Sistemas Reconfiguráveis – Eng. de Computação
Especificações para o segundo projeto

1º semestre de 2022

1. Objetivo

Descrever em linguagem VHDL, comentar códigos, simular o funcionamento e comentar os resultados da simulação dos blocos especificados nos itens 2, 3, 4, 5 e 6 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**.

O trabalho deverá ser entregue em um documento padrão ABNT para trabalhos acadêmicos (capa, folha de rosto, sumário, índice de figuras, introdução, codificação, simulação, conclusão etc.). O trabalho 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.

2. W_reg

O bloco W_reg tem um registrador de 8 bits.

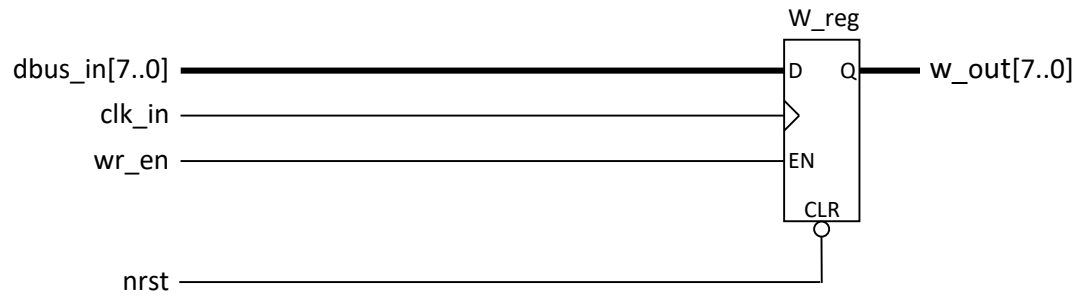
2.1. Entradas

nrst	Entrada de *reset* assíncrono. Quando ativada (nível lógico baixo), todos os bits do registrador deverão ser zerados. Esta entrada tem preferência sobre todas as outras.
clk_in	Entrada de *clock* para escrita no registrador. A escrita acontece na borda de subida do *clock*, desde que habilitada.
d_in[7..0]	Entrada de dados para escrita no registrador, com habilitação através de wr_en.
wr_en	Entrada de habilitação para escrita no registrador. Quando ativada (nível lógico alto), o registrador será escrito, síncrono com clk_in, com o valor de d_in. Caso contrário, nenhuma ação será efetuada.

2.2. Saída

w_out[7..0]	Saída de dados. Essa saída está sempre ativa, não dependendo de habilitação.

2.3. Diagrama



3. FSR_reg

O bloco FSR_reg tem um registrador de 8 bits.

3.1. Entradas

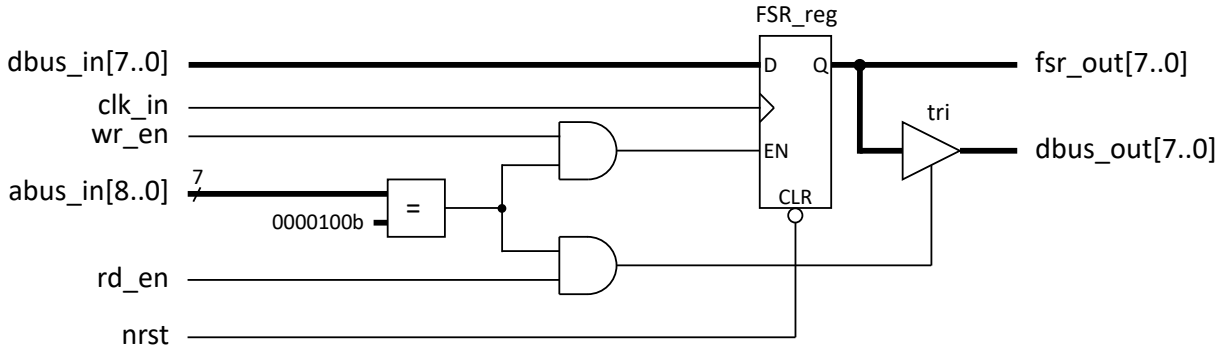
nrst	Entrada de *reset* assíncrono. Quando ativada (nível lógico baixo), todos os bits do registrador deverão ser zerados. Esta entrada tem preferência sobre todas as outras.
clk_in	Entrada de *clock* para escrita no registrador. A escrita acontece na borda de subida do *clock*, desde que habilitada.
abus_in[8..0]	Entrada de endereçamento. O registrador deve ser escrito ou lido quando abus_in[6..0] = "0000100". Dessa forma, apenas os 7 bits menos significativos de abus_in devem ser usados. Os outros dois bits não importam.
dbus_in[7..0]	Entrada de dados para escrita no registrador, com habilitação através de wr_en, e endereçamento por abus_in.
wr_en	Entrada de habilitação para escrita no registrador. Quando ativada (nível lógico alto), o registrador será escrito, síncrono com clk_in, com o valor de dbus_in, desde que o endereço correspondente, conforme especificado acima, esteja presente em abus_in. 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 do registrador, desde que o endereço correspondente, conforme especificado acima, esteja presente em abus_in. Quando desativada ou quando o endereço não corresponder ao especificado acima, a saída deverá ficar em alta impedância ("ZZZZZZZZ"). A operação de leitura deve ser completamente combinacional, ou seja, não depende de transição de clk_in.

