Sistemas Digitais II

Caio Vinícius Jun Pereira - 13731921
Caio Hokama Freitas - 11812883
João Pedro Lopes de Sousa Gomes - 13680810
Erick Diogo de Almeida Sousa - 13680960
Pedro Guilherme Croitor da Cruz - 13685002

Atividade Formativa 2

Análise dos casos de teste

A *testbench* fornecida tem 4 casos de teste:

1) 3 multiplicado por 6
2) 15 multiplicado por 1
3) Multiplicação por 0
4) Multiplicação por 1

Os testes 3 e 4 são os mais primordiais, já que, se um algoritmo de multiplicação não funciona para nenhum desses casos, não funcionaria para nenhum outro, por serem os mais básicos possíveis. No caso 3, o resultado deverá ser 0, independentemente do multiplicando. No caso 4, o resultado deve ser o próprio multiplicando.

Codificação da *testbench* fornecida

O código da *testbench* fornecida funciona de maneira extremamente simples.

Primeiramente, é "instanciado" o DUT (Device Under Test), que é um *component* do multiplicador com as portas mapeadas de acordo com as entradas a serem simuladas e as saídas a serem observadas. Assim que a simulação começa, os sinais de entrada que representam os fatores da multiplicação (*Va_in* e *Vb_in*) são atribuídos dos valores 3 e 6, respectivamente. Assim que a saída ready estiver em alto, isso significa que a multiplicação terminou. Nesse momento, a saída *result_out é* verificada. Caso o resultado seja o esperado (18), um print aparece no terminal confirmando o sucesso.

Após um período de clock, o processo é repetido para outros pares de entrada. Ao final dos testes, outro print indica o fim da simulação e o clock para de ser atualizado, evitando que o resto dos sinais mude.

Comparação dos tipos de *testbench*

Como mostrado anteriormente, a *testbench* fornecida é codificada de maneira completamente manual. Isto é, para adicionar um novo caso de teste, seria necessário simular novamente as outras entradas e saídas, como os sinais start e ready, além de verificar para cada um desses casos se o

resultado foi o esperado. Com isso, 15 casos de teste a mais resultariam numa quantidade enorme de linhas adicionadas ao arquivo, de forma bastante repetitiva

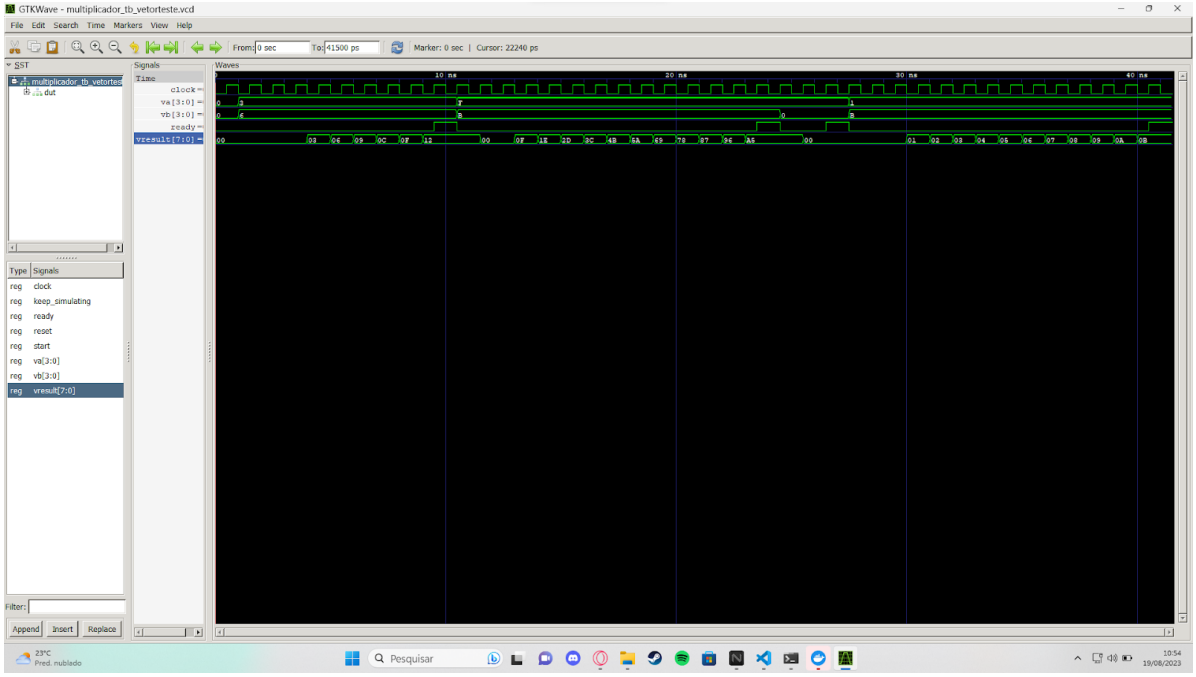
Quanto aos dois outros tipos de *testbench* (vetor de testes e arquivo de testes), adicionar mais casos de teste é um procedimento razoavelmente mais prático, já que é apenas necessário adicionar no arquivo ou no vetor as entradas e saídas esperadas para aquele caso, sem se preocupar com *clock* e sinais auxiliares como o *reset* ou *ready*. Dessa forma, seria necessário apenas uma linha escrita a mais para cada caso de teste.

