Projeto LSD
Filtro de Média Móvel

Pedro Pinto nº115304

Resumo

O objetivo deste trabalho foi desenvolver um filtro de média móvel, um componente essencial em sistemas de processamento de sinal e imagem. Este filtro serve para eliminar ruídos, resultando num sinal de saída que é uma versão "suavizada"do sinal de entrada. A implementação foi realizada numa FPGA (kit Terasic DE2-115), sendo a saída apresentada num display de 7 segementos e num monitor com interface VGA. O projeto foi estruturado em módulos e foram feitas várias simulações e testes para garantir o correto funcionamento do sistema. Com este trabalho, procurei não apenas desenvolver um filtro de média móvel funcional, mas também compreender e superar os desafios inerentes à sua implementação.

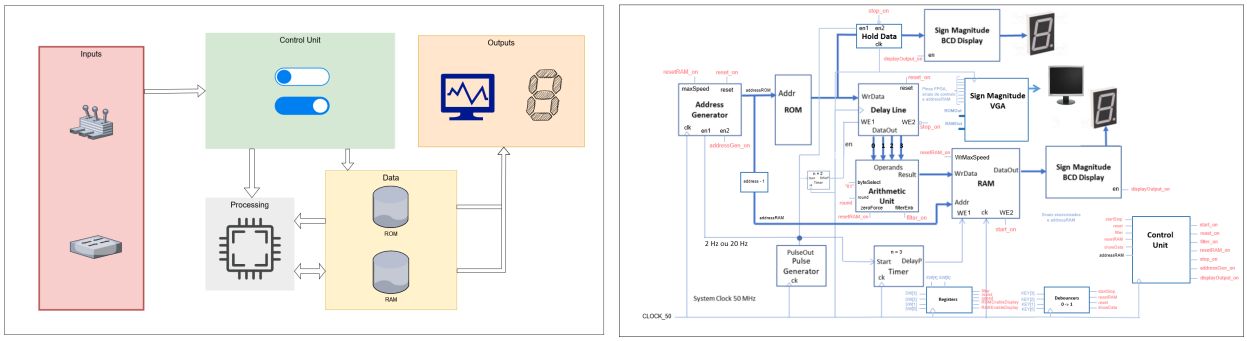
Keywords: *VHDL (Very High Speed Integrated Circuit (VHSIC) Hardware Description Language); Filtro de média móvel; FPGA (Field-Programmable Gate Array); Filtragem de sinal; Eliminação de ruído em processamento digital de sinal*

1. Arquitetura

A arquitetura do sistema é modular e consiste em cinco módulos interligados, conforme ilustrado na Figura 1a. Os módulos são implementados com os componentes apresentados na Figura 1b. De seguida passarei a descrever cada módulo e os seus principais componentes.

Inputs: Este módulo gere os sinais de entrada da FPGA. Utiliza registos sincronizados pelo clock do sistema para sincronizar os sinais de entrada, e debouncers para eliminar ruído em contactos mecânicos, como KEYS, garantindo transições nítidas entre estados.

Data: Este módulo é encarregado de gerir a memória central do sistema, que consiste numa ROM (Read-Only Memory) e numa RAM (Random Access Memory). Ambas têm dimensão de 256x8 bits. A ROM é inicializada com os valores a serem filtrados.



(a) **(b)**

Figura 1. (a) Diagrama de módulos (b) Diagrama de blocos

Outputs: Este módulo gere os sinais de saída da FPGA e divide-se em dois submódulos: a saída nos Displays de 7 Segmentos e a saída VGA. O componente SignMagnitudeBCDDisplay converte vetores de 8 bits (com sinal) para Displays de 7 Segmentos. A saída VGA, por outro lado, faz uso de um componente síncrono,

o SignMagnitudeVGA, que armazena e reproduz internamente os vetores, adaptando-se às proporções do dispositivo de vídeo ligado.

Control Unit: Este módulo coordena o sincronismo do sistema e gere os estados internos através do componente ControlUnitFSM. Este gera os sinais de controlo necessários para o funcionamento adequado de todos os componentes. Componentes adicionais como HoldData, TimerN e PulseGeneratorN são utilizados para regular a temporização de sinais/pulsos.

Processing: Este módulo é encarregado do processamento e alinhamento do sistema de acordo com o seu estado atual. Através do componente AddressGenerator, assegura o endereçamento adequado das memórias. Os valores lidos na memória são direcionados para o DelayLineRegister, que atua como uma linha de atraso em modo serial-in – parallel-out para palavras de 8 bits. Os valores da linha de atraso são processados pelo ArithmeticUnit, que filtra os valores recebidos e disponibiliza o resultado na saída.

