Relatório de Projeto

Título

Sistema de controle supervisório de uma delicatesse

Disciplina

Laboratório de Circuitos Lógicos

Professora

Fernanda Cecília Correia Lima Loureiro

*fernanda@dee.ufcg.edu.br*

Equipe

**Jefferson Emidio Lopes**

**Jefferson Gabriel Dantas de Almeida**

*jefferson.lopes@ee.ufcg.edu.br*

*jefferson.almeida@ee.ufcg.edu.br*

Campina Grande − PB

Dezembro de 2020

Lista de Ilustrações

Figura 01 - Waveform **04**

Figura 02 - Bloco *sistemaDeControle* top level **05**

Figura 03 - Bloco *core* **05**

Figura 04 - Bloco memory **06**

Figura 05 - Top-level em Verilog **07**

Figura 06 - Core em Verilog **08**

Figura 07 - Memory em Verilog **09**

Figura 08 - Pin Planner **10**

Sumário

[Introdução](#_heading=h.murqrky5y32z) **3**

[Objetivos](#_heading=h.eg5v5qmcbh18) **3**

[Expressões Lógicas](#_heading=h.w4d3h6h3uosz) **3**

[Circuito](#_heading=h.l45q0ftv8j62) **4**

[Código](#_heading=h.s2ic1cy1l67s) **6**

[Aplicações Alternativas](#_heading=h.ecvif4l2qq2h) **10**

[Melhorias e Dificuldades](#_heading=h.wq1c7vcz7e9f) **11**

1. **Introdução**

Atualmente, com a crescente emancipação dos sistemas de produção, emerge a necessidade de se sistematizar os métodos de supervisão e controle. Tais métodos são responsáveis por gerenciar a relação entre sistemas de controle e os usuários, bem como por realizar operações, como registro de dados. Diante disso, esses softwares — normalmente associados ao termo "supervisórios" — possuem um papel de fundamental importância para o correto funcionamento dos meios de produção.

1. **Objetivos**

Realizar a implementação de um sistema de controle supervisório de uma Delicatesse, que tem como banco de dados os códigos dos produtos e sua quantidade em estoque. O usuário deve poder adicionar e retirar produtos do estoque, porém, não admitindo valores negativos, nem sequer superiores a 255 (8 bits).

O armazenamento dos dados, que é o ponto mais importante do projeto, é feito em uma memória RAM assíncrona, contendo 256 endereços de memória, na qual cada endereço representa um produto, e tem 8 bits em cada endereço, espaço esse reservado para a quantidade daquele produto em estoque.

Além disso, o projeto pode ser dividido em dois grandes grupos, um responsável por adicionar certa quantidade de um produto em seu estoque, sem ultrapassar o limite de 256 produtos, e o modo que sua função é de remover uma quantidade de algum produto em estoque, sem deixar que esse número fique negativo.