Obs.: As operações de escrita e leitura são independentes entre si. Assim, os sinais wr_en e rd_en poderão estar ativos simultaneamente.

3.2. Saídas

dbus_out[7..0]	Saída de dados lidos com habilitação através de rd_en e endereçamento por abus_in. Quando não usada, deverá ficar em alta impedância ("ZZZZZZZZ").
fsr_out[7..0]	Saída correspondente ao registrador. Essa saída está sempre ativa, não dependendo de habilitação.

3.3. Diagrama



4. Status_reg

O bloco Status_reg tem um registrador de 8 bits.

4.1. Entradas

nrst	Entrada de *reset* assíncrono. Quando ativada (nível lógico baixo), todos os bits do registrador deverão ser zerados. Esta entrada tem preferência sobre todas as outras.
clk_in	Entrada de *clock* para escrita no registrador. A escrita em todos os registradores acontece na borda de subida do *clock*, desde que habilitada.
abus_in[8..0]	Entrada de endereçamento. O registrador deve ser escrito ou lido quando abus_in[6..0] = "0000011". Dessa forma, apenas os 7 bits menos significativos de abus_in devem ser usados. Os outros dois bits não importam.
dbus_in[7..0]	Entrada de dados para escrita no registrador, com habilitação através de wr_en, e endereçamento por abus_in.
wr_en	Entrada de habilitação para escrita no registrador. Quando ativada (nível lógico alto), o registrador será escrito, síncrono com clk_in, com o valor de dbus_in, desde que o endereço correspondente, conforme especificado acima, esteja presente em abus_in. 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 do registrador, exceto os bits 4 e 3, que deverão ser sempre lidos como '1'. Quando desativada ou quando o endereço não corresponder ao especificado acima, a saída deverá ficar em alta impedância ("ZZZZZZZZ").
z_in	Entrada de dado para escrita no bit 2 do registrador, com habilitação através de z_wr_en.
dc_in	Entrada de dado para escrita no bit 1 do registrador, com habilitação através de dc_wr_en.
c_in	Entrada de dado para escrita no bit 0 do registrador, com habilitação através de c_wr_en.
z_wr_en	Entrada para habilitação da escrita no bit 2 do registrador. Quando ativada (nível lógico alto), o bit 2 do registrador será escrito, síncrono com clk_in, com o valor de z_in. Essa entrada tem preferência sobre wr_en.
dc_wr_en	Entrada para habilitação da escrita no bit 1 do registrador. Quando ativada (nível lógico alto), o bit 1 do registrador será escrito, síncrono com clk_in, com o valor de dc_in. Essa entrada tem preferência sobre wr_en.
c_wr_en	Entrada para habilitação da escrita no bit 0 do registrador. Quando ativada (nível lógico alto), o bit 0 do registrador será escrito, síncrono com clk_in, com o valor de c_in. Essa entrada tem preferência sobre wr_en.

