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 5

1. Projeto lógico

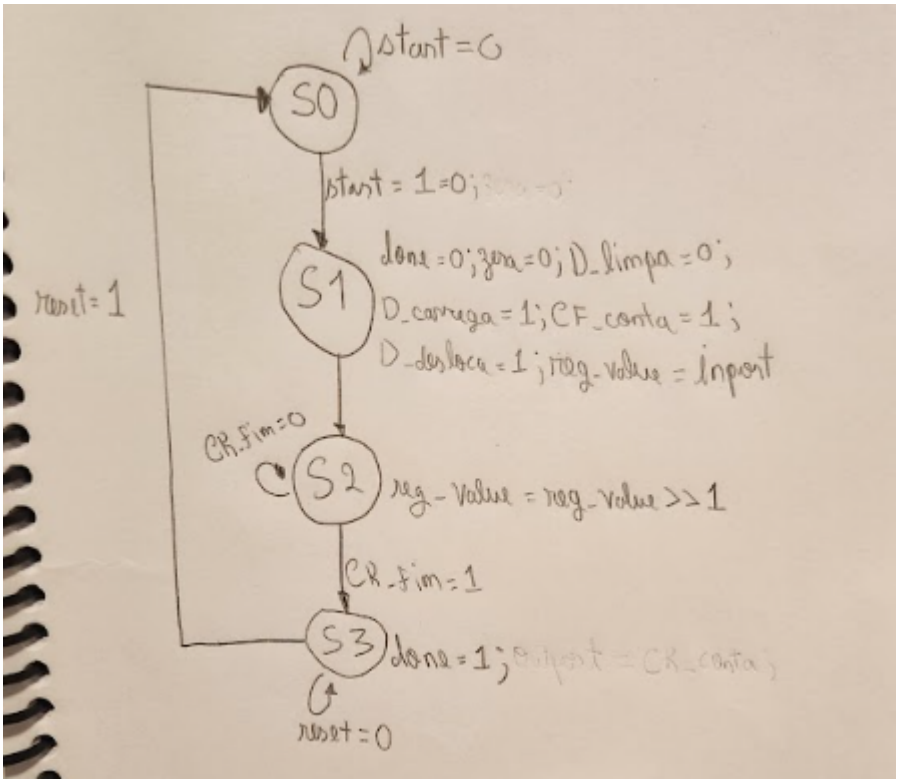
1.1. Planejamento inicial do algoritmo

A primeira ideia do grupo para fazer a contagem de “uns” foi somar todos os elementos da entrada, bit a bit, utilizando diversos somadores. Porém, seriam necessários diversos somadores, e o resultado seria um circuito puramente combinatório. Além disso, na atividade da aula 9, foram criados um deslocador e um contador. Então, o grupo resolveu desenvolver algo em torno desses dois componentes.

Tendo isso em vista, a ideia foi utilizar um contador que só atualiza sua contagem cada vez que o dígito analisado for um bit “1”, o que pode ser feito ligando esse dígito à entrada “conta” do contador, que seria o equivalente a um “enable”. No caso, o dígito analisado sempre seria o dígito menos significativo do sinal “reg_value” armazenado no registrador. Para analisar todos os dígitos um a um, até o mais significativo, seria usado o deslocamento para a direita. Um segundo contador, com saída “Q”, seria o responsável por contar até o valor 15 e assim determinar o fim da contagem.