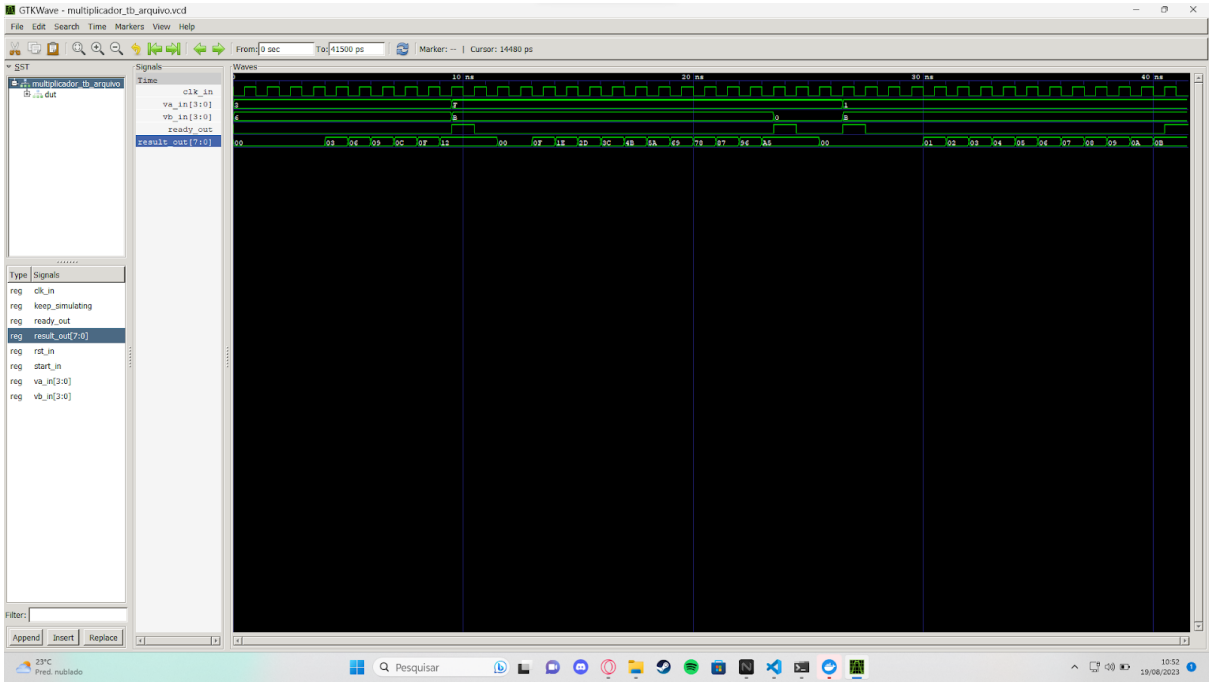
Workflow e resultados

Começamos a fazer a atividade durante a aula, assim que o enunciado foi liberado. Dois membros do grupo ficaram responsáveis por codificar, cada um, um tipo de *testbench* (arquivo ou vetor de testes). Enquanto isso, um deu auxílio e os dois restantes ficaram responsáveis pela documentação do trabalho e pelas respostas às perguntas teóricas do enunciado, que foram aqui resumidas.

Todos nós havíamos estudado no dia anterior os diferentes tipos de *testbench*, então só precisávamos de tempo para escrever o código e corrigir bugs. Nos baseamos no material fornecido pelo professor Midorikawa.

Ainda em sala de aula, a *testbench* de arquivo foi finalizada e testada, mas a *testbench* de vetor de testes estava dando alguns problemas quanto ao clock. Esse problema foi resolvido ao longo da semana, no fim das contas o problema era apenas que estávamos esquecendo de esperar por um ciclo de clock dentro do loop *for* do código. Aqui estão os resultados da *testbench* de vetor e de arquivo, respectivamente:


GTKWave - multiplicador_tb_vetorteste.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 41500 ps
Marker: 0 sec | Cursor: 22240 ps
SST
multiplicador_tb_vetortes
dut
Signals
Time
clock
va[3:0]
vb[3:0]
ready
vresult[7:0]
Waves
10 ns
20 ns
30 ns
40 ns
Type Signals
reg clock
reg keep_simulating
reg ready
reg reset
reg start
reg va[3:0]
reg vb[3:0]
reg vresult[7:0]
Filter:
Append
Insert
Replace
Pesquisar
23°C
Pred. nublado
10:54
19/08/2023


GTKWave - multiplicador_tb_arquivo.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 41500 ps
Marker: -- | Cursor: 14480 ps
SST
multiplicador_tb_arquivo
dut
Signals
Time
clk_in
va_in[3:0]
vb_in[3:0]
ready_out
result_out[7:0]
Waves
10 ns
20 ns
30 ns
40 ns
Type Signals
reg clk_in
reg keep_simulating
reg ready_out
reg result_out[7:0]
reg rst_in
reg start_in
reg va_in[3:0]
reg vb_in[3:0]
Filter:
Append
Insert
Replace
Pesquisar
23°C
Pred. nublado
10:52
19/08/2023