INSTITUTO FEDERAL DE SANTA CATARINA

Lucas Coelho Raupp
Yago Castro Rosa

AP3
Modificações Timer

São José,
14/04/2023

Objetivo

Esta atividade prática tem como objetivo modificar um projeto de timer existente (timer_new_qsf.qar), fornecido pelo professor, para incluir uma série de funcionalidades adicionais. Na primeira etapa, o projeto será atualizado para a arquitetura *single_clock*, contará com um contador de milésimo de segundo e terá um SDC adicionado. Serão instanciados um bin2bcd e um bcd2ssd para cada contador (milésimos, segundos e minutos). Um script de teste (run_tb_milesimo.do) será criado para testar as novas funcionalidades e assegurar o funcionamento antes de gravarmos na placa.

Na segunda etapa, um PLL será adicionado para mudar a frequência do contador “r_reg” de 50 MHz para 500 KHz. O script do teste anterior será modificado para testar as novas funcionalidades.

Na terceira etapa, os contadores serão modificados para BCD e o componente bin2bcd será removido. Um novo script de teste (run_tb_500KHz_bcd.do) será criado para testar as novas funcionalidades.

Por fim, na quarta etapa, o contador binário será substituído por um contador LFSR e um novo script de teste (run_lfsr_tb.do) será utilizado para encontrar o valor do ciclo desejado. Ao final, a quantidade de elementos lógicos e registradores serão armazenados em uma tabela para cada etapa do projeto e o funcionamento do sistema será apresentado em sala de aula.

Parte 1

O código original realizava a contagem apenas de minutos e segundos, então foi necessário adaptá-lo para que a contagem também incluísse milésimos de segundo:

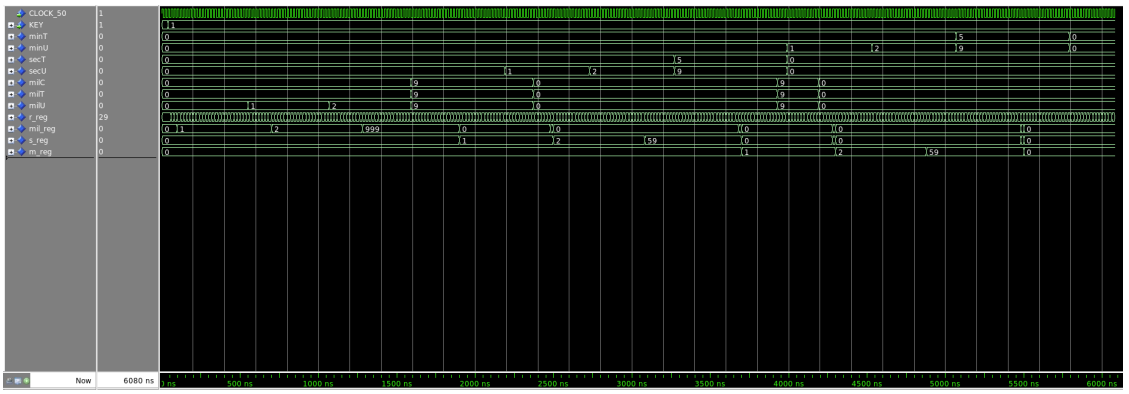
- Declarada a variável “ms” do tipo std_logic_vector em *port* e os sinais “ms_reg” e “ms_next” do tipo unsigned na arquitetura single_clock (que garante o sincronismo). Diferente dos minutos e segundos, os milissegundos são contados até 999, fazendo com que as variáveis contenham 10 bits.
- A virada do clock incrementa os milissegundos ao invés dos segundos, que por sua vez incrementa os segundos quando chega ao valor de 999 + 1. O restante do código permanece o mesmo.
- No código do timer, foi adicionado um novo bin2bcd que utiliza 3 display para mostrar unidades, dezenas e centenas de milissegundos.
- Adicionamos o arquivo SDC disponibilizado pelo professor e criamos uma variável para o clock de 50MHz utilizado nesta etapa.

O projeto foi compilado com sucesso, gerando 285 elementos lógicos e 129 registradores. A frequência obtida em Compilation Report -> Timing Analyzer -> Slow 1200 mv 86C -> Fmax Sumary -> Restricted Max foi de 155,67 MHz.

Para assegurar que as modificações foram feitas corretamente e que o contador está realizando a contagem conforme o esperado, criamos um script de teste que verifica cada transição entre milissegundo, segundo e minuto.

