RELATÓRIO PROJETO FINAL CIRCUITOS DIGITAIS

Daniel Augusto Pires de Castro[1]

[1] Universidade Tecnológica Federal do Paraná – Curitiba – PR – Brasil

Resumo. Este artigo tem por objetivo apresentar o Projeto Final da disciplina de Circuitos Digitais, ofertada pelo professor Fabio Kurt Schneider durante o semestre acadêmico de 2023/1. O projeto se trata de um sistema de implementação do algoritmo de Machine Learning "Árvore de Decisão" com o uso de Scikit-Learn, o qual converte seu resultado em um arquivo VHDL utilizando Python, podendo, assim, realizar predições em uma interface da placa DE10-LITE implementada no Quartus.

Palavras-chave: Árvore de Decisão – converte – resultado – predições – interface – placa.

1 Introdução

Avaliando os requisitos do trabalho, evidencia-se que foi proposto um projeto de interesse pessoal com relação à disciplina de Circuitos Digitais e de nível de complexidade equivalente aos sugeridos anteriormente. Com isso, idealizou-se um sistema que envolvesse Machine Learning, área da inteligência artificial que realiza operações por meio de um modelo treinado por dados fornecidos. Para tanto, foi escolhida a "Árvore de Decisão" que possui como resultado uma sequência de operações condicionais a qual pode ser replicada (alterando as sintaxes) em um arquivo VHDL, e transmitida por meio de uma interface limitada aos poucos dígitos da placa DE10-LITE e às poucas ações de entrada dos botões e chaves. Assim, neste presente documento será abordado de maneira sucinta o desenvolvimento do projeto através dos procedimentos utilizados em cada componente.

2 Desenvolvimento

O sistema é composto por duas pastas principais: python e quartus. A primeira pasta compreende a parte em que é treinado o algoritmo e transcrita sua saída em VHDL. Já na segunda é implementada uma interface genérica para a placa DE10-LITE do tipo 10M50DAF484C7G (TERASIC INC, 2016). O desempenho das operações do projeto consiste em executar o código em Python, especificar a localização do arquivo de dados (em

tabela do tipo csv), especificar o alvo das predições, criar um símbolo para o arquivo VHDL atualizado, compilar o projeto do ambiente Quartus, programar na placa e fornecer os valores dos cinco atributos utilizados para prever o valor do alvo resultante. A seguir serão aprofundadas as etapas referentes à produção do projeto.

2.1 Desenvolvimento em Python

No espaço de trabalho para o Python, separou-se o código em três arquivos principais: *_main.py*, *gerenciador_dados.py e conversor_vhdl.py*. No arquivo *main* são feitas as operações de leitura das especificações de arquivo e alvo, além de instanciar variáveis e chamar funções presentes em outros arquivos. No arquivo *gerenciador_dados.py* são realizadas todas as operações referentes à manipulação da tabela fornecida no arquivo, com cuidado especial em colocar os dados em uma matriz, filtrá-los conforme a profundidade (5 por padrão) e consistência dos atributos e cuidado apropriado dos campos categóricos (aqueles de valores discretos, em sua maioria do tipo *strings*). Já em *conversor_vhdl.py* é convertido o resultado do algoritmo (em *string*) para poder transcrever no arquivo *atributos.vhdl* no diretório quartus.

A programação da Árvore de Decisão concebeu-se por meio do Framework Scikit-Learn (SKLEARN, 2007), o qual possui diversas ferramentas para efetuar operações de Machine Learn. Com isso, bastou separar a tabela em uma matriz X de 5 atributos e em um *array* y com o atributo alvo para usar a função *decision_tree.fit(X, y)*, o que gerou como resultado a sequência de estruturas condicionais *if* e *else* para transcrever no VHDL. Além do mais, como a interface utilizada é genérica, é impresso de saída os campos referentes a cada atributo utilizado, bem como o significado numérico de cada atributo categórico (por exemplo, '*true' = 0* e '*false' = 1*). Na Figura 1 a seguir há um exemplo de saída do que é impresso utilizando uma base de dados de diagnóstico de diabetes (BIGML, 2010).

```
Digite o nome do arquivo de dados (incluindo .csv): diabetes.csv
Digite o nome do target para predições: Diabetes

[Atributos em ordem e seus respectivos campos]
FIELd_0: Glucose
FIELd_1: Blood-pressure
FIELd_2: Insulin
FIELd_3: BMI
FIELd_4: Age

[Categóricos em ordem e seus respectivos campos]
PrEdiCao:
>> 0: true
>> 1: false

Arquivo VHDL sobrescrito com sucesso, agora basta criar o símbolo no quartus e compilar o programa.
```

Figura 1. Tela do terminal de execução do código em Python.