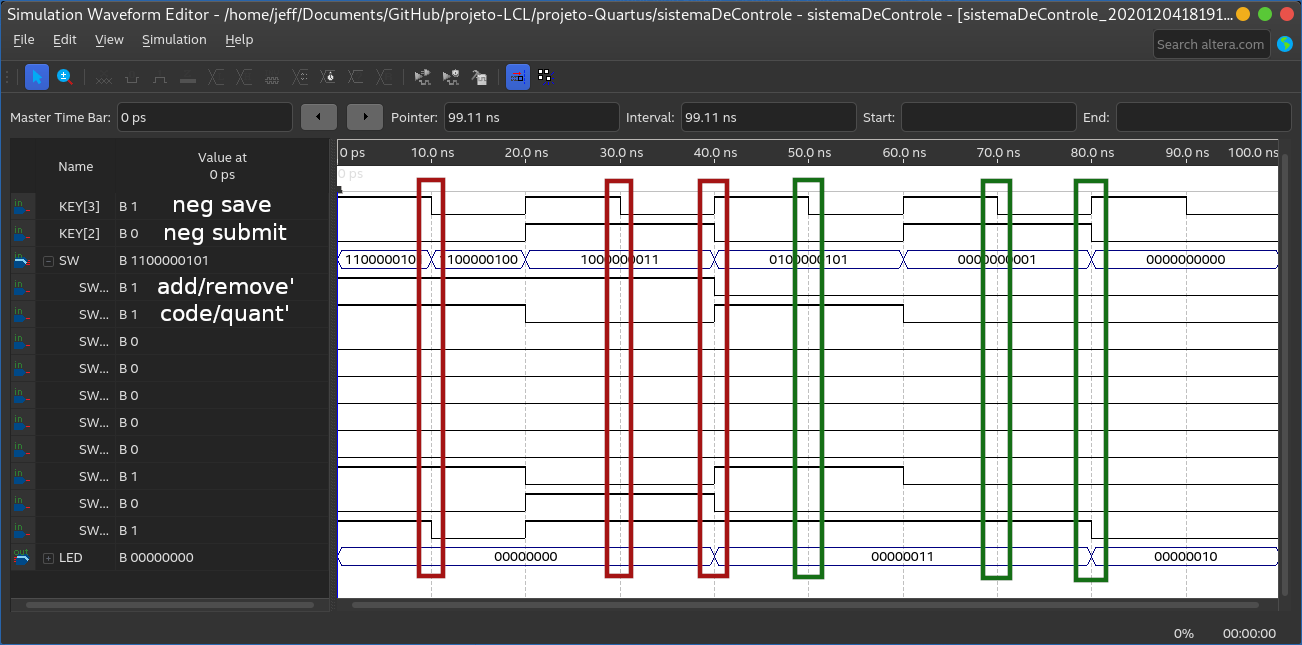
1. **Expressões Lógicas**

Como entradas para o circuito, foram usados 10 switchs e 2 botões da placa DE1-SoC. Para a entrada dos dados principais, como o próprio código do produto e sua quantidade, foi usado 8 switches, sendo eles do *switch[0]* ao *switch[7]*, que representam a entrada *data\_in*. A entrada *Mode*, que é conectada ao *switch[9],* é responsável por alterar entre o modo de adição, *Add*, e o de remoção, *Remove*. Já conectado ao *switch[8]* temos a entrada *c\_q*, que representa a configuração entre o modo *Code*, que é responsável por armazenar a entrada *in* na variável code e o modo *Quant*, que salva a entrada *in* na variável quant. A entrada *save* é usada como um clock manual, que é detectada em sua borda de descida e faz com que o dado na entrada *data\_in* seja salvo em uma variável dependendo do valor de *c\_q*. Já a entrada submit funciona parecido com a anterior, pois também é usada como um clock manual e em sua borda de descida, os dados armazenados em code e quant são enviados para o banco de dados (bloco de memória RAM assíncrona).

Para as saídas, é usado apenas o *data\_out*, que mostra o valor armazenado em um endereço de memória depois de ter sido realizado alguma função como Add ou Remove. A saída é transmitida através de um vetor de LEDs de LED[0] a LED[7] para mostrar o valor do endereço em binário.

A imagem abaixo mostra o funcionamento do circuito, onde as marcações em vermelho chamam a atenção para o uso no modo *Add*, que adiciona uma quantidade a um certo produto, determinado pelo seu código. Por outro lado, nas marcações em verde, chama-se atenção para o uso no modo *Remove*, que retira uma dada quantidade de um produto em estoque.

Figura 01 - Waveform.

