Introdução

Circuitos eletrônicos complexos são a base de muitas tecnologias atuais, dentre elas todos os eletroeletrônicos e eletrodomésticos. Portanto é essencial que haja uma base de conhecimento prático para os profissionais que atuam nesta área. Desta forma a criação de projetos para treinamento vem a ser indispensável. Mas é necessário conhecer algumas técnicas de desenvolvimento para completar seus objetivos com êxito.

Inicialmente é necessário analisar qual é o problema lógico a ser resolvido, a partir disso pode-se escolher qual é a melhor forma de resolver o problema com mais eficiência e baixo custo. Além disso, a análise prévia possibilita a divisão do projeto em múltiplas partes, reduzindo a complexidade dos circuitos e organizando melhor o processo de desenvolvimento.

A próxima etapa é visualizar quais componentes serão necessários para desenvolver o sistema digital, por exemplo, contadores, registradores, lógicas digitais, MUX ou DEMUX, e o número de entradas e saídas do sistema. Desta forma pode-se montar a tabela verdade para se encontrar o circuito correspondente.

É interessante que a cada etapa do projeto haja simulação do maior número de casos de teste possível, para afirmar que o circuito esteja o mais preciso possível.

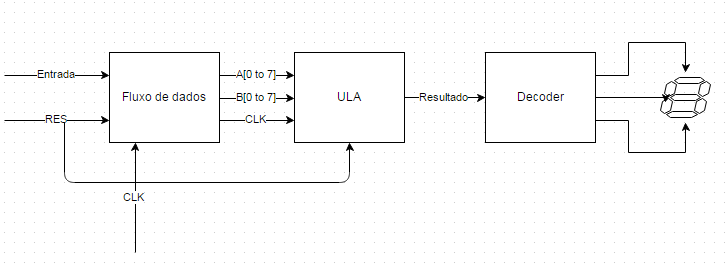
Portanto, o projeto a seguir apresentado seguirá à risca todas as dicas anteriormente passadas para proporcionar um melhor desenvolvimento.

Objetivos

O projeto tem como objetivo criar um tipo de calculadora simplificada com algumas funções simples como, soma, subtração, AND de cada bit e OR de cada bit. A tabela abaixo mostra algumas restrições e outras informações sobre o sistema.

|  |
| --- |
| Operações com valores de até 7 bits de entrada |
| Apenas duas entradas |
| Soma de números inteiros positivos |
| Subtração de números inteiros positivos |

O esquema abaixo descreve o comportamento do sistema.



Como já dito as entradas terão 7 bits portanto o número decimal máximo permitido por tais entradas é de 127. As operações de AND e OR serão baseadas em cima de cada bit de entrada, ou seja, o bit A[0] fará a operação de AND\OR com o bit B[0], assim sucessivamente até o sexto bit.

Montagem do sistema

Para facilitar a construção do sistema ele foi dividido em três partes principais, o fluxo de dados, a unidade de operações lógicas e aritméticas e a decodificação do resultado para quatro displays de sete segmentos.

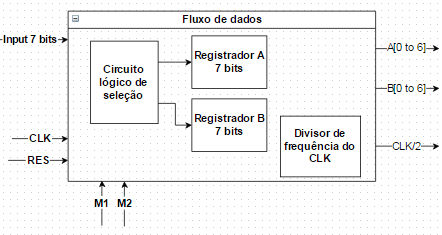
A pesar de as divisões terem sido feitas em apenas 3 partes principais, o projeto ao todo teve diversas divisões. Para cada função diferente foi criado um componente, isso facilita na montagem final. A seguir a tabela mostra todos os subsistemas criados e suas respectivas funções.

|  |  |
| --- | --- |
| Componente | Função |
| Half Adder | Utilizados para criar o circuito somador. |
| Full Adder |
| Half Subtractor | Utilizados para criar função de subtração |
| Full Subtractor |
| AND Function | Utilizados para fazer a função AND e OR, respectivamente, entre os bits das entradas. |
| OR Function |
| Mux4to1 | Multiplexador de 4 entradas de 7 bits |
| Adder7b | Unidade que integra os somados de 1 bit e constrói um somador de 7 bits. |
| Subtractor7b | Unidade que integra os subtratores de 1 bit e constrói um subtrator de 7 bits. |
| Register7b | Registrador de 7 bits, utilizado para armazenar as entradas do sistema. |
| Register8b | Registrador de 8 bits, utilizado para guardar o resultado das operações garantindo que o resultado não será perdido antes de ser decodificado. |
| FFD | Flip-flop tipo D, utilizado para a construção dos registradores. |
| FFT | Flip-flop tipo T, utilizado para fazer a divisão das ondas de clock e tornar o circuito assíncrono. |
| Comp2 | Circuito responsável por fazer o complemento de dois dos resultados negativos, possibilitando que eles sejam decodificados posteriormente. |

Além dos componentes anteriores existem outros, que devem ser melhor explicados para enfatizar suas funções.

