MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES

Aluno: Vinícius da Luz Souza Data: 07/07/2019
Matéria: DLP II - 2019/1

Prática Máquina de Estados
Relatório A4

Este relatório tem por objetivo descrever as atividades realizadas no desenvolvimento de um exercício prático de máquinas de estados. Serão exibidos os diagramas de máquina de estado que foram desenvolvidos (diagrama de bolha e diagrama ASM), bem como o diagrama RTL final da implementação.

Para desenvolvimento do exercício, foi escolhido um circuito denominado Árbitro, cujo objetivo é gerenciar o acesso a um recurso compartilhado no sistema. O circuito foi desenvolvido em hardware FPGA, através da descrição VHDL, utilizando o *software* Quartus II. No Quartus, o kit FPGA escolhido foi DE2-115 da família Cyclone IV E (Altera - TERASIC), *device* EP4CE115F29C7.

O Circuito Árbitro

O circuito escolhido para desenvolvimento da atividade tem por objetivo garantir o acesso exclusivo e ordenado a um recurso que é compartilhado no sistema. O compartilhamento de recursos é altamente utilizado em sistemas eletrônicos, no qual um mesmo recurso é compartilhado entre vários processos. Podemos tomar como exemplo, o uso de uma placa de rede em um sistema computacional. Considerando que exista somente uma placa de rede no computador e vários processos querendo utilizá-la, teremos problemas na alocação deste recurso caso não haja um árbitro gerenciando o acesso ao dispositivo. Esse árbitro pode ser desenvolvido tanto em *software* quanto em *hardware* e, neste relatório, abordaremos sua implementação em hardware.

O árbitro desenvolvido possui capacidade para atender até 4 solicitantes, rotacionando sua alocação, ou seja, não há prioridade para alocação do recurso entre os solicitantes. Além disso, para que um solicitante não tome o recurso por tempo indeterminado, foi adicionada uma entrada de *timeout*, em que, quando acionada, o recurso é liberado. O gerenciamento do tempo de timeout não foi implementado neste circuito, ficando a cargo de um dispositivo externo.

A figura a seguir exibe o diagrama projetado para a máquina de estados.