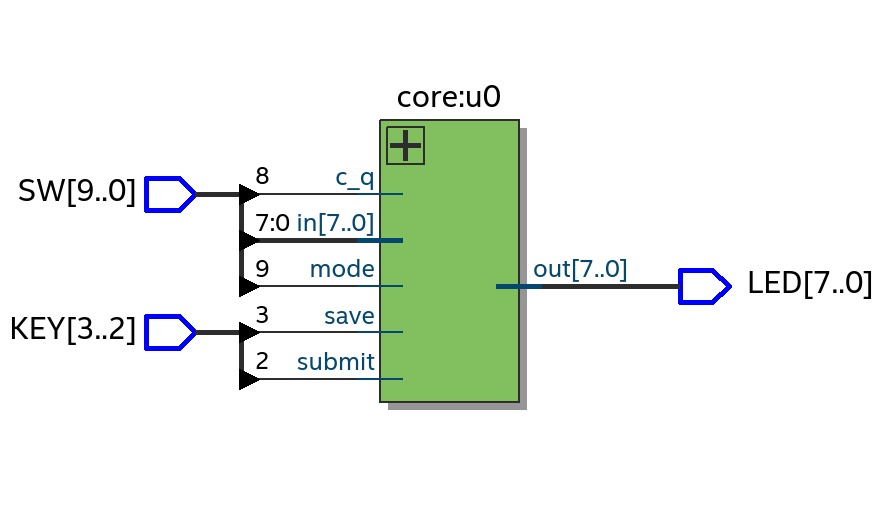


Fonte: Autoria Própria

1. **Circuito**

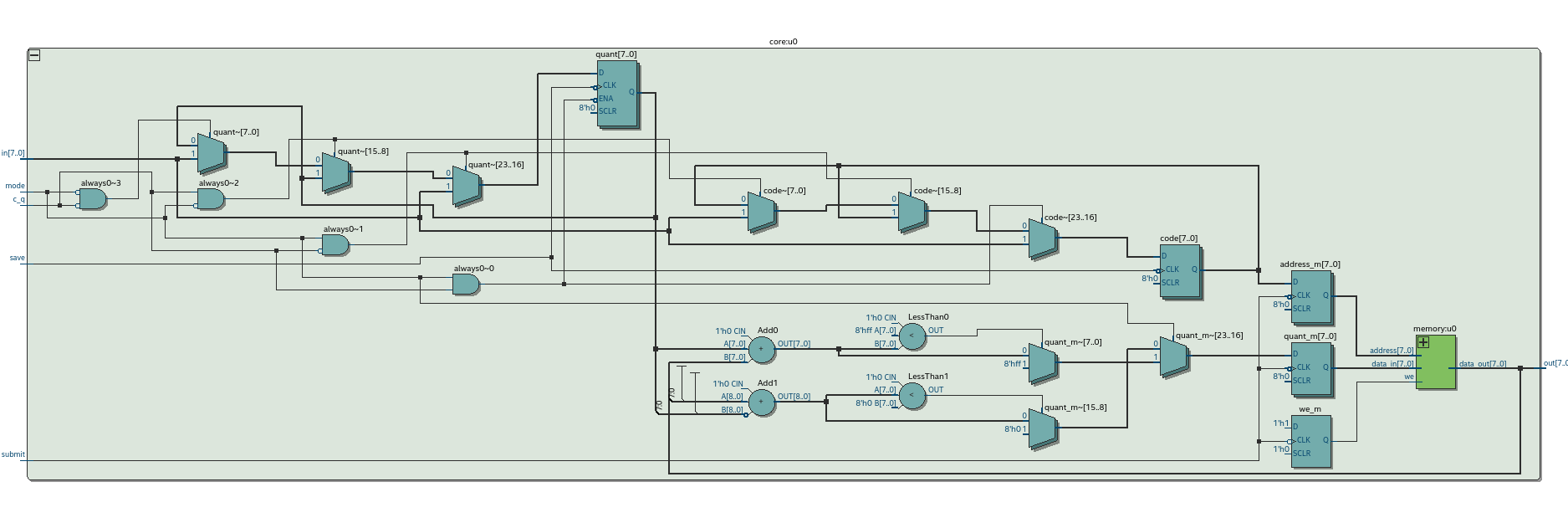
O circuito foi dividido em 3 importantes módulos. O primeiro sendo o próprio top-level intitulado de *sistemaDeControle*, responsável por entrar com todos os vetores de *switches* e botões, assim como o vetor de LEDs na saída. No segundo módulo, chamado de *core*, é onde todo o processamento das entradas é feito, em adição de controlar o bloco de memória RAM assíncrono, chamado de *memory*, que é o terceiro e último bloco, um elemento chave para todo o projeto.

Figura 02 – Bloco *sistemaDeControle* top level.



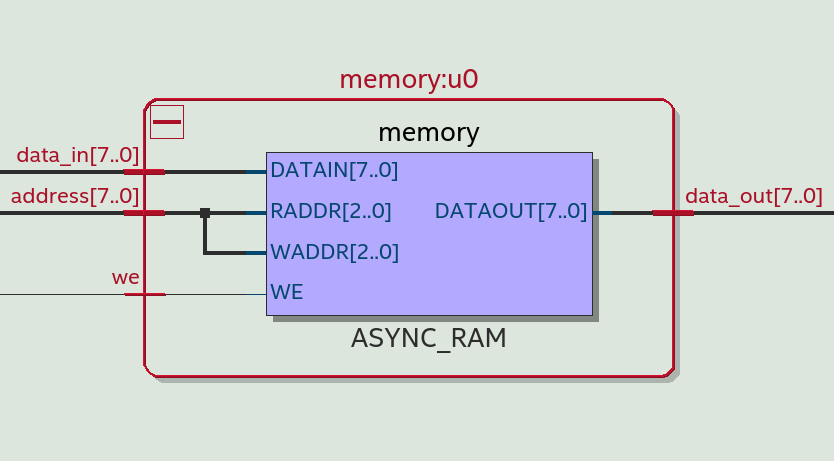
Fonte: Autoria Própria

Figura 03 - Bloco *core*.



