Sistemas Reconfiguráveis – Eng. de Computação
Especificações para o terceiro trabalho

2º semestre de 2024

1. Objetivo

Descrever em linguagem VHDL, descrever códigos, simular o funcionamento e comentar os resultados da simulação dos blocos especificados nos itens 2 e 3 a seguir. **Os blocos são projetos independentes entre si** e deverão ser criados em pastas separadas. A documentação de cada bloco deverá ser feita em **capítulos separados** no relatório do trabalho. Poderá ser usado código concorrente ou sequencial, e todas as entradas e saídas deverão ser do tipo **STD_LOGIC** ou **STD_LOGIC_VECTOR**. Obrigatoriamente deverá ser usada a **versão 9.1sp2** do *software* Quartus.

Deverá ser entregue um relatório do trabalho na forma de um documento padrão ABNT para trabalhos acadêmicos (capa, folha de rosto, índice de figuras, etc, etc) em um arquivo no formato pdf, via Canvas. Nesse relatório, cada projeto deverá estar em um capítulo próprio. O relatório deverá conter os códigos VHDL completos com comentário e descrição funcional; figuras referentes aos sinais de entrada e saída (formas de onda) para simulação (casos de testes) e sua descrição. Os casos de teste devem ser suficientes para demonstrar o correto funcionamento dos circuitos sintetizados.

Além do relatório, deverá ser entregue um arquivo compactado (.zip ou .rar), com todos os arquivos dos projetos gerados no ambiente Quartus. Nesse arquivo, cada projeto deverá estar em uma pasta própria.

2. Port_io

O bloco port_io tem dois lados. Um lado possui entradas e saídas para comunicação com o barramento interno do processador e o outro, entradas ou saídas para interfaceamento externo ao processador, com a direção definida através de um registrador.

2.1. Entradas e saídas do lado do processador

nrst	Entrada de *reset* assíncrono. Quando ativada (nível lógico baixo), o registrador **port_reg** deverá ter todos os seus bits zerados e o registrador **tris_reg** deverá ter todos os seus bits setados (= '1'). Esta entrada tem preferência sobre todas as outras.
clk_in	Entrada de *clock* do sistema. Todos os registradores devem ser escritos síncronos com a borda de subida desse *clock*.
abus_in[8..0]	Entrada de endereçamento para os registradores internos. Os endereços dos registradores são especificados através de 4 "generic", conforme especificado no item 2.4.
dbus_in[7..0]	Barramento de entrada de dados, com 8 bits, para a escrita nos registradores, com habilitação através de wr_en, e endereçamento por abus_in.
wr_en	Entrada de habilitação para escrita nos registradores. Quando ativada (nível lógico alto), o registrador **tris_reg** ou **port_reg** será escrito, síncrono com clk_in, com o valor de dbus_in, desde que o endereço correspondente esteja presente em abus_in. Caso contrário, nenhuma ação será efetuada.
rd_en	Entrada de habilitação para leitura. Quando ativada (nível lógico alto), o barramento dbus_out deverá receber o conteúdo do registrador **tris_reg** ou do **latch** de entrada, desde que o endereço correspondente esteja presente em abus_in. Quando essa entrada estiver desativada ou quando o endereço não corresponder, a saída deverá ficar em alta impedância ("ZZZZZZZZ").
dbus_out[7..0]	Barramento de saída de dados, com 8 bits. Habilitação através de rd_en e endereçamento por abus_in. Durante as operações de leitura, comporta-se como saída. Quando não em uso, fica em alta impedância ("ZZZZZZZZ"). A operação de leitura deve ser completamente combinacional, ou seja, não depende de transição de clk_in.

2.2. Lado da porta

port_io[7..0] Porta bidirecional ou seja, modo = INOUT, com 8 bits. A direção será configurada, bit a bit, através do registrador **tris_reg**.

2.3. Endereçamento

Para especificar os endereços dos registradores, devem ser incluídos na "*entity*" do código quatro "*generic*", todos eles de 9 bits:

port_addr	Especifica o endereço de escrita no registrador **port_reg** (quando wr_en = '1') ou para leitura do **latch** (quando rd_en = '1').
tris_addr	Especifica o endereço de escrita no registrador **tris_reg** (quando wr_en = '1') ou para leitura do mesmo registrador (quando rd_en = '1').
alt_port_addr	Endereço alternativo a port_addr. Quando não em uso, deve ser configurado com o mesmo valor de port_addr.
alt_tris_addr	Endereço alternativo a tris_addr. Quando não em uso, deve ser configurado com o mesmo valor de tris_addr.