2.2 Desenvolvimento em Quartus

Para começar, foi implementada uma máquina de estados gerenciada por um contador hexadecimal crescente de 0 a 11 controlado pelo botão PB1. Esse contador, mostrado na Figura 2, é síncrono e foi aproveitado de laboratórios anteriores ao longo do semestre, sabendo que se utilizou um VHDL auxiliar para as expressões lógicas do mapa de Karnaugh da sequência (arquivo *contador_hexadecimal_crescente_karnaught*.vhdl). Para cada um desses estados existem detectores desenvolvidos em VHDL para determinar o que será mostrado nos *displays* de 7 segmentos (arquivo *estados_display.vhdl)* e qual atributo será atualizado conforme a entrada do botão PB0 (arquivo *estados_pb0*).

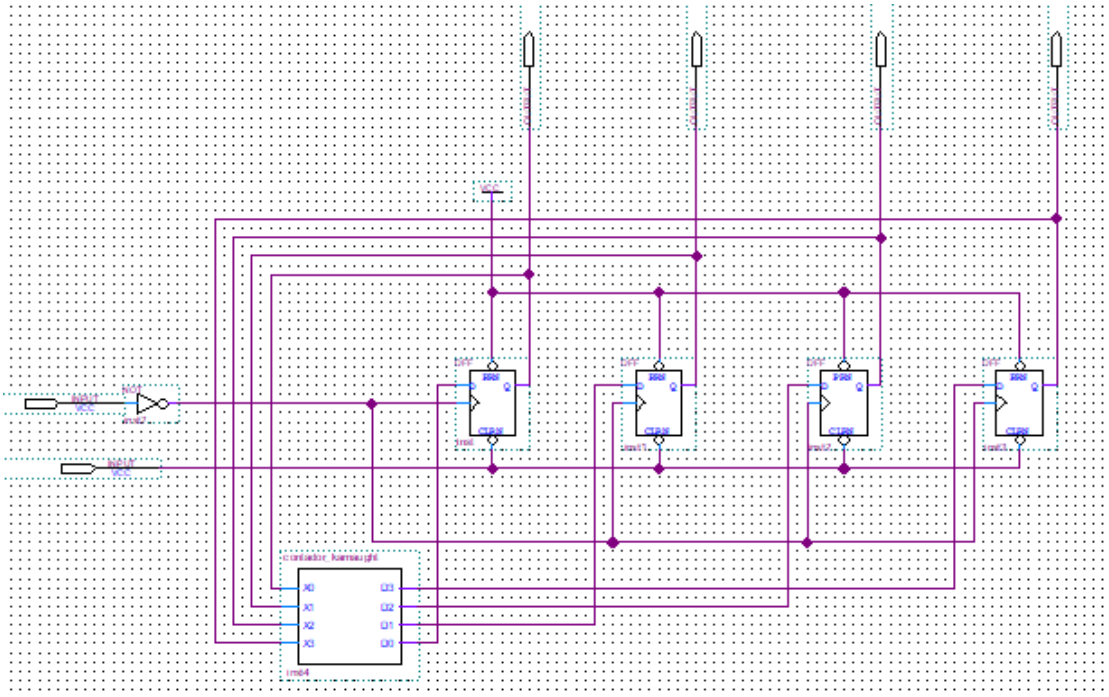

Figura 2. Arquivo *contador_hexadecimal_crescente.bdf*.