2. Implementação

A implementação do sistema seguiu uma estratégia faseada. Optei por dividir a tarefa em partes mais pequenas, concentrando-me num componente específico de cada vez, conforme demonstrado na Figure 2. Apesar de ter planeado todas as tarefas desde o início, estas sofreram pequenas alterações durante a implementação, e surgiram inclusive novas tarefas à medida que o projeto foi sendo refinado.

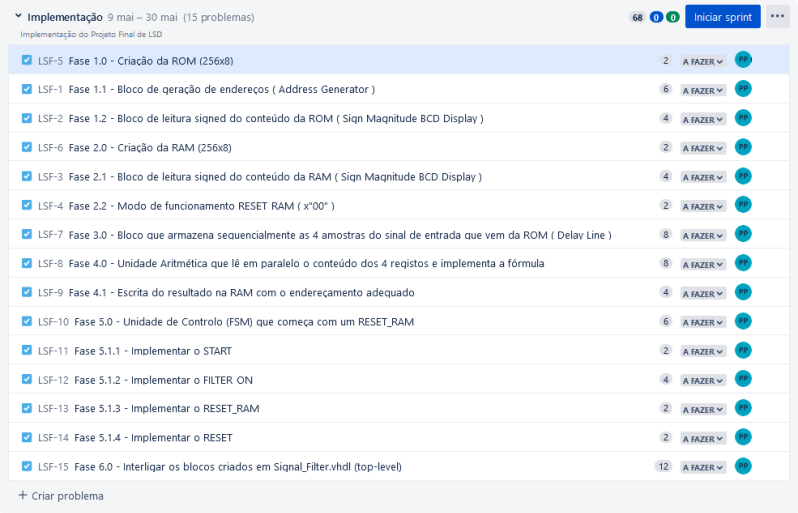


Figura 2. Backlog do projeto [2].

2.1. ROM e RAM

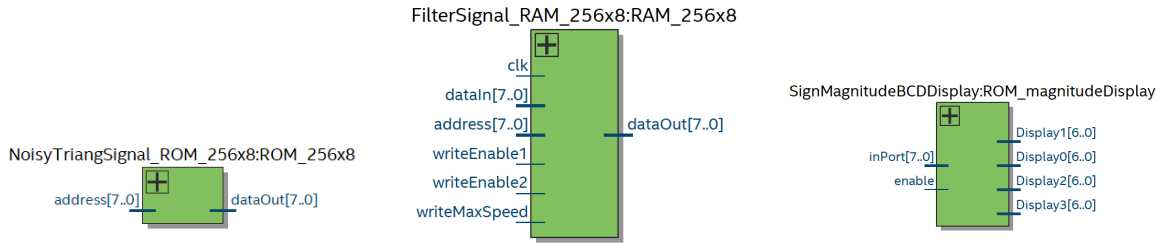
A ROM é utilizada para ler os dados que se pretendem filtrar, enquanto a RAM armazena os dados após o processamento. Implementei a ROM (Read-Only Memory), de dimensão 256x8bits, conforme ilustrado na Figura 3a. A RAM (Random Access Memory), com dimensão idêntica à da ROM, com operações de escrita e leitura, como se pode ver na Figura 3b.

2.2. SignMagnitudeBCD

Este componente é responsável por representar números com sinal em 4 Displays de 7 Segmentos (Sinal, Centenas, Dezenas, Unidades), como ilustrado na Figura 3c. Ao implementar este componente, recorri a uma série de subcomponentes. O Bin2BCDSigned_8Bits recebe uma entrada de 8 bits, analisa a polaridade do valor e converte o valor absoluto em BCD. Depois, invoquei o módulo Bin7SegDecoder três vezes para codificar estes valores BCD para um Display de 7 Segmentos. Por fim, encaminhei os valores codificados para cada Display. Nesta etapa, certifiquei-me de que a polaridade do valor, positiva ou negativa (conforme determinado pelo Bin2BCDSigned_8Bits), estava corretamente representada.