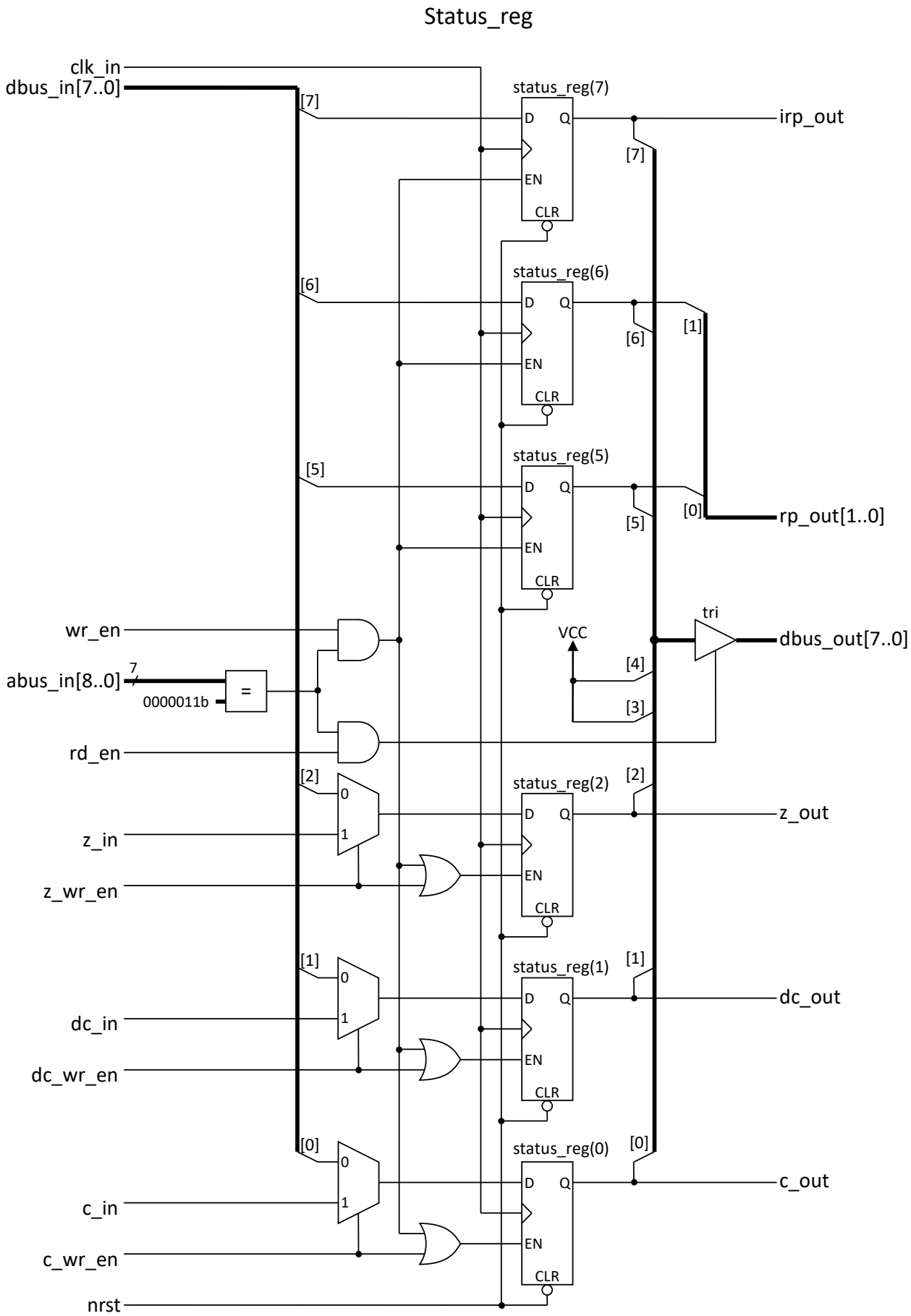
Obs. 1: As operações de escrita e leitura são independentes entre si. Assim, os sinais wr_en e rd_en poderão estar ativos simultaneamente ou não.