Para os estados 0, 2, 4, 6 e 8 são indicados nos *displays* o atributo que deve ser selecionado para os estados 1, 3, 5, 7 e 9, respectivamente. No momento em que deve ser escrito o atributo, utiliza-se o mesmo sistema usado no laboratório 2.2 em que o número binário deveria ser especificado nas chaves de 0 a 3 para registrar e deslocar nos displays conforme os dígitos do atributo. Esse circuito é ilustrado nas Figuras 3a e 3b. Já no estado 10 é escrito “Predic” nos *displays* para seguir ao estado 11 no qual obtêm-se o resultado das decisões da Árvore. A partir disso é resetada a máquina conforme a detecção implementada no arquivo *resetor.vhdl*.

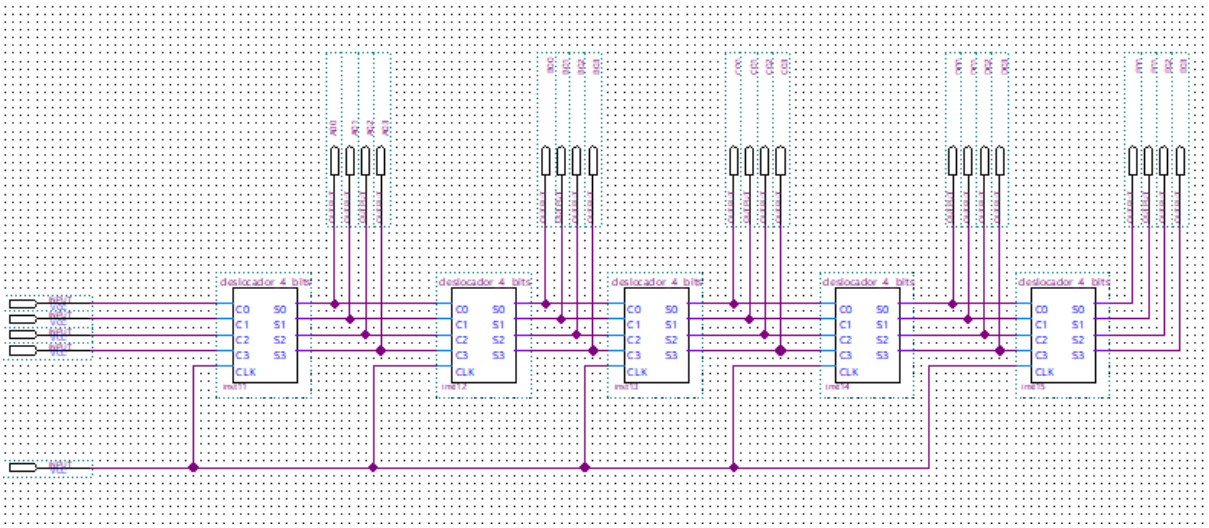

Figura 3a. Arquivo *seletor_atributo.bdf*.

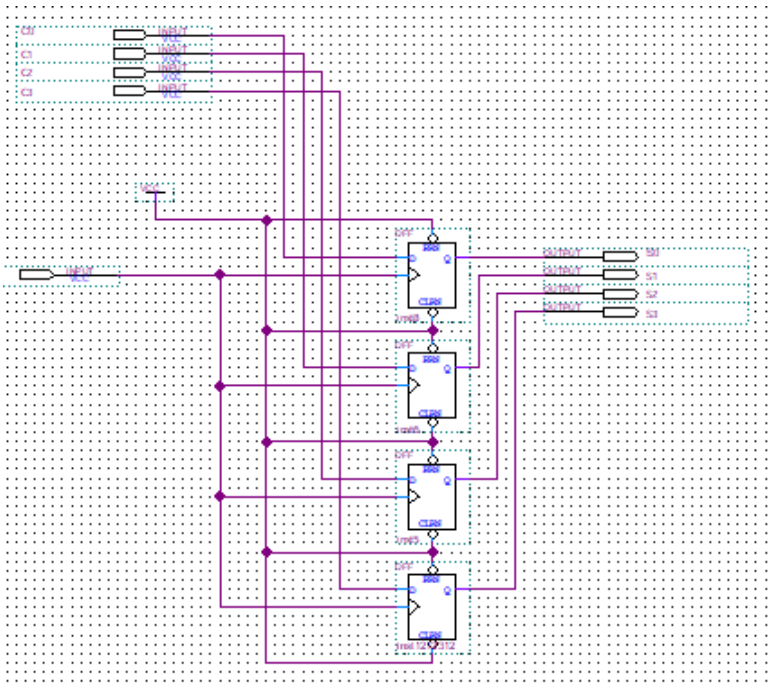

Figura 3b. Arquivo *deslocador_4_bits.bdf*.