Fonte: Autoria Própria

Figura 04 - Bloco *memory*.

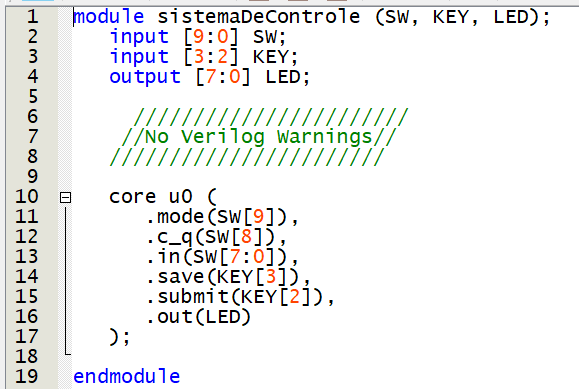


Fonte: Autoria Própria

1. **Código**

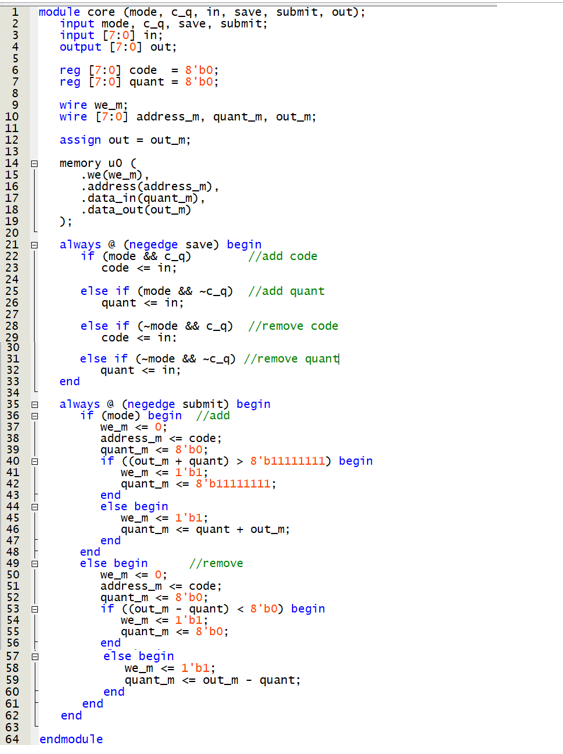
O código do projeto foi feito em *SystemVerilog*, para tirar proveito do uso de vetores dessa linguagem, principalmente no módulo *memory*. Ademais, todo o projeto foi compilado sem receber nenhum tipo de Warnings do verilog, chamando atenção para o cuidado aplicado em toda a programação. Além de conter todo o código em *Verilog*, também foi adicionado a configuração usado no Pin Planner do quartus, visto que o projeto foi realmente implementado em uma placa FPGA, mais especificamente a DE1-SoC.

Figura 05 – Top-level em Verilog.