* Fluxo de dados (dataFlux)

A unidade de fluxo de dados tem a função de controlar a entrada de dados, selecionar as memórias para efetuar o registro e, no caso deste projeto, divide o clock com um FFT para garantir a chegada dos dados na ULA.



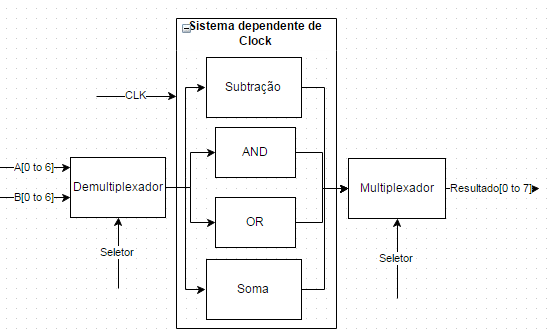
Esquema de conexões do fluxo de dados.

|  |  |
| --- | --- |
| Inputs e Outputs | |
| Input 7 bits | Entrada de dados vinda do usuário. |
| CLK | Clock do sistema. |
| M1 e M2 | Seletores de memória. Ao selecionar-se M1 grava-se no registrador A, e M2 grava-se no registrador B. |
| CLK/2 | Clock de saída. Metade do clock de sistema. |
| A e B | Saída dos dados guardados nos registradores A e B. |
| RES | Reset do sistema. Limpa todas as unidades de memória. |

* UFA

A criação da UFA, unidade de funções aritméticas, tem como objetivos apenas realizar operações lógicas e aritméticas, a exceção é a utilização de multiplexadores e de-multiplexadores para selecionar as operações desejadas. O esquema e a tabela a seguir mostram sua estrutura e pinagem.

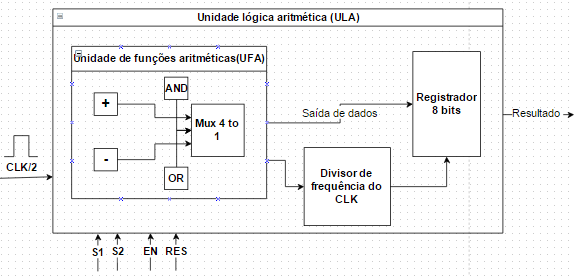
|  |  |
| --- | --- |
| Inputs e Outputs | |
| A e B | Dados de entrada vindos do fluxo de dados. |
| Seletor | Seleciona a função aritmética a ser reproduzida. |
| Resultado | Saída com o valor resultante da função aritmética. |
| CLK | Clock de entrada. Seu ciclo é exatamente a metade do clock da ULA. |



Esquema de conexões da UFA.

* ULA

Unidade que integra dentre outros componentes uma UFA, e aplica outras funções sobre os dados resultantes das operações aritméticas, como, por exemplo, o armazenamento desses dados em um registrador de 8 bits, e a quebra de clock para que não haja problemas sobre as operações lógicas, e salvamento dos dados.



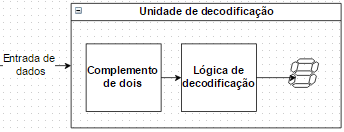
Esquema de conexões da ULA.

|  |  |
| --- | --- |
| Inputs e Outputs | |
| S1 e S2 | Seletor de operações aritméticas |
| EN | Enable. Tem a função de ativar a função aritmética escolhida. (Operador de igual de uma calculadora). |
| RES | Reset. Limpa os dados guardados no registrador de resultados. |
| CLK/2 | Clock de entrada. Tem exatamente a metade do ciclo do clock do fluxo de dados. |

* Decodificação (decoder)

Unidade utilizada apenas para converter os resultados finais das funções aritméticas em dados equivalentes a números decimais em displays de 7 segmentos.