2.3. AddressGenerator

Este componente tem a responsabilidade de gerar o endereço da ROM. Inicialmente concebido como um simples contador de 0 a 255 (8 bits), o AddressGenerator teve de sofrer pequenas alterações para suportar



(a) **(b)** **(c)**

Figura 3. (a) Read-Only Memory (b) Random Access Memory (c) SignMagnitudeBCD

todos os estados do sistema. Estas alterações incluíram a adição de dois sinais de habilitação e uma entrada chamada MaxSpeed. Quando ativada, esta entrada faz com que o contador ignore os sinais de habilitação e conte à velocidade do relógio de entrada. Além disso, o gerador de endereços possui uma função de reset que reconfigura o endereço para 0x"00", conforme ilustrado na Figura 4a.

2.4. DelayLineRegister

Este componente tem o objetivo de criar uma linha de atraso, funcionando em modo serial-in – parallel-out, para palavras de 8 bits, como pode ser visto na Figure 4b. Esta configuração permite que os valores possam ser lidos pela ArithmeticUnit. Se os sinais writeEnable estiverem ativos e ocorrer uma transição ascendente do relógio, o componente realiza um deslocamento lógico para a direita e introduz o novo valor (proveniente da ROM).

2.5. ArithmeticUnit

Este componente foi concebido para implementar a seguinte fórmula que engloba 4 amostras do sinal de entrada x,

$$y_k = \frac{x_{k-2} + x_{k-1} + x_k + x_{k+1}}{4}, \qquad k = 2, ..., 254, \tag{1}$$

$$y_0 = x_0, \qquad y_1 = x_1, \qquad y_{255} = x_{255}. \tag{2}$$

Observando a fórmula (1), percebemos que temos de inserir na posição k da RAM a média de 4 valores da ROM, considerando que precisamos do valor x_{k+1}. Ao observar a fórmula (2), notamos que os primeiros 2 valores da RAM serão iguais aos primeiros 2 valores da ROM, bem como o último valor da RAM será igual ao último valor da ROM. Isso implica que o filtro deve estar ativo apenas para k=2, ..., 254. Portanto, em termos de endereços, concluímos que o endereço da RAM obtem-se subtraindo "1"ao endereço da ROM e que precisamos de duas entradas de controlo, Figura 4c, uma chamada filterEnb, que ativa o cálculo da média, e outra chamada byteSelect, que seleciona qual dos valores de entrada será colocado na saída quando o filtro não estiver ativo. Neste projeto, esse valor está predefinido como "01", selecionando o segundo valor das entradas. Contudo, se outra média fosse implementada, este componente poderia ser reutilizado. Para entender melhor o Sincronismo do Sistema, consulte a Tabela 1.

AddressROM	DelayLineRegister	FilterEnb	byteSelect	Result	AddressRAM
0	x_0, x_{255}, x_{254}, x_{253}	0	01	$y_{255} = x_{255}$	255
1	x_1, x_0, x_{255}, x_{254}	0	01	$y_0 = x_0$	0
2	x_2, x_1, x_0, x_{255}	0	01	$y_1 = x_1$	1
3	x_3, x_2, x_1, x_0	1	xx	y_2 = Med(..)	2
...	...	1	xx	...	...
...	...	1	xx	...	...
254	x_{254}, x_{253}, x_{252}, x_{251}	1	xx	y_{253} = Med(..)	253
255	x_{255}, x_{254}, x_{253}, x_{252}	1	xx	y_{254} = Med(..)	254
0	x_0, x_{255}, x_{254}, x_{253}	0	01	$y_{255} = x_{255}$	255
1	x_1, x_0, x_{255}, x_{254}	0	01	$y_0 = x_0$	0

Tabela 1. Tabela explicativa do Sincronismo do Sistema

Para calcular a expressão (1), levando em conta a precisão do cálculo, implementei três opções de arredondamento: arredondamento para baixo, arredondamento para cima e arredondamento aritmético, controlados pela entrada "round"de 2 bits.