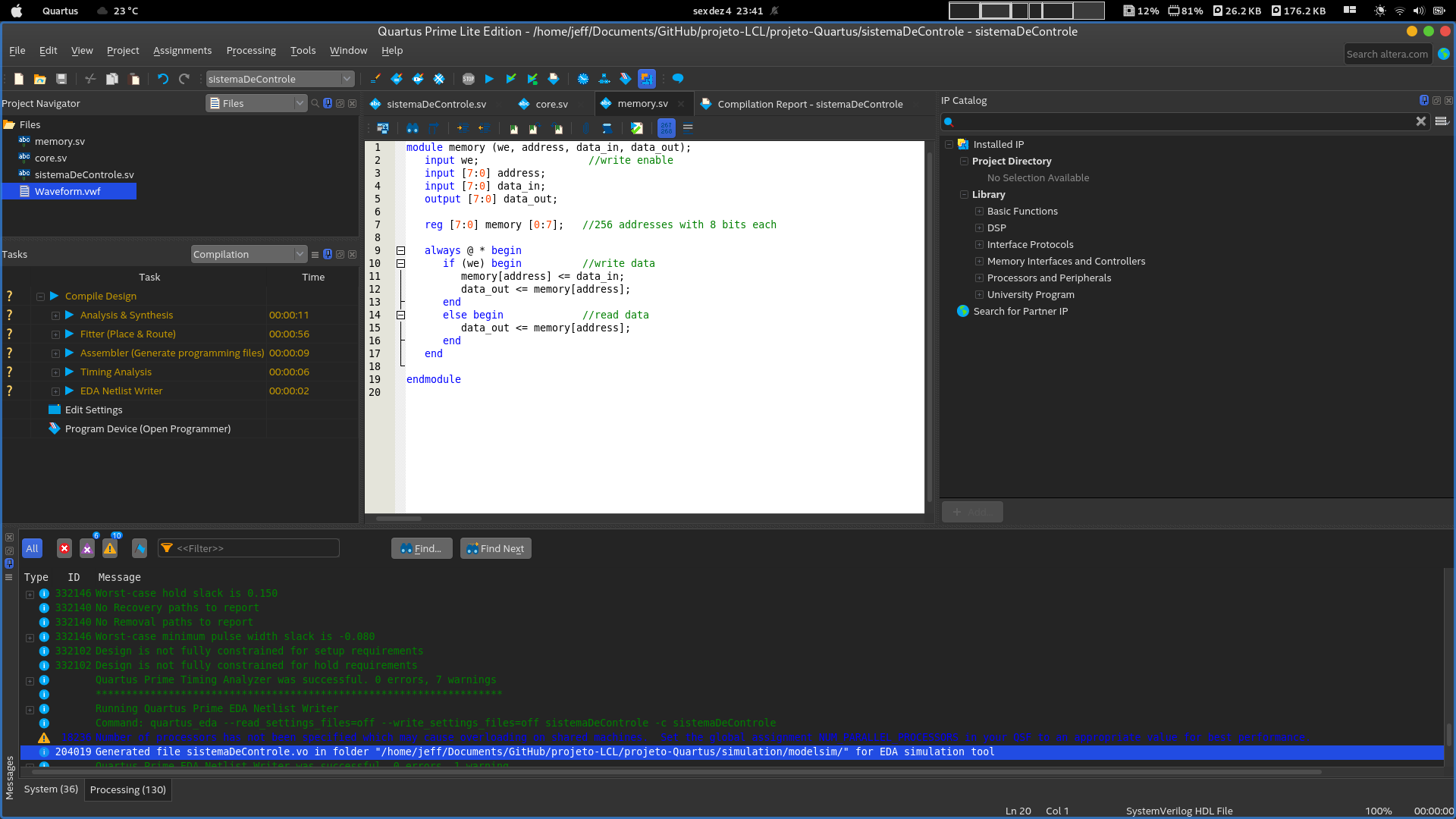
Fonte: Autoria própria.

Figura 06 - Core em Verilog.