Segue abaixo uma captura de tela do teste realizado comprovando o funcionamento:

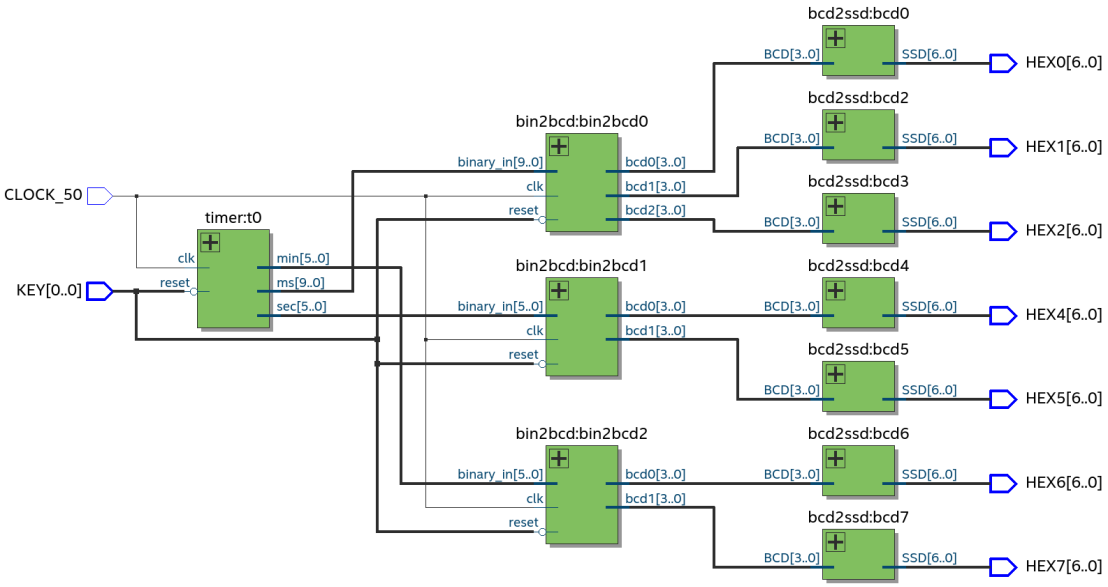
Figura 1 - Gráfico do run_tb_milesimo.do



Autor: De autoria própria.

O RTL Viewer permite a visualização de todas as componentes do projeto e como elas estão conectadas. Segue abaixo o RTL Viewer da parte 1 do projeto:

Figura 2 - RTL Viewer da parte 1.



Autor: De autoria própria.

Parte 2

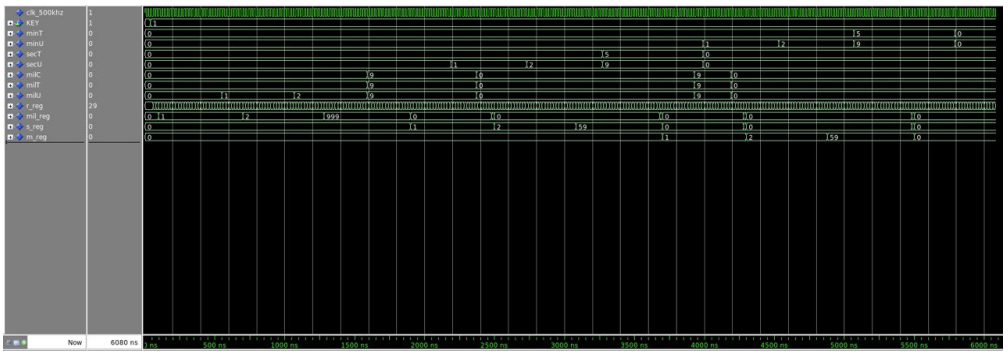
Para esta etapa, mantivemos o que foi feito anteriormente e adicionamos um arquivo PLL que altera a frequência de 50 MHz para 500 KHz. Com o PLL adicionado, modificamos o valor do registrador de 50000 HZ para 500 HZ, uma vez que sem esta alteração os milissegundos teriam a sua contagem proporcional aos segundos após programarmos na placa. Em suma, ao dividir o clock por 100, a contagem voltou a ser condizente com a realidade. Foram realizadas as seguintes modificações no código:

- No timer, “r_reg” foi alterado para que contasse até 499 e depois disso incrementa os milissegundos, seguindo a lógica do novo clock mencionada anteriormente.
- Adicionamos ao projeto um arquivo PLL que recebe um clock de entrada de 50 MHz e o divide por 100, transformando-o em um de 500 KHz.
- Na entidade “top_timer_de2_115”, instanciamos o PLL criado para converter o sinal de 50 MHz em um de 500 KHz. Desta forma, todos os circuitos bin2bcd utilizam este novo clock de 500 KHz.