2.5.1. Arredondamento para baixo (round = "0x")

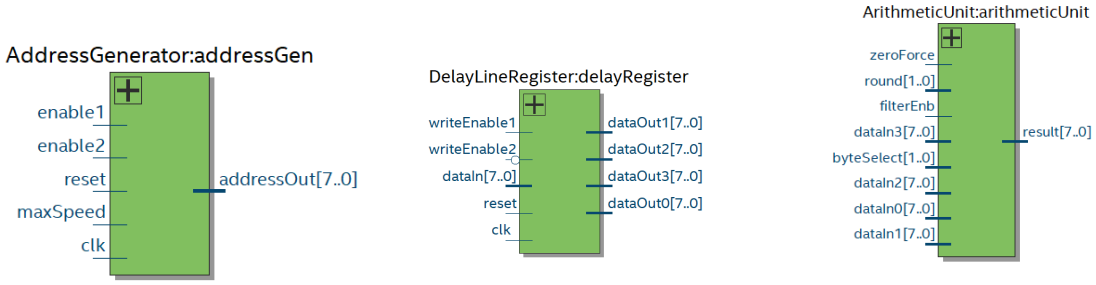
No arredondamento para baixo, a implementação é bastante simples. Basta somar os 4 valores (com sinal), conservando 11 bits (+3 bits - *Full Precision*) para manter a precisão. Depois disso, é necessário executar um "shift aritmético"para a direita de 2 casas. Como descartamos o resto, o arredondamento é sempre realizado para baixo.

2.5.2. Arredondamento para cima (round = "11")

No arredondamento para cima, somamos os 4 valores (com sinal), mantendo 11 bits (+3 bits - *Full Precision*) para manter a precisão. Como queremos arredondar para cima, temos que somar 3 ("11") para que os bits perdidos no deslocamento sejam adicionados. Depois disso, executamos um "shift aritmético"para a direita de 2 casas. Como descartamos o resto mas somámos "11", o arredondamento é sempre realizado para cima.

2.5.3. Arredondamento aritmético (round = "10")

No arredondamento aritmético, somamos os 4 valores (com sinal), mantendo 11 bits (+3 bits - *Full Precision*) para manter a precisão. Como queremos arredondar aritméticamente, temos que somar 2 ("10") para que os bits perdidos no deslocamento sejam compensados. Depois disso, executamos um "shift aritmético"para a direita de 2 casas. Como descartamos o resto mas somámos "10", o arredondamento é sempre realizado para o valor mais próximo (arredondamento aritmético).



(a) **(b)** **(c)**

Figura 4. (a) AddressGenerator (b) DelayLineRegister (c) ArithmeticUnit

2.6. ControlUnitFSM

A Unidade de Controlo tem a responsabilidade de gerir e coordenar a sequência de operações entre os diversos módulos. A Unidade de Controlo foi implementada como uma Máquina de Estados Finitos (FSM - Finite State Machine) de Moore, representada na Figure 5. Esta abordagem permite que a saída seja determinada apenas pelo estado atual, não dependendo das entradas atuais. Isso significa que cada estado tem uma saída fixa e uma vez que o estado é determinado a saída é conhecida.