Obs. 2: As operações de escrita nos bits 0, 1 e 2 através dos sinais de habilitação z_wr_en, dc_wr_en e c_wr_en são independentes entre si. Assim, esses sinais poderão ocorrer simultaneamente ou não.

4.2. Saídas

dbus_out[7..0]	Saída de dados lidos com habilitação através de rd_en e endereçamento por abus_in. Quando não usada, deverá ficar em alta impedância ("ZZZZZZZZ").
irp_out	Saída correspondente ao bit 7 do registrador. Essa saída está sempre ativa, não dependendo de habilitação.
rp_out[1..0]	Saída correspondente aos bits 6 e 5 do registrador. Essa saída está sempre ativa, não dependendo de habilitação.
z_out	Saída correspondente ao bit 2 do registrador. Essa saída está sempre ativa, não dependendo de habilitação.
dc_out	Saída correspondente ao bit 1 do registrador. Essa saída está sempre ativa, não dependendo de habilitação.
c_out	Saída correspondente ao bit 0 do registrador. Essa saída está sempre ativa, não dependendo de habilitação.

4.3. Diagrama


Status_reg
clk_in
dbus_in[7..0]
[7]
status_reg(7)
D
Q
EN
CLR
irp_out
[7]
[6]
status_reg(6)
[6]
[1]
[5]
status_reg(5)
[5]
[0]
rp_out[1..0]
wr_en
abus_in[8..0]
7
0000011b
=
rd_en
VCC
tri
dbus_out[7..0]
[4]
[3]
[2]
0
1
z_in
z_wr_en
status_reg(2)
[2]
z_out
[1]
dc_in
dc_wr_en
status_reg(1)
[1]
dc_out
[0]
c_in
c_wr_en
status_reg(0)
[0]
c_out
nrst

5. Stack

O bloco Stack tem um conjunto de 8 registradores de 13 bits.

5.1. Entradas

nrst	Entrada de *reset* assíncrono. Quando ativada (nível lógico baixo), todos os bits dos registradores deverão ser zerados. Esta entrada tem preferência sobre todas as outras.
clk_in	Entrada de *clock* para escrita nos registradores. A escrita acontece na borda de subida do *clock*, desde que habilitada.
stack_in[12..0]	Entrada de dados para a pilha.
stack_push	Entrada de habilitação para colocar valores na pilha. Quando ativada (nível lógico alto), o valor presente na entrada stack_in deve ser escrito no primeiro registrador da pilha. Ao mesmo tempo, o segundo registrador recebe o conteúdo do primeiro, e assim por diante, até o oitavo registrador, que recebe o conteúdo do sétimo registrador.
stack_pop	Entrada de habilitação para retirar valores da pilha. Quando ativada (nível lógico alto) o conteúdo do segundo registrador deve ser transferido para o primeiro, o do terceiro registrador para a segundo, e assim por diante, até o sétimo registrador, que recebe o conteúdo do oitavo. Esse, por sua vez, recebe o valor zero (todos os bits iguais a '0').

Obs. 1: Normalmente, ao operações de *push* e *pop* não ocorrem simultaneamente. No entanto, caso as entradas stack_push e stack_pop sejam ativadas ao mesmo tempo, a operação *pop* deve ter preferência.