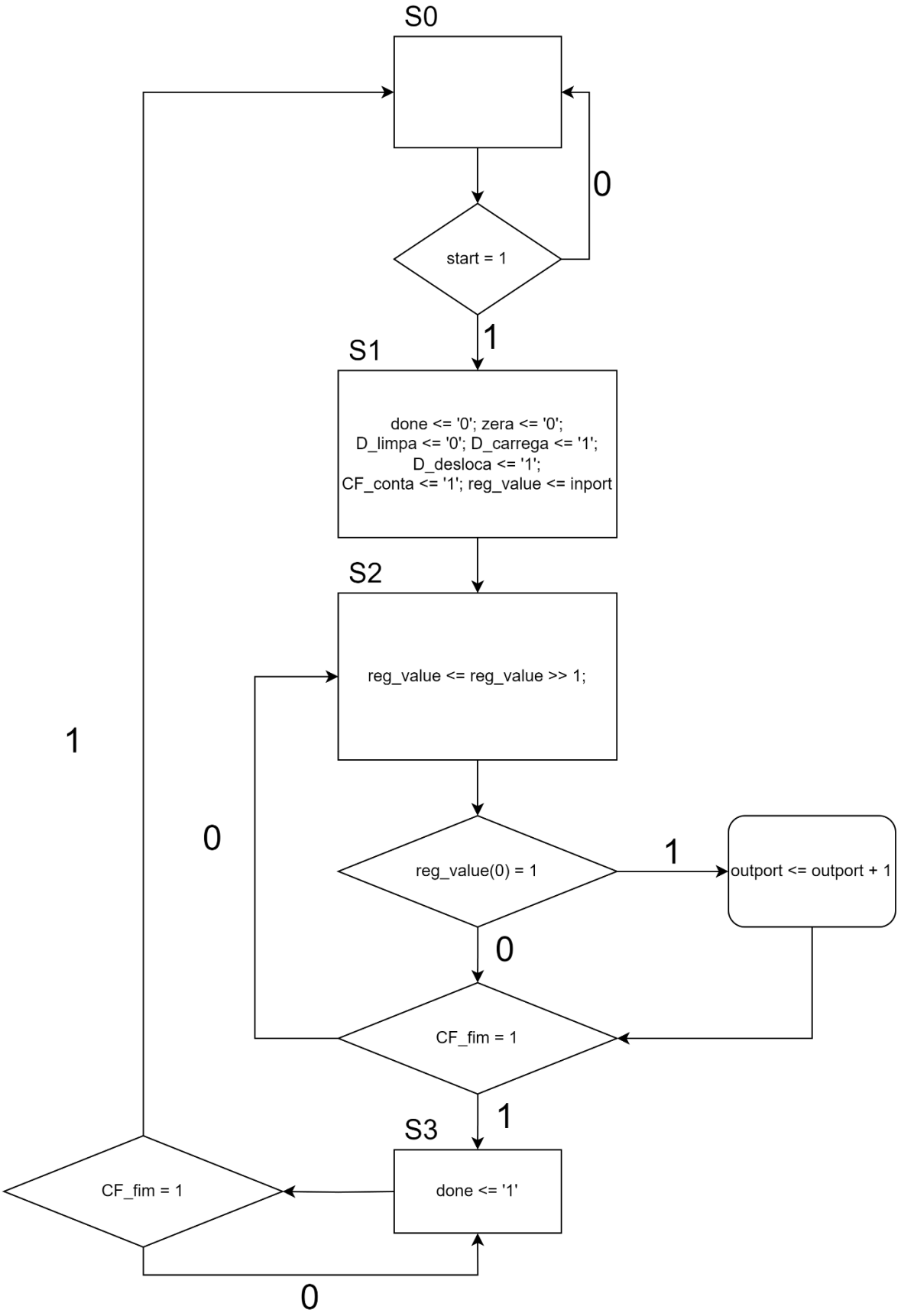
1.2. Finite State Machine with Data (FSMD)

Foram utilizados 4 estados. O S0 é um estado *idle*, em que só se aguarda o sinal *start* virar “1” para iniciar o fluxo. O estado S1 é responsável por zerar todos os sinais. Já o estado S2 é responsável pela contagem em sim, enquanto no último estado, S3, a contagem já chegou ao fim, então somente se espera a entrada *reset* para volar ao estado S0.