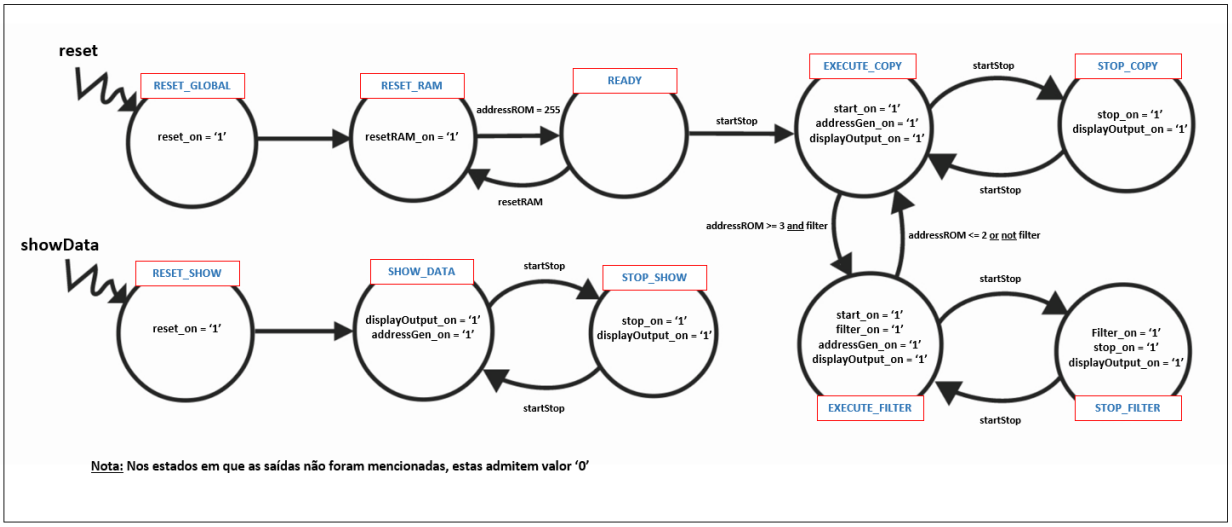


Figura 5. Maquina de Estados

2.7. SignMagnitudeVGA

Este componente é responsável por converter os valores recebidos das memórias para um monitor com interface VGA. Na implementação deste componente, recorri a uma interface feita pelo Professor Tomás Oliveira e Silva [1]. Implementei duas memórias internas (RAM) para armazenar os valores recebidos das memórias do sistema. Esta abordagem permite a construção de um gráfico que representa não apenas os valores actuais mas também os valores anteriormente lidos, conforme se pode ver na Figura 6. Para suportar diferentes monitores, o código foi desenhado para se ajustar com base nas dimensões do monitor em uso, centrando o conteúdo e convertendo os valores em coordenadas (x, y). Para esta conversão, aplicou-se a seguinte fórmula: $y = 256 - (x + 128)$.

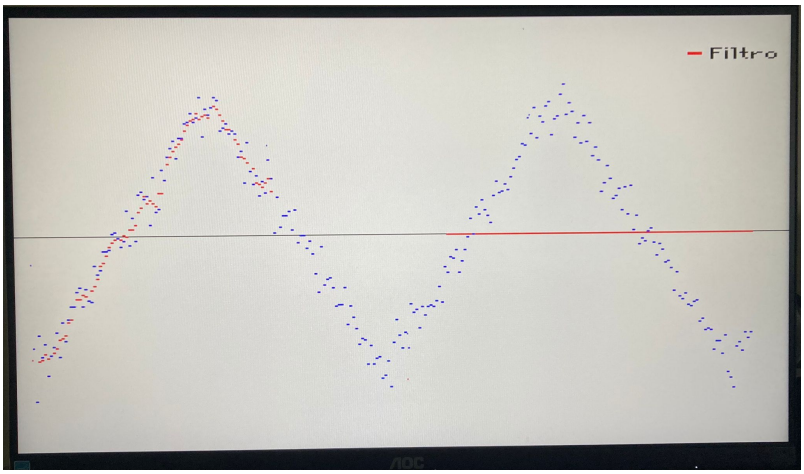
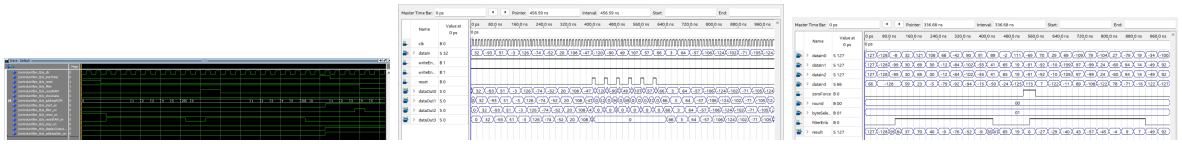


Figura 6. Exemplo monitor VGA.

3. Validação

O processo de validação do projeto foi realizado através da utilização de Test Bench e WaveForms. O Test Bench foi usado especificamente para a máquina de estados, permitindo testar e validar o seu comportamento quando existem variações nas entradas, Figura 7a. As WaveForms foram amplamente usadas para a maioria dos componentes do sistema, Figuras 7b e 7c. Assim, com a aplicação dessas técnicas, foi possível garantir a correta funcionalidade de cada parte do sistema.



(a) **(b)** **(c)**

Figura 7. (a) Test Bench da FSM (b) WaveForm da DelayLineRegister (c) WaveForm da ArithmeticUnit

4. Manual do Utilizador

Compreendendo a complexidade deste projeto, decidi criar um vídeo explicativo para facilitar a compreensão e utilização do sistema. O vídeo apresenta de forma prática e simplificada todos os estados internos e as funcionalidades do sistema.

5. Conclusão

Em conclusão, o trabalho realizado correspondeu às expectativas e aos objetivos traçados no início do projeto. Consegui criar os componentes propostos e conectá-los de forma a criar um sistema funcional. Para além do proposto consegui ainda visualizar os sinais num monitor VGA. Estou bastante satisfeito com o resultado final, a implementação deste projeto foi uma experiência muito enriquecedora, pois permitiu aprofundar os conhecimentos adquiridos ao longo do semestre, assim como ganhar sensibilidade para questões novas, como o detalhe dos arredondamentos e questões específicas do sincronismo.

Referências

[1] Tomás Oliveira e Silva, Laboratórios de Sistemas Digitais - Documentação do material de apoio aos projetos finais, Universidade de Aveiro, 2017.