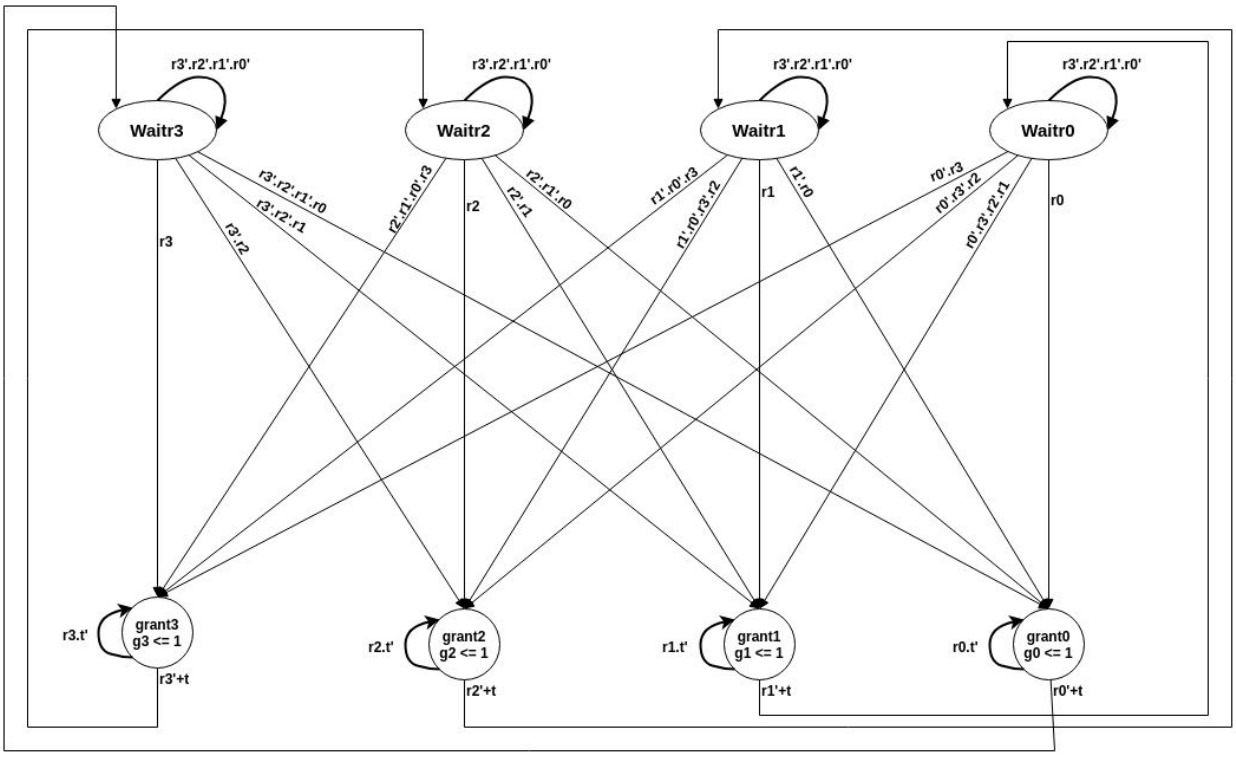

Figura 1 - Diagrama de máquina de estado (bolha) do circuito Árbitro

Visando a prática didática, o diagrama de bolhas foi convertido em um gráfico ASM *(Algorithmic State Machine)*. Essa forma de visualização é, em geral, mais utilizada em grandes máquinas de estados, em que a representação com diagrama de bolhas pode ficar confusa ou até mesmo “suja”, diante dos vários estados necessários na máquina desenvolvida.

Nesse gráfico, de acordo com a opinião do autor deste relatório, fica mais clara a visualização dos estados, bem como suas saídas *mealy* e *moore*, facilitando a implementação do código.

