação de um sistema de transmissão assíncrona serial de palavras em ASCII com 4 letras. Para utilização de palavras maiores e/ou outros tipos de transmissão, mudanças devem ser feitas.

Podem ser considerados trabalhos futuros a serem realizados a simples recepção e análise desses dados transmitidos serialmente, análise de sincronia neste tipo de projeto e criação de um sistema genérico, com palavras com número de caracteres diferente de quatro.

Referências:

[1] Digilentic, em:
<https://reference.digilentinc.com/learn/courses/unit-4-2/start>, acessado em 08 de Julho às 16:45