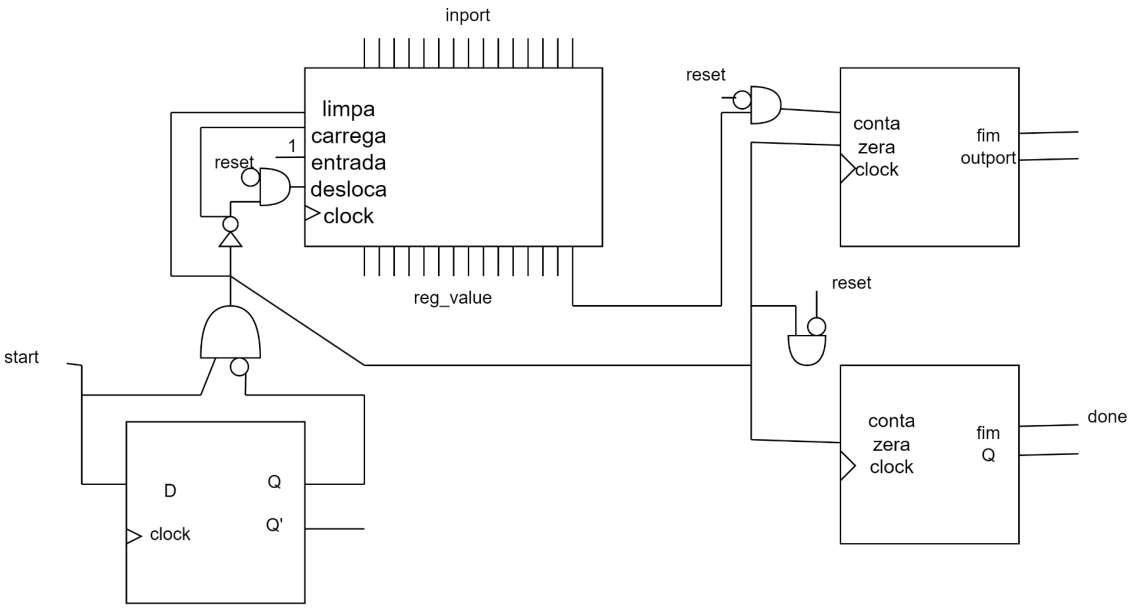
1.3. Diagrama ASM

Seguindo o proposto nas seções anteriores, segue o diagrama ASM desenvolvido:


S0
0
start = 1
1
S1
done <= '0'; zera <= '0';
D_limpa <= '0'; D_carrega <= '1';
D_desloca <= '1';
CF_conta <= '1'; reg_value <= inport
S2
reg_value <= reg_value >> 1;
1
0
reg_value(0) = 1
1
outport <= outport + 1
0
CF_fim = 1
1
S3
CF_fim = 1
done <= '1'
0

1.4. Circuito lógico

O circuito foi construído exatamente da maneira proposta na seção 1.1. A única diferença é que aqui, está presente um Flip-Flop tipo D que armazena o valor da variável *start*, juntamente com uma porta “and” que detecta a borda de subida desse sinal.



2. Definição dos casos de teste

Para os casos de teste, decidiu-se testar os casos extremos, quando todos os bits da entrada são “1” ou todos são “0”. Além disso, foram adicionados casos para testar erros do tipo *off-by-one,* isto é, se o circuito poderia não funcionar por conta do primeiro ou do último bits, especificamente.

Seguem os casos de teste na ordem entrada-saída:

```
("000000000000000", "0000"),
("111111111111111", "1111"),
("101010101010101", "0100"),
("010101010101010", "0011"),
("100000000000000", "0001"),
("011111111111111", "1110"),
("000000000000001", "0001"),
("111111111111110", "1110")
```

3. Codificação do circuito em VHDL

A codificação está presente em anexo e foi feita de acordo com o proposto no circuito da seção 1.4. A única diferença é que não foi necessário implementar o Flip-Flop, pois o VHDL disponibiliza as funções de *rising_edge* e