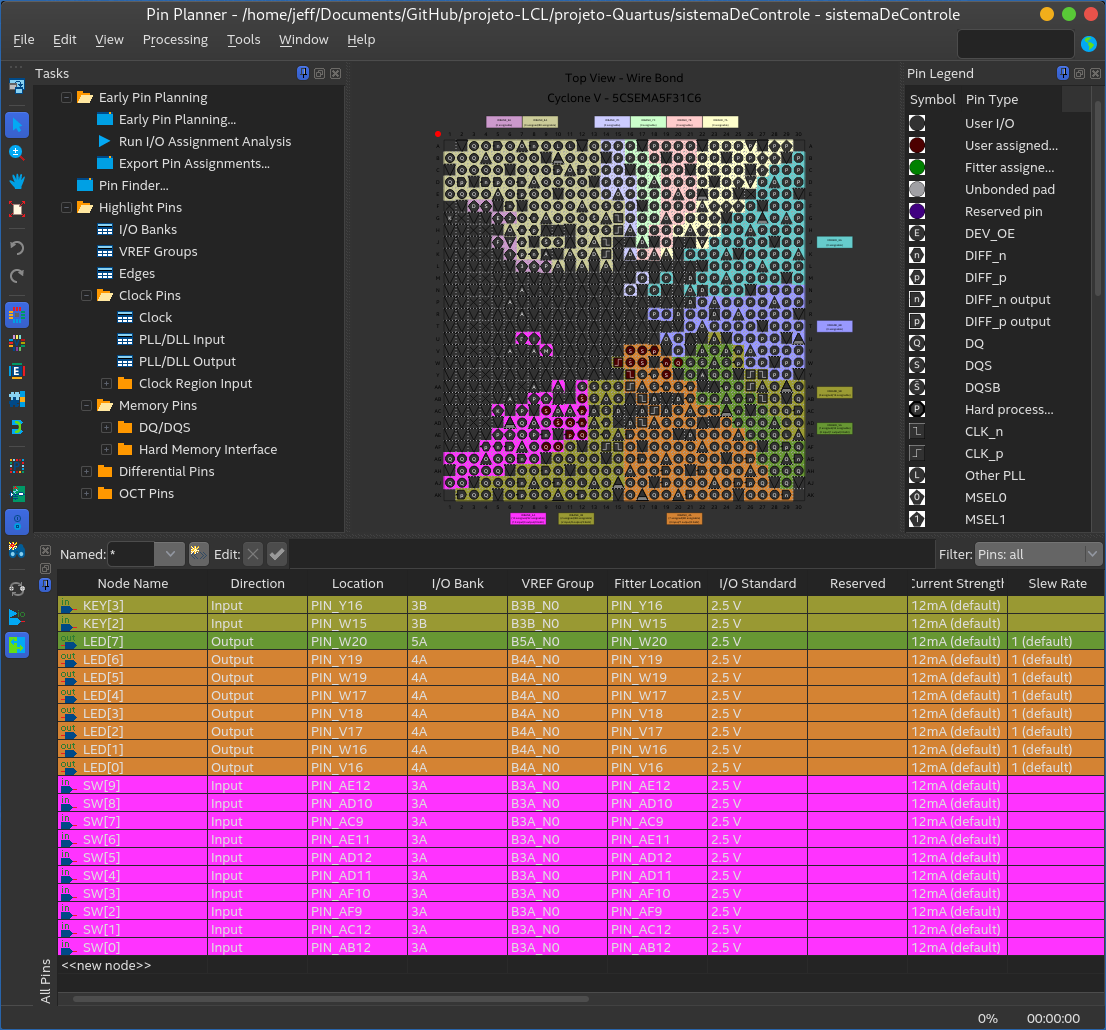
Fonte: Autoria própria.

Figura 07 - Memory em Verilog.



Fonte: Autoria própria.

Figura 08 - Pin planner.



1. **Aplicações Alternativas**

O projeto utiliza-se de conceitos abrangentes para ser aplicado em qualquer estabelecimento comercial, com os devidos ajustes. Para implementação de uma aplicação para sistema supervisório, em primeiro lugar, é necessário um completo conhecimento do processo ou máquina(s) a serem supervisionada(s). Com esta informação é possível, utilizando os recursos de criação de telas, desenvolver aplicações para sistema supervisório com telas sinóticas para cada uma das etapas existentes no processo a ser supervisionado. O desenvolvimento destas aplicações para sistema supervisório envolve custos de engenharia, pois deve ser adaptada para cada tipo de processo a ser controlado. Este trabalho é desenvolvido por especialistas em utilização dos sistemas supervisórios sob orientação de especialistas no processo a ser controlado ou supervisionado. Desta forma, o produto gerado consegue conciliar as facilidades oferecidas pelos ambientes de programação dos softwares supervisórios com o expertise dos engenheiros de processo.

1. **Melhorias e Dificuldades**

Como melhoria, poderia-se implantar um sistema de monitoramento em tempo real da entrada, na qual seria usando um display para expressar o resultado em decimal da entrada, inserido pelo usuário. Com relação ao software, poderia ser implementado um método que fosse usado somente para ler um endereço de memória, ou seja, somente mostrar a quantidade de determinado produto, dado que a entrada seria o seu código.

Quanto às dificuldades, o processo de *debug* implementado ao código no Quartus se mostrou muito trabalhoso, tendo em vista que o conhecimento fornecido acerca do uso do *debugger* foi inexpressivo. Além disso, a baixa usabilidade do próprio programa da Intel, devido a diversos erros, comprometendo o desenvolvimento de projetos maiores como este.