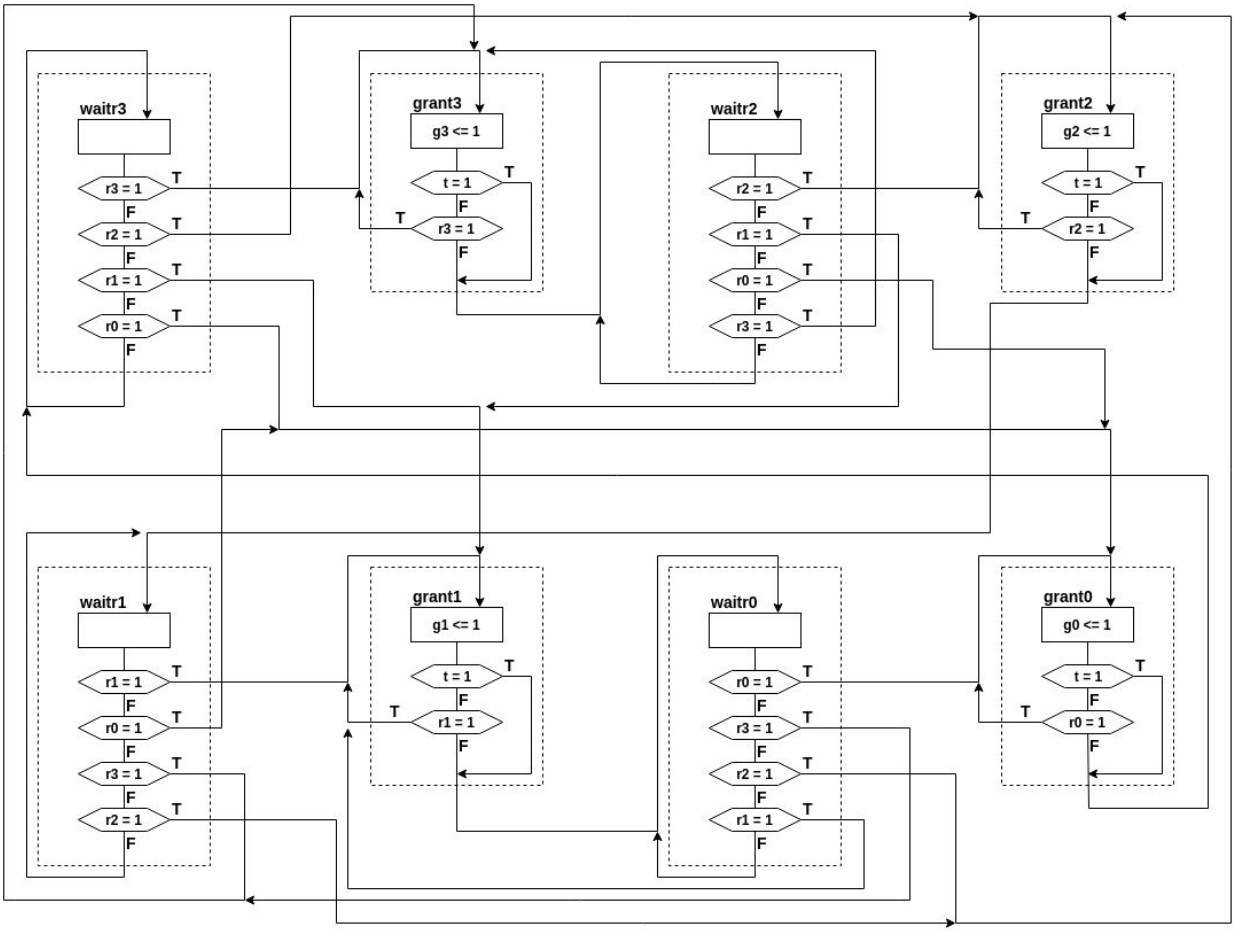

Figura 2 - Gráfico ASM do circuito Árbitro

De acordo com os gráficos apresentados, é possível constatar o funcionamento dos estados da máquina, sendo eles, os estados de *wait* e *grant* (esperar e conceder). É possível explicá-los da seguinte forma:

- **Wait (esperar):** neste estado, a máquina aguarda uma solicitação de acesso ao recurso, dando preferência ao solicitante atrelado àquele estado de *wait* (por exemplo, no estado *wait2*, a máquina dará preferência a liberação *grant2*). Ela só sai deste estado quando algum solicitante requerer acesso ao recurso.

- **Grant (conceder):** neste estado, o acesso foi concedido a algum solicitante e há duas maneiras de sair dele: caso o solicitante libere o recurso ou caso ocorra um *timeout*. Neste caso, o próximo estado deve ser o *wait* do próximo solicitante (por exemplo, em *grant2,* o próximo estado será *wait1*). Desta forma, a máquina fornece acesso de forma cíclica (justa).

O gráfico a seguir exibe a implementação gerada com descrição VHDL para a máquina de estados mencionada.

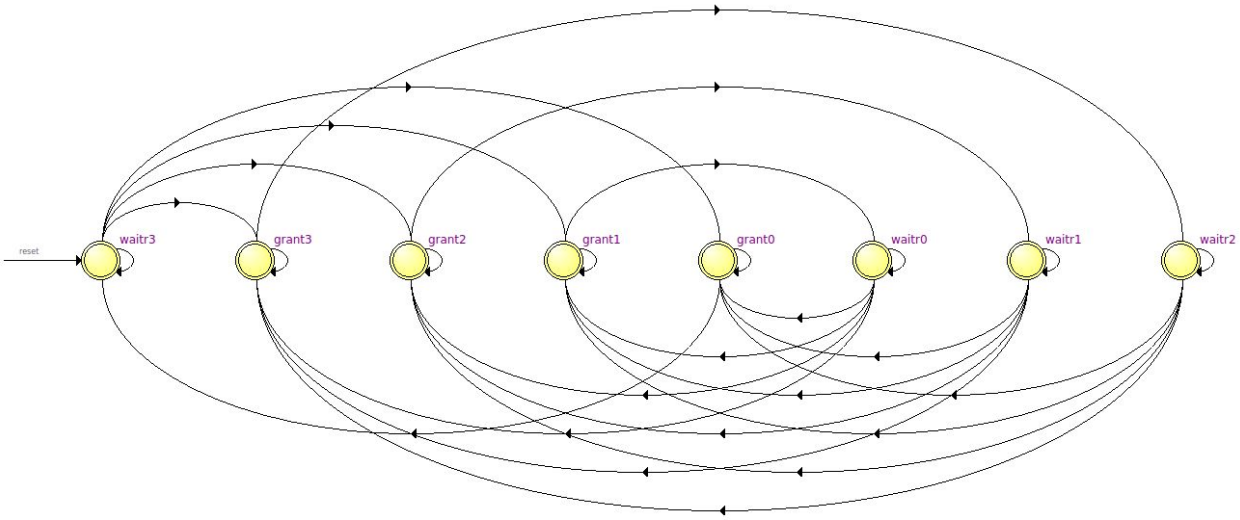

Figura 3 - Máquina de estados gerada em VHDL

De acordo com o gráfico da figura 3, é possível constatarmos que a máquina de estados foi implementada corretamente, já que o diagrama gerado confere com o que foi projetado, como mostrado na figura 1.

A figura a seguir exibe o diagrama RTL completo do sistema.