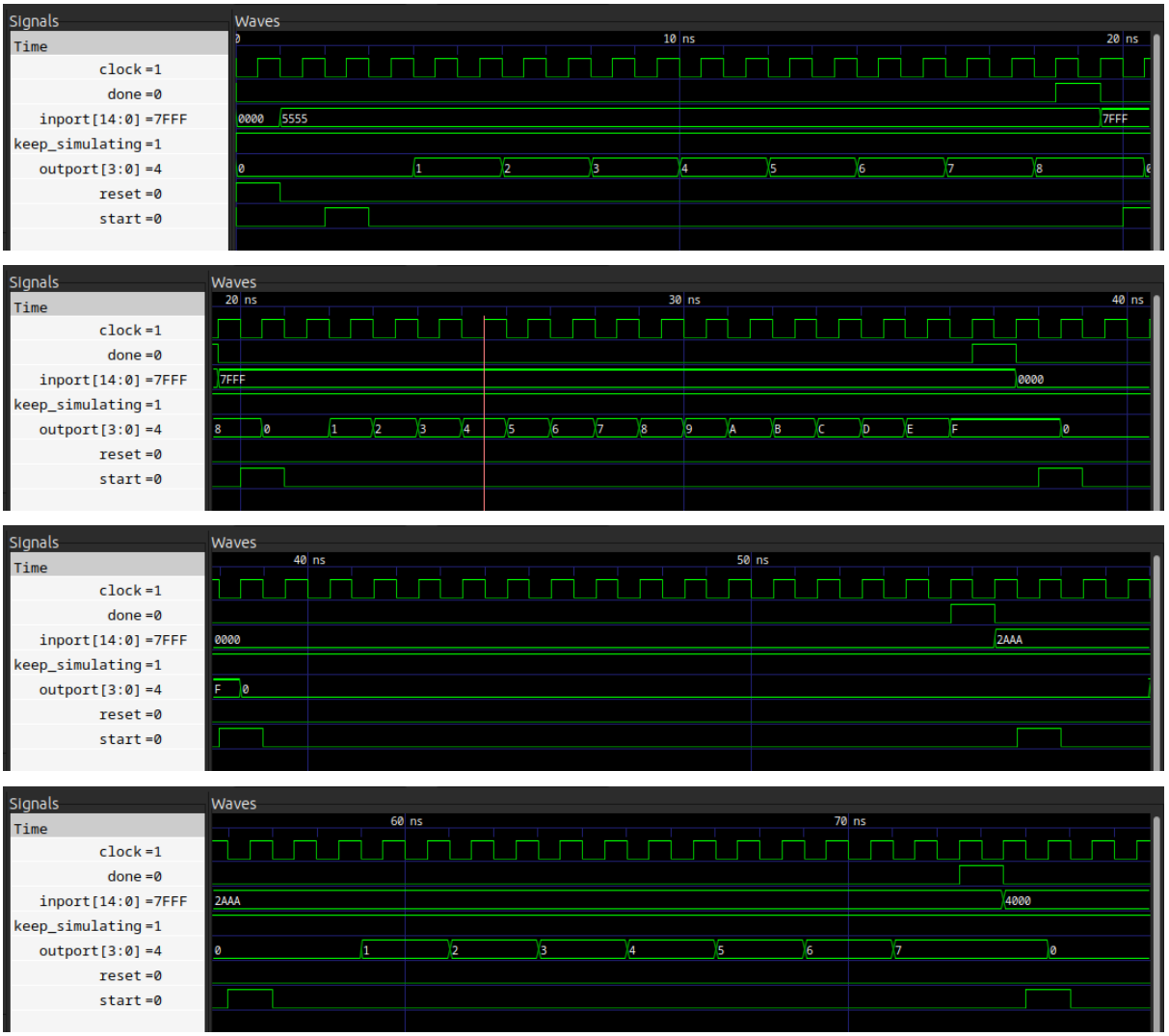
process, que servem justamente para detectar a borda de subida de um sinal. Além disso, o circuito foi separado em duas partes, o *fluxo de dados* e a *unidade de controle.*