O projeto foi compilado com sucesso, gerando 250 elementos lógicos e 122 registradores. A frequência obtida em Compilation Report -> Timing Analyzer -> Slow 1200 mv 86C -> Fmax Sumary -> Restricted Max foi de 90,29 MHz. Os resultados são melhores do que os apresentados na etapa anterior, contendo menos elementos lógicos, menos registradores e resultados práticos iguais. Entretanto, a frequência máxima obtida é menor do que a apresentada na etapa anterior, fazendo com que o circuito deva operar em um limite mais contido.

Segue abaixo uma captura de tela do teste realizado comprovando o funcionamento:

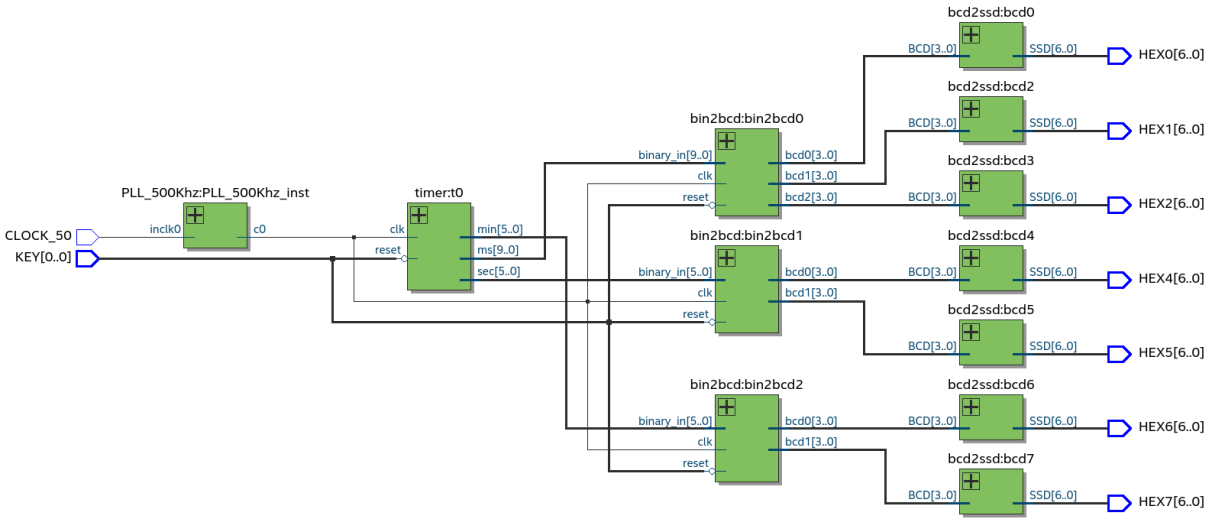
Figura 3 - Gráfico do run_tb_500Khz_bcd.do



Autor: De autoria própria.

Segue abaixo o RTL Viewer da parte 2 do projeto, que desta vez contém um PLL para alterar a frequência do clock:

Figura 4 - RTL Viewer da parte 2.



Fonte: De autoria própria.

