Calculo do deslocamento de corpos usando Quatérnions

Allan Ribeiro - 2037807
Lucas P. S. Leyser - 1860275

21 de Dezembro de 2022

1 Introdução

A intenção era fazer o uso de FPGA para cálculo do deslocamento de uma partícula no tempo fazendo uso da abstração de quaternios, que evita problemas algébricos, diferentemente da abordagem dos ângulos de Euler.

Fez se o uso dos dados de giroscópio e acelerômtetros de um smartphone, e suas respectivas leituras descritas em um arquivo csv, fazendo-se a amostragem em tempo da leitura da aceleração das variações, em eixo x, y e z. Os dados são então filtrados de string para ponto flutuante, processados por um socket, escrito em python, que por sua vez realiza a conexão com a placa.

Na placa Cyclone II, foram escritos arquivos em VHDL que fazem a abstração do uso de registradores para guardar um quaternion, sendo descrito como o valor normalizado de dimensão única. Foram usados 3 registradores, um de controle das operações, um como buffer, e outro que acumula o resultado das operações, devolvendo o resultante da multiplicação de todos os quaternios recebidos.

Sem sucesso, o presente relatório dá visão para problemas que futuros desenvolvedores terão, se usada a mesma abordagem. Dentre os inúmeros, lista-se o empecilho principal: uma operação efetiva de multiplicação. Os procedimentos são detalhados abaixo.

Alguns pontos importantes que leitores devem saber é que a estrutura para a multiplicação dos quaternios em sí está pronta, sendo necessária a correção das operações. Um testbench também foi elaborado para uso com os sources disponíveis, e está funcional. O parsing dos dados também foi realizado, e as respectivas funções estão prontas, além de um envio efetivo realizado.

2 Desenvolvimento do projeto

2.1 Aquisição de dados

Para amostrar os dados a serem usados no projeto, foi feito uso do phyphox, e sua funcionalidade de leitura de diversos sensores de um dispositivo que rode android. Este aplicativo permite a visualização em tempo real das métricas de diversos sensores, além da captura e exportação dos dados em CSV, um formato padrão em texto de armazenamento de dados.

Os dados foram exportados e disponibilizados num arquivo, que por sua vez é processado pelo servidor, que mais tarde alimenta a placa com os mesmos. A formatação transforma a 4-tupla de valores de string (tempo, qx, qy e qz) em pontos fixos, com a mesma quantidade de casas decimais. Isso é importante para facilitar a entrada e armazenamento dos dados na placa. Mais tarde, serão codificados novamente em string para envio efetivo pela rede.

2.2 Conexão de rede

A conexão em sí foi construída em 2 partes. Primeiro, um servidor em python espera por uma conexão em socket, que roda em um computador à parte, aqui referido como servidor. O servidor, após filtrar os pontos, os envia em 4-tupla à placa, um após o outro. Uma vez enviados, espera por uma tupla do mesmo tipo vinda da placa.

A conexão TCP/IP com a placa Altera DE2 Cyclone II, cliente, foi projetada com base no projeto "VHDL FPU"realizado pelo aluno Luciano Bonzatto , no segundo semestre de 2022. Estabelecida com sucesso, é necessário a especificação do endereço de servidor a contatar.

Importante notar que a placa recebe uma string, e não uma tupla. Futuros esforços devem ser concentrados na conversão da string recebida, novamente para uma tupla, e em seguida em ponto fixo.

Os dados são, por fim, escritos em 3 registradores internos da placa, um como indicador de operação (load e mult), um como buffer, e um terceiro acumulador de operações, que é lido ao final das operações com todos os quaternios desejados. Esse é o resultado devolvido ao servidor.

2.3 Multiplicação

Para o cálculo da multiplicação dos quaternos, foram utilizados sinais do tipo "sfixed"que utiliza um ponto fixo determinando qual a parte inteira e qual a parte decimal do valor recebido, que é então armazenado em um registrador de 32 bits. Isso resulta em uma precisão de 2^{-15} nas operações.

A operação, infelizmente, apresenta um problema: em teste de mesa, a tentativa de multiplicar 0.5 por 0.5 resulta no deslocamento errôneo dos bits.

```
Registrador de 32 bits

00.00000000000000000000000000000000
|| \ Parte decimal
||
| \ Parte inteira
|
 \ Sinal

0.5 * 0.5 = 0b0010 * 0b0010 = 0b0001 = 0.25 (caso ideal)

0.5 * 0.5 = 0b0010 * 0b0010 = 0b000001 = 0.0625 (caso real)
```

Mas o caso é isolado. Para outros casos, o resultado é caótico à primeira vista. Na tentativa de resolver o problema, foram feitos testes de outras operações, e os resultados também são incorretos, porém consistentes.

Devido à falta de suporte a esse tipo de sinal por parte do Qsys, foi preciso utilizar portas do tipo `std_logic_vector` para o recebimento dos valores e a própria FPGA faz a cópia desse valor para um sinal do tipo `sfixed`. Partimos do pressuposto que, como os valores são sinalizados e entre 0 e 1, sempre teríamos dois bits iniciais, o primeiro para o sinal e o segundo para a parte inteira do valor. O restante é a parte decimal do valor, ou seja, utilizando um sinal com o ponto fixo entre o segundo e terceiro MSB do `std_logic_vector` a cópia pode ser feita diretamente, bit a bit.

Foram feitas outras 2 tentativa de abstrair o cálculo: Receber os dados como ponto flutuante, converter para binário, e operar, ou mesmo receber os dados em binário, e fazer a operação bit a bit. Outro problema apareceu aqui, advindo da abstração usada para fazer a multiplicação entre os pontos. No caso de um dos números ser negativo, a multiplicação em complemento de 2 não foi implementada com sucesso. Resolvido este problema, a multiplicação deve funcionar efetivamente. Além disso, python não suporta a escrita dos tipos em binário puro. Apenas como string, e a representação não é satisfatória.

Outra tentativa consistiu em trabalhar com inteiros, fazendo-se shift dos bits, multiplicando-os, e retornando à posição original. Ocorreram os mesmos problemas da abordagem por complemento de 2.

3 Conclusão e recomendações gerais

O projeto teve avanços, e está perto de estar completo. Novamente, a conexão de rede funciona. A multiplicação também, mas existem erros na operação.

Como recomendações gerais para futuros uso do projeto, tem-se: trocar a versão da placa para evitar problemas de licença, problemas de compilação e geração de componentes/-circuitos (Qsys). Adaptar um ambiente GNU/Linux para uso (o Windows possui problemas de caminhos. Será necessário várias edições de arquivos manualmente, para correções dos respectivos problemas), e uso de rede local (um roteador basta).

A forma como a multiplicação foi feita é dependente da biblioteca ieee_proposed, representada pelos arquivos `fixed_float_types_c.vhdl, float_pkg_c.vhdl` e `fixed_pkg_c.vhdl`. Os três tem problemas com versões de vhdl, cujo dialeto deve ser 2008.

As bibliotecas devem ser inserida como arquivos do projeto no Quartus (caso simulação em RTL), no Qsys (para geração dos componentes), e compiladas internamente do ModelSim (para simulação em RTL).

A multiplicação em binário na FPGA é considerada, por muitas referências, como uma área obscura para se implementar. Sem sucesso, algumas recomendações são: implementar uma lógica de complemento de 2 funcional, uso de outro tipo preferível (real não é sintetizável), ou mesmo uso de outros formátos numéricos mais simples de representar. Por fim, se o usuário achar quaiser referências a respeito, divulgue-as.