2.4. Descrição de funcionamento

Existem dois registradores nesse bloco. O registrador **tris_reg** configura a função de cada bit da porta, se entrada ou saída. Inicialmente, após o *reset*, o **tris_reg** tem todos os bits em '1', configurando, assim, todos os bits da porta como entrada. Para configurar um determinado bit da porta como saída, deve ser escrito '0' no bit correspondente do **tris_reg**. A operação de escrita nesse registrador acontece na borda de subida de clk_in quando (abus_in = tris_addr ou abus_in = alt_tris_addr) e wr_en = '1'. O conteúdo de **tris_reg** é lido quando (abus_in = tris_addr ou abus_in = alt_tris_addr) e rd_en = '1'.

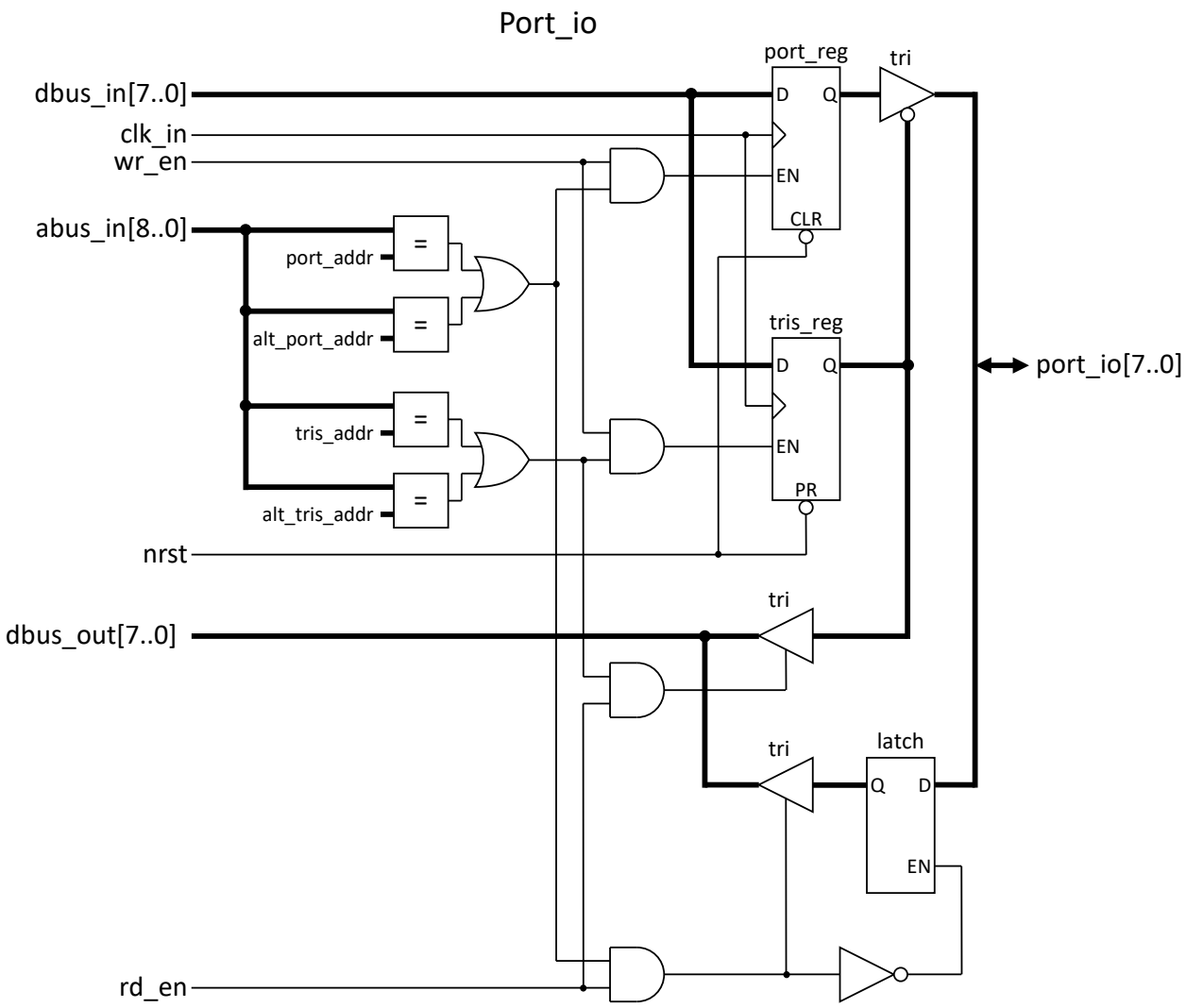
O registrador **port_reg** armazena os dados de saída da porta. A operação de escrita nesse registrador acontece na borda de subida de clk_in quando (abus_in = port_addr ou abus_in = alt_port_addr) e wr_en = '1'. Após o *reset*, todos os bits ficam zerados. Estando um determinado bit da porta configurado como saída (através da escrita em **tris_reg**), o valor armazenado nesse bit no registrador **port_reg** será o valor da porta no bit correspondente. Estando configurado como entrada, o valor nesse bit no registrador **port_reg** não terá efeito algum.