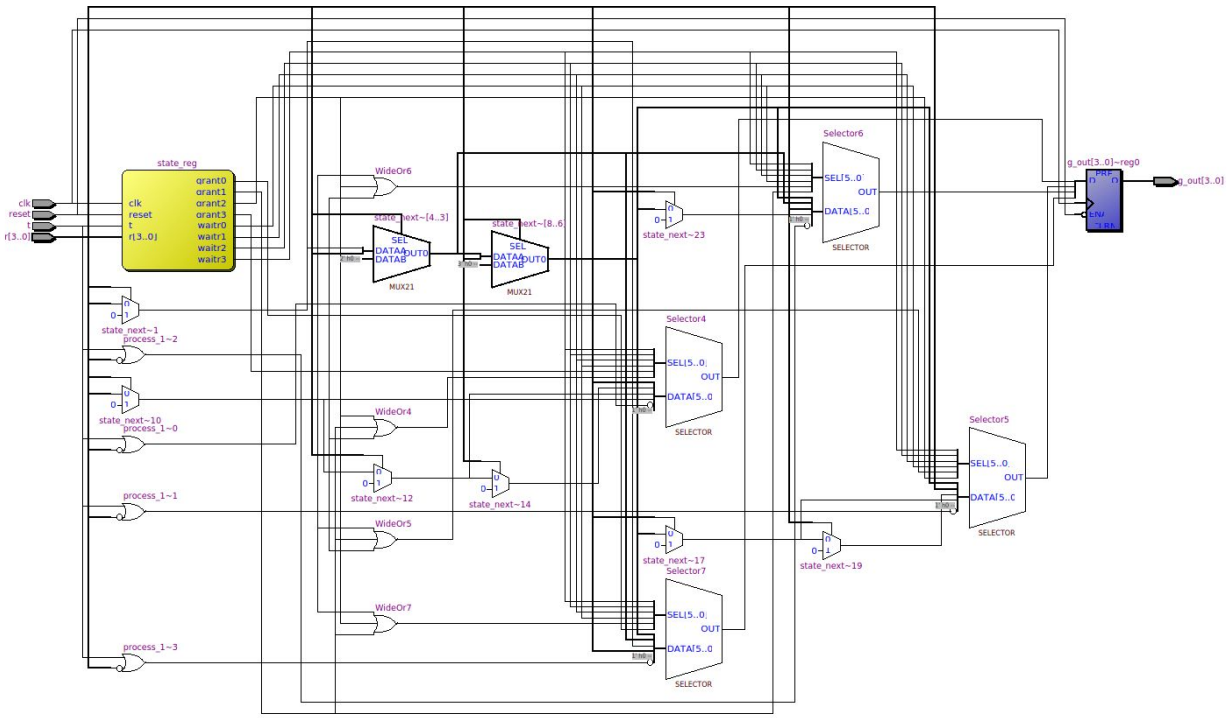

Figura 4 - Diagrama RTL completo do sistema Árbitro

De acordo com o diagrama exibido na figura 4, é possível identificarmos a presença de um *FlipFlop* na saída do circuito. Esse *FlipFlop* está presente devido a implementação da lógica *"look-ahead output buffer",* que visa proteger a saída de um *glitch*.

Um *glitch* é um sinal intermediário que acontece enquanto o valor é final calculado e antes de estar estabilizado, ou seja, é, normalmente, um sinal indesejado. Desta forma, devido a lógica de saída estar ligada diretamente a saída do circuito, os sinais de *glitch* podem ser percebidos e causarem algum comportamento indesejado no circuito subsequente. Para corrigir este problema, é possível adicionarmos um *FlipFlop* diretamente na lógica de saída, porém incrementaríamos um ciclo de clock na resposta do circuito. Outra forma, seria utilizarmos a lógica *"clever state machine"*, que agrega informações nos estados do circuito, porém esta implementação aumenta a complexidade, tornando a manutenção (eventual ampliação do circuito) mais trabalhosa. Sendo assim, a técnica escolhida foi a *"look-ahead output buffer"*, que adiciona um *FlipFlop* na saída, mas assume como entrada o próximo estado, já deixando calculado o valor de saída na entrada do último FlipFlop do circuito e, desta forma, economizando um ciclo de clock que seria perdido se simplesmente adicionássemos um *FlipFlop* na borda do circuito.

A figura a seguir demonstra a implementação da lógica *"look-ahead output buffer".*