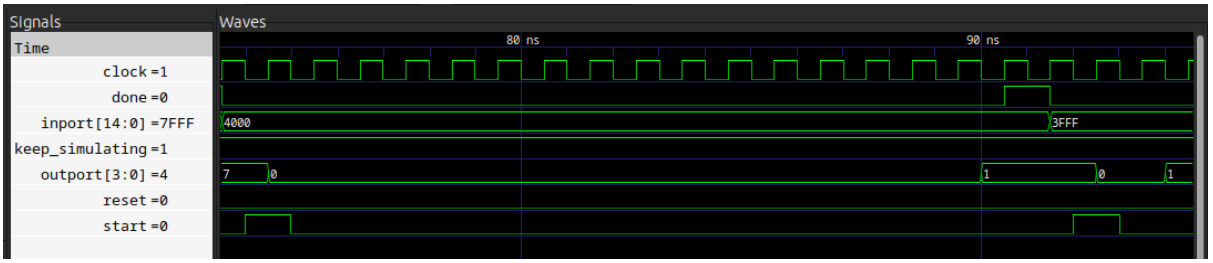
4. Codificação do testbench em VHDL

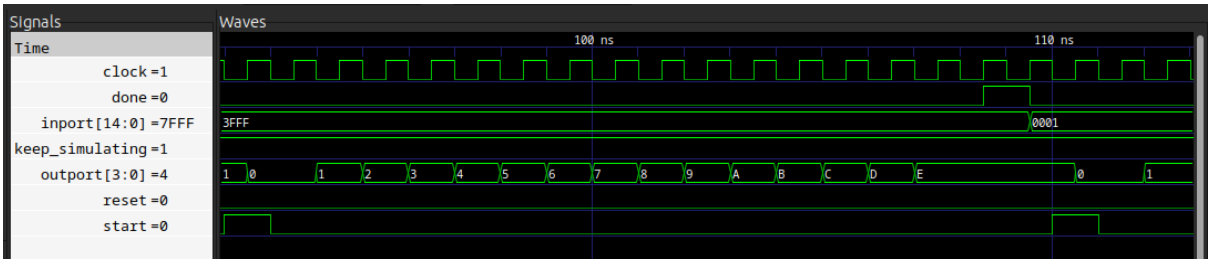
A codificação da testbench também está presente em anexo. O grupo reutilizou a testbench feita na atividade formativa 2 também com pequenas alterações para se adaptar às novas entradas e saídas esperadas. Resolveu-se usar uma testbench do tipo “vetor de testes”, devido à praticidade com que se pode trocar os valores de entrada e de saída.

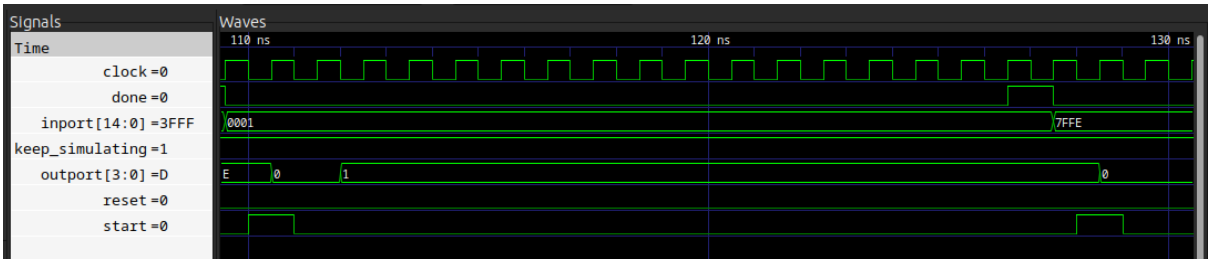
5. Simulação

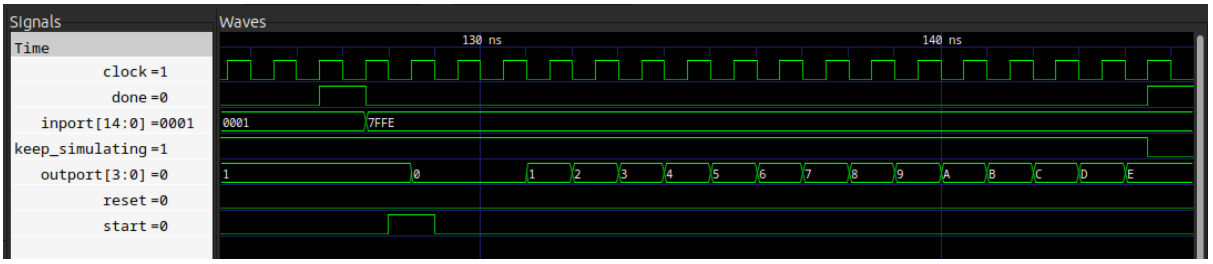
Utilizando os testes definidos anteriormente, o grupo projetou uma testbench e obteve os seguintes resultados.

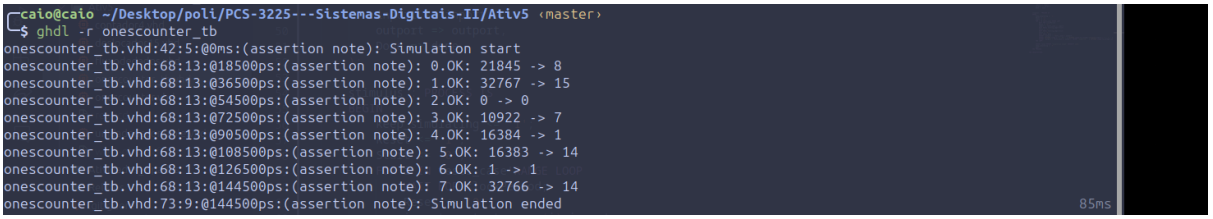












6. Análise da verificação funcional

O intuito do projeto era a criação de um contador de 1, dado uma entrada de 15 bits quantos uns tinham nessa entrada. Sendo assim, como visto nas imagens acima, o grupo atingiu o resultado esperado. Os códigos em VHDL utilizados estarão presentes no anexo.