Após a seleção do atributo, cada dígito é encaminhado para um conversor de base decimal para base binária (arquivo *conversor.vhdl*) para no fim se tornar uma das entradas de 16 bits para o arquivo *atributos.vhdl*, o mesmo que foi alterado e adaptado com o código em Python conforme a base de dados especificada. Todo esse processo é replicado para cada um dos atributos. Nesse arquivo, por fim, gera-se a predição na base decimal, que é convertida em outro conversor (arquivo *conversor_bin_dec.vhdl*) que o transforma em um número de cinco dígitos decimais para o *display*. Dessa forma, é produzida uma interface genérica o suficiente para dispor de um modelo de 5 atributos devidamente treinado.

3 Resultados e Considerações Finais

Todo o processo de desenvolvimento foi registrado e salvo em um repositório Git com o seguinte endereço remoto: <*https://github.com/daniapc/projeto-final-digitais*>. Avaliando em uma perspectiva funcional, houve êxito na execução do algoritmo em Python, principalmente ao se tratar de qualquer base de dados consistente. O arquivo principal do projeto que mostra o circuito como um todo pode ser visualizado na Figura 4, evidenciando que a implementação em quartus também funcionou conforme o esperado.

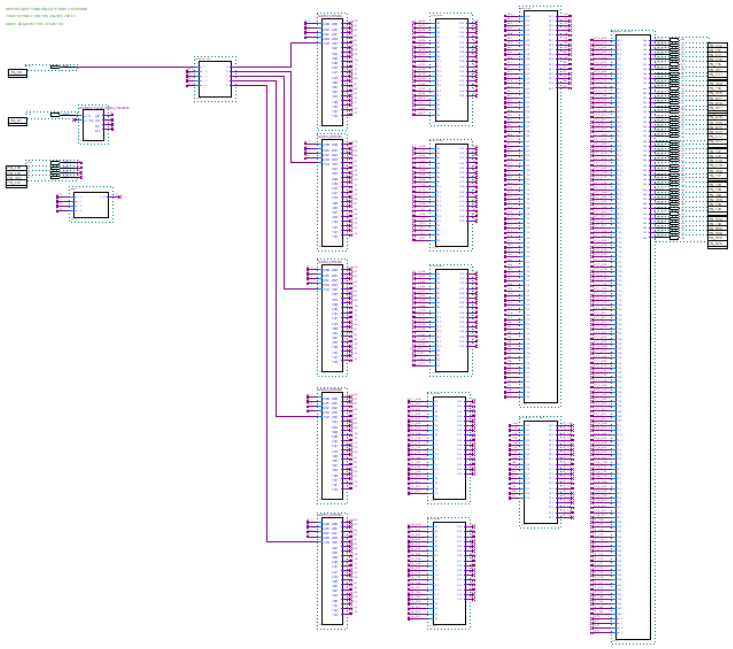

Figura 4. Arquivo *projeto-final.bdf*.

Apesar de tudo, é válido ressaltar que o treinamento do algoritmo não foi efetuado com total precisão, o que indica uma parte a se aprimorar no projeto em algum outro momento. Além do mais, faltou realizar um tratamento com valores decimais para a parte do quartus, outro ponto que poderia ser aperfeiçoado. Contudo, ainda pode-se legitimar o funcionamento da maior parte dos fatores, destacando o empenho e aprendizado.

REFERÊNCIAS

TERASIC INC. DE-10-Lite User Manual. November 21, 2016. Hsinchu, Taiwan.

SKLEARN. Scikit-learn: Machine Learning in Python. 2007. Disponível em: <https://scikit-learn.org/stable/>

BIGML. BigML. 2010. Disponível em: <https://bigml.com/>