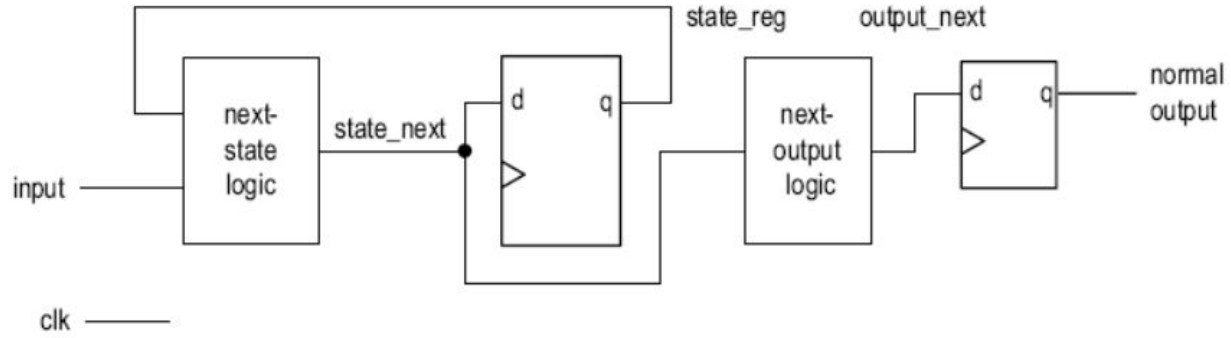

Figura 5 - Lógica *look-ahead output buffer*

Após a implementação do circuito, foram executados testes em seu funcionamento e os resultados podem ser observados nas figura as seguir.

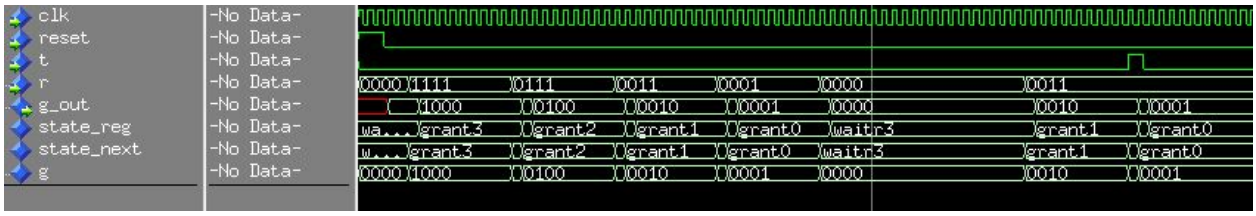

Figura 6 - Simulação circuito árbitro parte 1

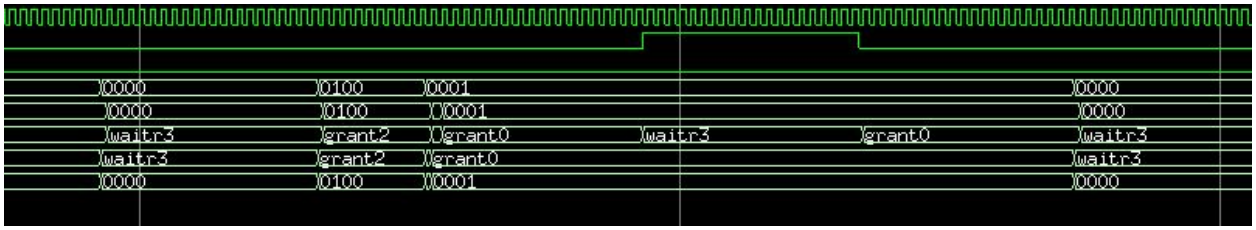

Figura 7 - Simulação circuito árbitro parte 2

De acordo com os testes gerados, é possível visualizar que o circuito está concedendo os acessos de forma correta no estado de *wait*, fazendo-o de forma cíclica entre os solicitantes. Também é possível visualizar que, no estado de *grant*, o circuito só é liberado quando o solicitante desaloca o recurso ou quando ocorre um *timeout*, que também está funcionando corretamente.

Além dos testes de funcionamento do circuito, a partir da simulação, observamos o correto funcionamento da lógica *"look-ahead output buffer"*, pois, quando uma entrada é alterada (sinal *r*), o próximo estado já é calculado e disponibilizado na saída intermediária (sinal *g*). Em seguida, com o bater do *clock*, o estado é disponibilizado na saída do circuito (sinal *g_out*), preservando o ciclo de *clock*.

Obs.1: para simular, execute o comando *"do test_arbiter.do"* no simulink.

Obs.2: para funcionamento da simulação, o arquivo *test_arbiter.do* deve estar no caminho: <*diretorio_do_projeto*>*/simulation/modelsim* .

Conclusão

A partir da atividade desenvolvida, foi possível projetar uma máquina de estados com base em um gráfico de bolha e um gráfico ASM. Foi necessário desenhar estes gráficos, promovendo a fixação do conhecimento acerca desses diagramas.

Além disso, o desenvolvimento da máquina de estados proporcionou um melhor entendimento sobre as saídas *Mealy* e *Moore*, bem como uma melhor compreensão sobre o funcionamento da lógica *"look-ahead output buffer"*, necessária para a prevenção contra os *glitches* na saída do circuito.