Por exemplo, se o registrador **tris_reg[7..0]** tiver sido escrito com "00001111" e o registrador **port_reg[7..0]** tiver sido escrito com "10101111", nos bits **[7..4]** de **port_io** haverá uma saída com o valor "1010". Os bits **port_io[3..0]** dependerão do que estiver ligado a eles, pois estão configurados como entrada.

A operação de leitura nos mesmos endereços que permitem a escrita em **port_reg**, não lê o registrador **port_reg**, mas um sim *latch* que trava os valores presentes na porta. Normalmente esse *latch* está no estado transparente. Na operação de leitura, que acontece quando (abus_in = port_addr ou abus_in = alt_port_addr) e rd_en = '1', o *latch* é travado.

2.5. Diagrama simplificado

O diagrama é apresentado na próxima página


Port_io
dbus_in[7..0]
clk_in
wr_en
abus_in[8..0]
port_addr
alt_port_addr
tris_addr
alt_tris_addr
=
=
=
=
nrst
dbus_out[7..0]
rd_en
port_reg
D
Q
EN
CLR
tri
tris_reg
D
Q
EN
PR
port_io[7..0]
tri
tri
latch
Q
D
EN

3. RAM_mem

O bloco RAM_mem compreende quatro áreas de memória, chamadas de *general purpose registers* sendo três de 80 (oitenta) bytes e uma com 16 (dezesseis) bytes.

3.1. Entradas

nrst	Entrada de *reset* assíncrono. Quando ativada (nível lógico baixo), todos os bits da memória RAM deverão ser zerados. Esta entrada tem preferência sobre todas as outras.
clk_in	Entrada de *clock* do sistema. A escrita na memória acontece na borda de subida do *clock*, desde que habilitada.
abus_in[8..0]	Entrada de endereçamento. Aponta a célula de memória a ser escrita ou lida, desde que habilitada. As faixas de endereço para cada área de memória estão definidas no item 3.3.
dbus_in[7..0]	Entrada de dados para escrita na memória, com habilitação através de wr_en, e endereçamento por abus_in.
wr_en	Entrada de habilitação para escrita na memória. Quando ativada (nível lógico alto), a memória será escrita, síncrona com clk_in, com o valor de dbus_in, desde que o endereço correspondente esteja presente em abus_in, conforme especificado no item 3.3. Caso contrário, nenhuma ação será efetuada.
rd_en	Entrada de habilitação para leitura. Quando ativada (nível lógico alto), a saída dbus_out deverá corresponder ao conteúdo da memória endereçada. Quando desativada ou quando o endereço não corresponder ao especificado no item 3.3, a saída deverá ficar em alta impedância (“ZZZZZZZZ”).

3.2. Saídas

dbus_out[7..0]	Barramento de saída de dados, com 8 bits. Habilitação através de rd_en e endereçamento por abus_in. Durante as operações de leitura, comporta-se como saída. Quando não em uso, fica em alta impedância (“ZZZZZZZZ”). A operação de leitura deve ser completamente combinacional, ou seja, não depende de transição de clk_in.