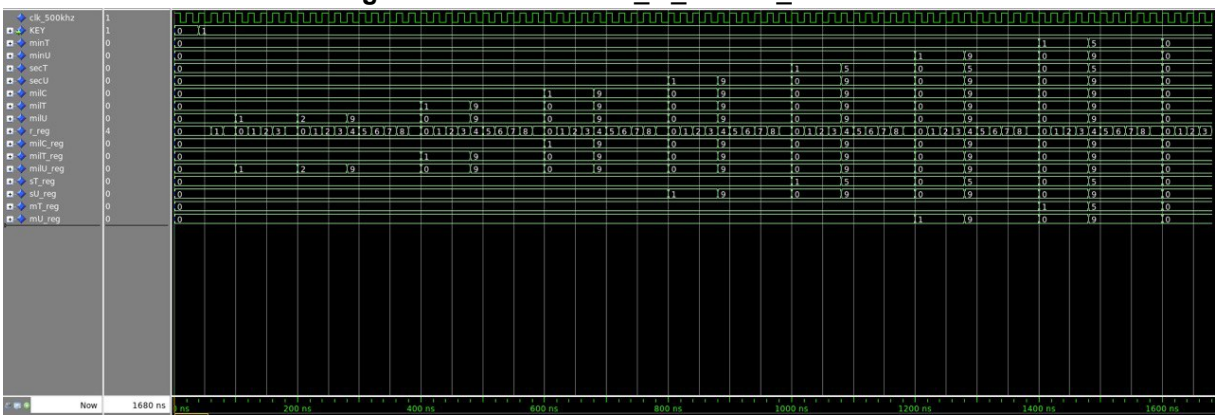
Parte 3

Mantivemos novamente as alterações realizadas no exercício anterior, porém desta vez reduzimos o processo entre a contagem e a exibição no display ao dispensarmos o uso do componente bin2bcd. Para isso, alteramos o “timer” de tal forma que ao invés de realizarmos acréscimos no valor calculado até atingirmos o limite da contagem (59 para segundos e minutos e 999 para milésimos), separamos unidades, dezenas e centenas em contagens separadas. Desta forma, o resultado de cada cálculo foi armazenado em uma variável que nada mais é do que a própria contagem em BCD.

O projeto foi compilado com sucesso, gerando 128 elementos lógicos e 37 registradores. A frequência obtida em Compilation Report -> Timing Analyzer -> Slow 1200 mv 86C -> Fmax Sumary -> Restricted Max foi de 47,94 MHz. A redução do número de elementos lógicos e de registradores é notável se compararmos com os resultados obtidos no exercício anterior, já que uma porção considerável do circuito foi simplificada ao descartarmos o conversor de binário para BCD. Entretanto, mais uma vez, notamos uma queda na frequência máxima de operação.

Segue abaixo uma captura de tela do teste realizado comprovando o funcionamento:

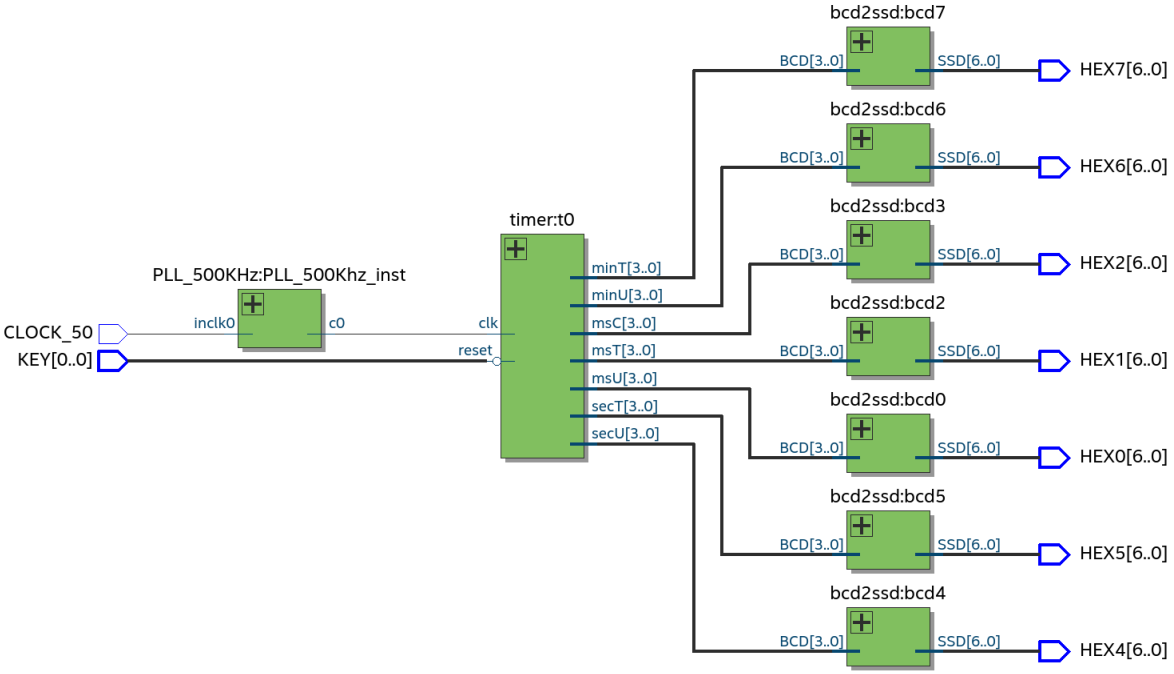
Figura 5 - Gráfico do run_tb_500Khz_bcd.do



Fonte: De autoria própria.

Segue abaixo o RTL Viewer da parte 3 do projeto, que desta vez não necessita das componentes bin2bcd:

Figura 6 - RTL Viewer da parte 2.



Fonte: De autoria própria.