É importante enfatizar que a decodificação não é uma operação tão simples como se parece, pois não existe um padrão de repetição simples, em números binários, que facilite tal operação. Além disso, ao se efetuar a operação de subtração há um grande problema, pois quando seu resultado é negativo é necessário tratar o caso do bit de sinal, e além disso, a conversão do resultado de binário para decimal não proporciona o resultado correto equivalente. Então foi necessário agregar o conceito de complemento de dois, ele proporciona exatamente o valor equivalente de uma subtração negativa em termos de binário decimal. Portanto, para que a decodificação seja correta, foi adicionado um componente que trata esse tipo de problema, utilizando o conceito anteriormente explicado.



(Esquema de conexões do decoder.)

Desenvolvimento

O clock utilizado no sistema foi quebrado 3 vezes, inicialmente ao sair do fluxo de dados, para garantir a chegada dos dados à entrada da ULA, e posteriormente dentro da ULA, na passagem de dados para o registrador dos resultados das funções aritméticas. O trabalho com o clock deve ser minucioso, pois qualquer atraso pode acarretar em erros graves nos resultados.

Ao longo do desenvolvimento do sistema muitas tabelas verdade foram feitas para auxiliar na visualização de padrões e na criação de circuitos menores. Entre elas, a mais importante é de seleção de equivalência binário decimal para os displays de 7 segmentos.

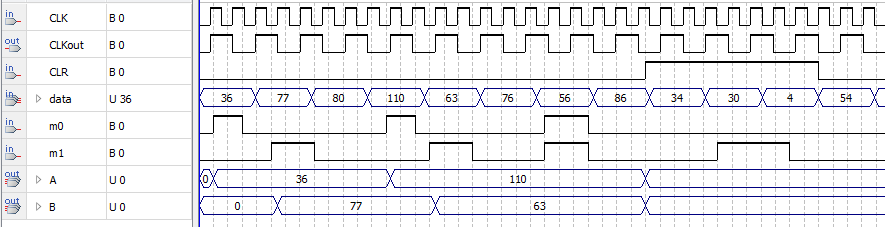
|  |  |  |
| --- | --- | --- |
| Tabela de equivalência binário decimal para a ligação dos displays | | |
| Equivalência decimal | SIGNAL equivalente | Valor binário |
| 0 | N0 | 0000001 |
| 1 | N1 | 1001111 |
| 2 | N2 | 0010011 |
| 3 | N3 | 0000110 |
| 4 | N4 | 1001100 |
| 5 | N5 | 0100100 |
| 6 | N6 | 0100000 |
| 7 | N7 | 0001111 |
| 8 | N8 | 0000000 |
| 9 | N9 | 0000100 |

Para evitar a repetição dos valores binários foram criados SINAIS com os valores equivalentes aos números decimais.

Para a decodificação foi necessário criar condições para cada caso diferente dos 8 bits, ou seja, para cada estado do valor que entra no decodificador há um condicional que atribui um valor equivalente ao decimal, incluindo os valores negativos, como anteriormente mencionado. O fato de existir um condicional para cada estado de entrada proporciona 256 casos diferentes para se avaliar. (A Tabela condicional pode ser vista [aqui](Decoder2.xlsx)).

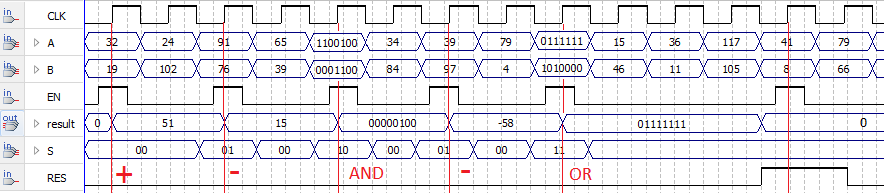
Como em todo projeto, foi necessário efetuar a compilação e simulação de cada componente criado, para garantir que nada de errado fosse ocorrer, e prevenir dificuldades posteriores em encontrar possíveis erros.

* Simulação do fluxo de dados



Como podemos observar, o registro dos dados só ocorre quando o clock está em nível lógico 1 juntamente com os seletores de posição m0 e m1. Também foram testados os casos de teste em que m1 e m2 estejam selecionados ao mesmo tempo, e em que algum seletor esteja ativo ao mesmo tempo que o Clear.

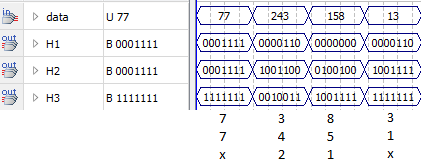
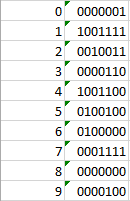
* Simulação da ULA



Outra vez, pode-se notar que as operações ocorrem apenas quando o clock está em nível lógico 1, juntamente com o EN ativado, o que ativa a unidade de funções aritméticas fornecendo o resultado.