3.3. Endereçamento

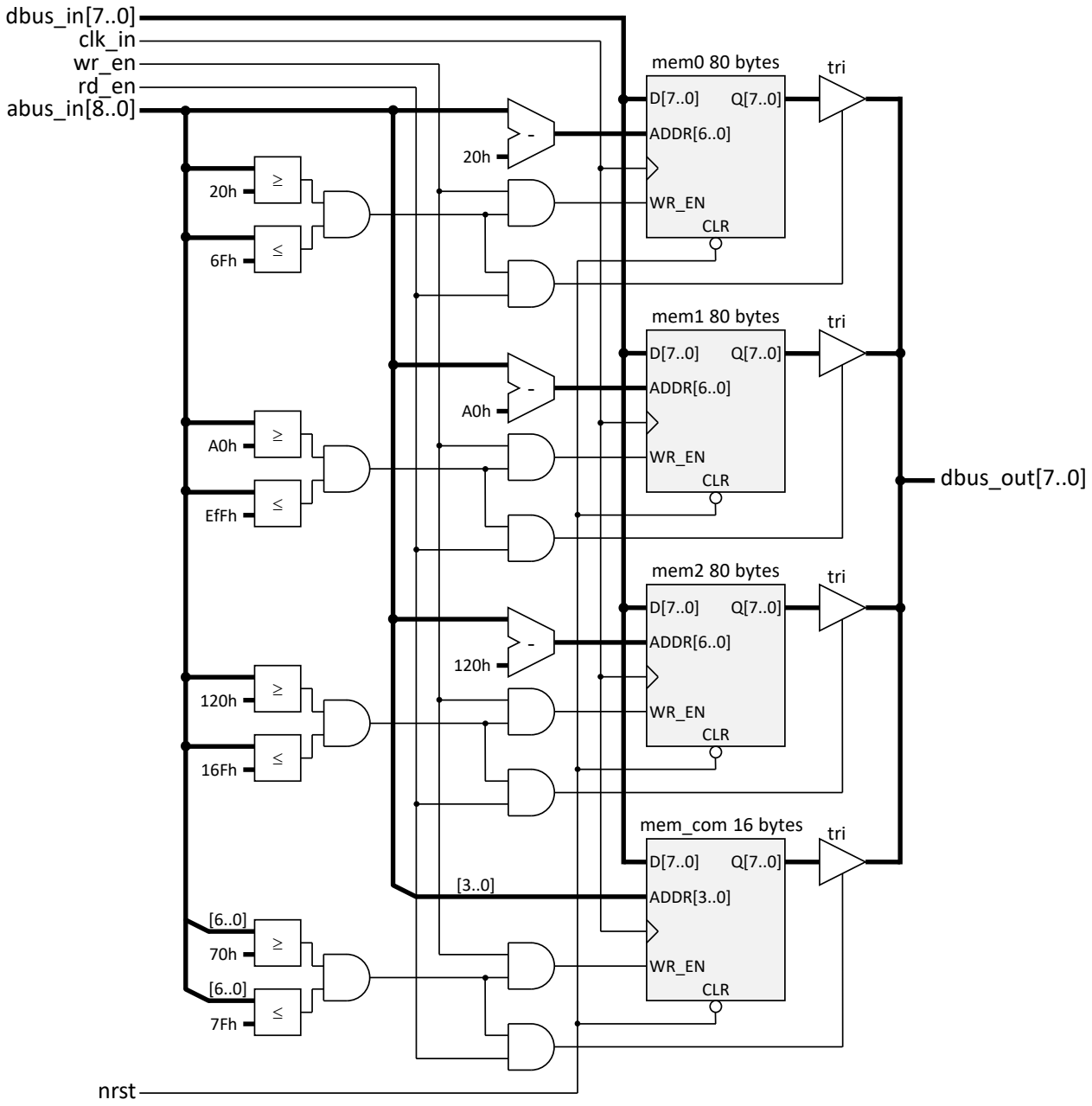
mem0	020h ~ 06Fh (80 bytes). Em decimal: 32 a 111
mem1	0A0h ~ 0EFh (80 bytes). Em decimal: 160 a 239
mem2	120h ~16Fh (80 bytes). Em decimal: 288 a 367
mem_com	070h ~ 07Fh (16 bytes). Em decimal: 112 a 127

Obs. A área de memória mem_com também pode ser endereçada através dos endereços 0F0h ~ 0FFh, 170h ~ 17Fh ou 1F0h ~1FFh. Dessa forma os bits 8 e 7 de abus_in não importam para o endereçamento dessa área específica, sendo utilizados apenas os bits 6 a 0.

3.4. Diagrama simplificado

Na próxima página

RAM_mem


dbus_in[7..0]
clk_in
wr_en
rd_en
abus_in[8..0]
nrst
20h
6Fh
A0h
EfFh
120h
16Fh
[6..0]
70h
[6..0]
7Fh
20h
A0h
120h
[3..0]
mem0 80 bytes
mem1 80 bytes
mem2 80 bytes
mem_com 16 bytes
D[7..0]
Q[7..0]
ADDR[6..0]
ADDR[3..0]
WR_EN
CLR
tri
dbus_out[7..0]