Parte 4

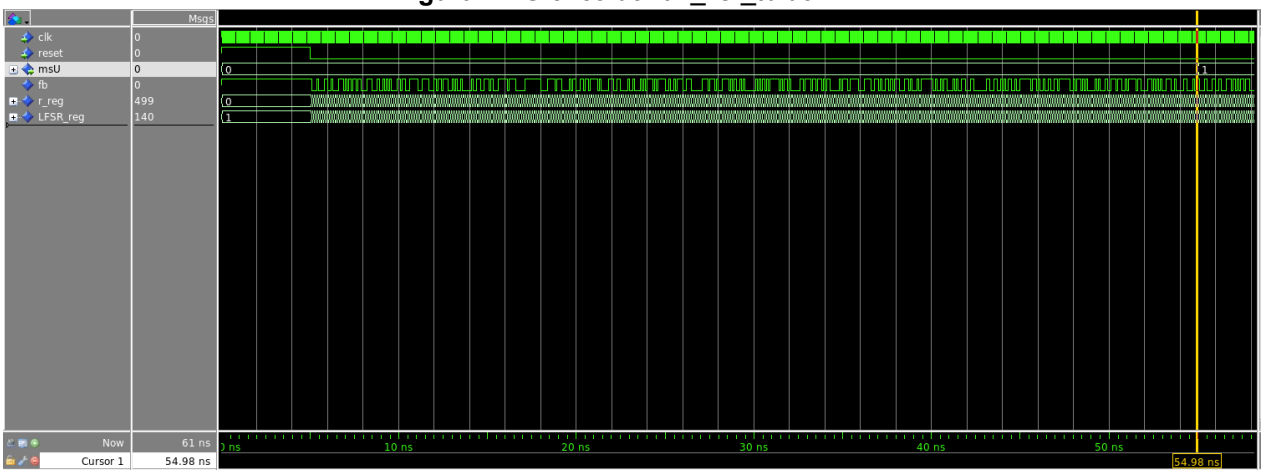
Por fim, substituímos o contador binário utilizado nas etapas anteriores por um contador LFSR, que por sua vez proporcionará uma contagem mais eficiente. Enquanto o contador binário adota uma contagem ordenada entre um valor mínimo e um valor máximo, o contador LFSR utiliza um registro de deslocamento de bits e feedback para gerar uma sequência de números pseudo aleatórios. Essa diferença fará com que o novo método utilize menos flip-flops para realizar a mesma contagem.

Para realizarmos as mudanças necessárias no código, alteramos a classe “timer” removendo o “r_reg” e “r_next” e substituindo-os por “LFSR_reg”, “LFSR_next” e “fb”. Os valores pseudo aleatórios são obtidos por meio de uma “semente” que contém a mesma quantidade de bits do registrador, já o registro de deslocamento pode ser consultado por meio de uma tabela. Utilizamos a tabela de TAPs fornecida pelo professor e concluímos que para uma contagem com 9 bits era necessário realizar o “XOR” entre os bits 5 e 0. A entidade “top_timer_de2_115” permaneceu inauterada.

O projeto foi compilado com sucesso, gerando 118 elementos lógicos e 37 registradores. A frequência obtida em Compilation Report -> Timing Analyzer -> Slow 1200 mv 86C -> Fmax Sumary -> Restricted Max foi de 117,34 MHz. Comparando os resultados deste experimento com os da terceira etapa, notamos que houve uma redução na quantidade de elementos lógicos, já que o método de realizar a contagem foi bastante simplificado, já a quantidade de registradores foi mantida, uma vez que a quantidade de bits a ser contada permanece a mesma em ambas as etapas.

Anexamos a seguir uma captura de tela do teste realizado comprovando o funcionamento:

Figura 7 - Gráfico do run_lfsr_tb.do



Fonte: De autoria própria.

Julgamos desnecessário adicionar o RTL Viewer da parte 4 do projeto, uma vez que ele permanecerá muito parecido com a parte 3. A única alteração está dentro do “timer” que desta vez utiliza um contador LFSR ao invés de um binário.

Considerações Finais

Ao término deste trabalho, concluímos que existem vários artifícios para otimizarmos um projeto em VHDL. Cada diminuição na quantidade de registradores e elementos lógicos torna o circuito mais otimizado e com melhor eficiência energética. Segue abaixo uma tabela que demonstra a mudança nesses parâmetros em cada uma das etapas:

Tabela 1 - Número de elementos lógicos, registradores e frequência máxima em cada uma das etapas do projeto.

	Binário	Binário	BCD	BCD
	500MHz	500KHz	500KHz	500KHz LFSR
LE	285	250	128	120
Registers	129	122	37	37
Freq. (MHz)	155,67	90,29	47,94	73,17

Fonte: De autoria própria.

Para uma análise dos valores dispostos na tabela, verifique o descritivo em cada uma das etapas.