É importante notar que os resultados fornecidos em número binário, provenientes das operações AND e OR, possuem um bit a mais do que a entrada. Esse bit é proveniente das funções de soma e subtração. Portanto, a saída foi padronizada com 8 bits para facilitar o envio dos dados para o decoder. Tal bit não influencia nas funções AND e OR pois nestes casos ele é sempre 0.

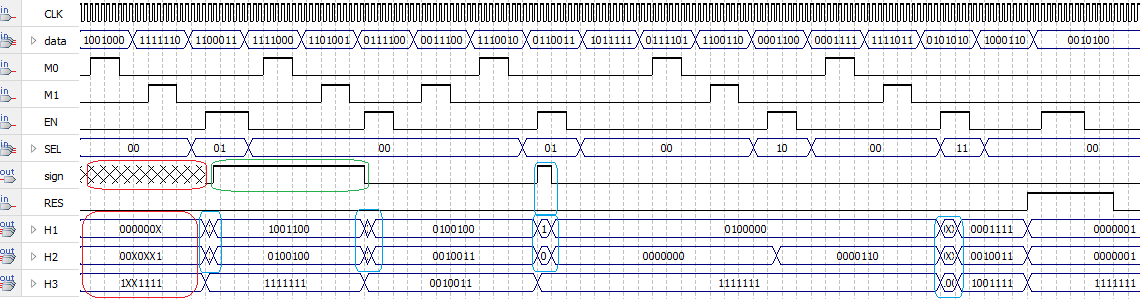
* Simulação do decoder

Considerando que o bit mais significativo para o decoder é o H3 e analisando a tabela de equivalência binário decimal, temos as decodificações corretas para os valores de entrada acima.

O valor x é considerado como display apagado.

* Simulação da unificação de todas as unidades (Calculator)



Existem alguns pontos muito importantes de se enfatizar nesta simulação. Começando pelo clock tão rápido. A escolha de um clock com uma frequência tão alta foi necessária para efetuar a simulação pois, como vimos anteriormente, a sua divisão ocorre 3 vezes, o que aumentaria muito o tempo entre ciclos. Portanto, como o período de simulação tem um tempo máximo de 100us foi necessário utilizar tamanha frequência de clock. Ao transportar o sistema para uma placa FPGA, seu clock de 50Kh será suficiente, pois o tempo de reação dos usuário é muito grande em comparação a tamanha frequência.

Pode-se perceber que existem alguns contornos coloridos na simulação, eles destacam as mudanças em relação às unidades particulares.

O círculo verde destaca o sinal do valor resultante, ou seja, se o número resultante for positivo a linha ‘sign’ vai estar em nível lógico 0, e caso negativo a linha estará em nível 1. Essa valor será utilizado para ligar um sinalizador em outro display.

Os círculos vermelhos e azuis são relativos a ausência de dados concretos. Nos círculos vermelhos temos valores aleatórios na saída dos displays pois ainda não houve nenhuma operação para inicializar os dados a serem impressos. No caso dos azuis temos um pequeno problema, que é o ruído gerado enquanto o processo aritmético está ocorrendo, ou seja, ainda não há um valor concreto na saída de dados.

Conclusão

É importante enfatizar que na criação de um projeto sempre deve-se documentar o que está sendo feito em cada uma de suas partes, principalmente se ele for muito extenso. Outro ponto importante é a divisão do sistema principal em subsistemas, isso permite que o programador\desenvolvedor possa trabalhar com maior organização do projeto, e também facilita muito a unificação do sistema final. Caso o projeto desenvolvido não fosse dividido em 3 partes seu circuito ficaria muito mais confuso de se entender.

Na criação de qualquer projeto deve-se analisar cuidadosamente o tempo de ativação dos sistemas, sempre se baseando em alguma entrada de clock, e considerando o atraso de propagação das portas lógicas.

Para que a lógica funcione corretamente é aconselhado montar as tabelas verdade de todas as entradas e saídas, mesmo que elas não se dependam entre si, isso facilita muito a compreensão da lógica e organiza o projeto. Além disso é muito interessante utilizar as ferramentas de simplificação de circuitos, como por exemplo o mapa de Karnaugh, que muitas vezes pode diminuir consideravelmente o número de portas lógicas utilizadas no projeto, com isso diminuindo-se os custos.