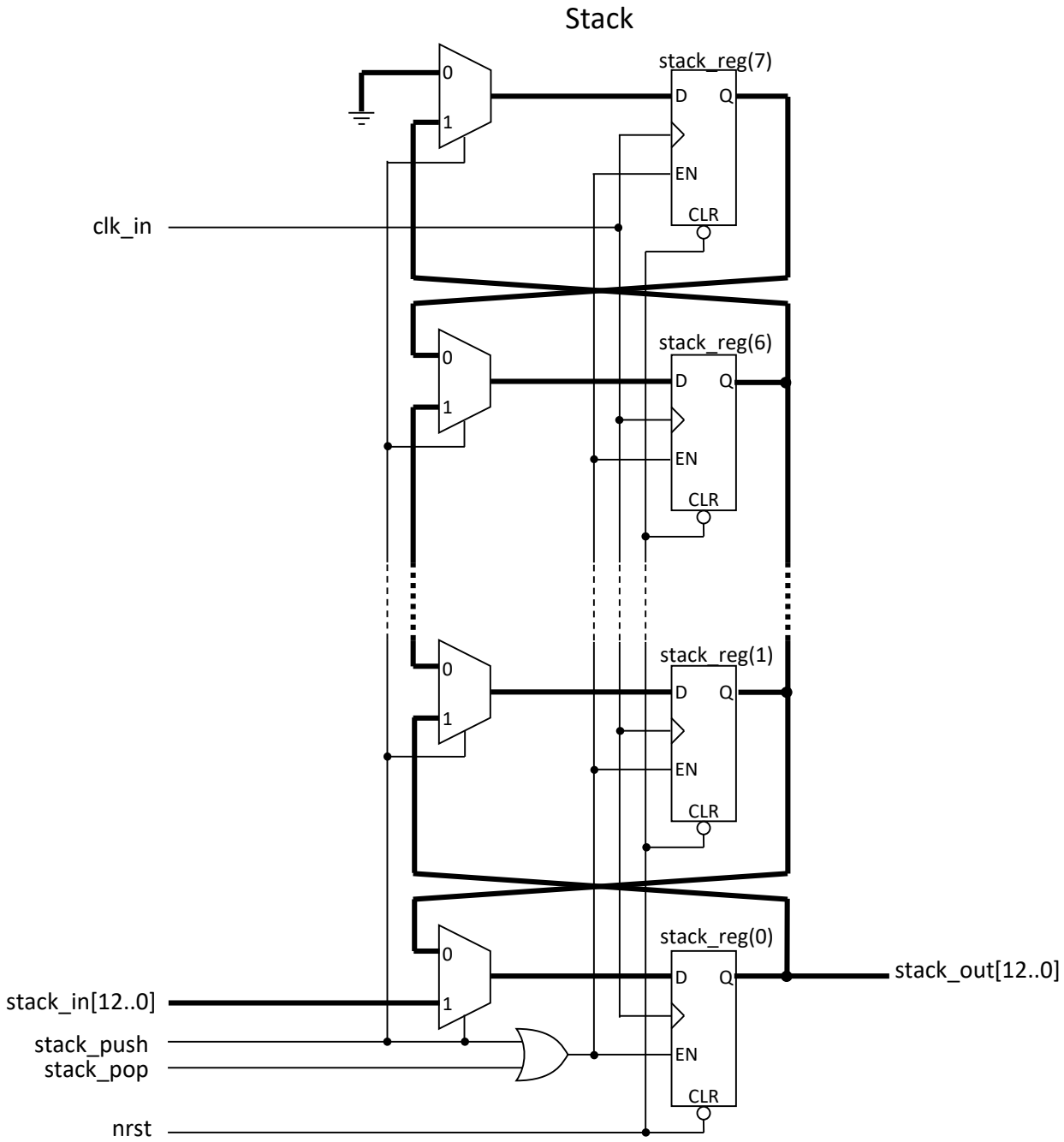
Obs. 2: Os valores armazenados na pilha através da operação *push* podem ser recuperados através da operação *pop*, na ordem inversa com que foram escritos (primeiro a entrar = último a sair). No entanto, mais de 8 operações *push* em sequência farão com que o conteúdo mais antigo da pilha seja perdido (chamado de *stack overflow*).

5.2. Saídas

stack_out[12..0]	Saída correspondente à primeira posição da pilha. Essa saída está sempre ativa, não dependendo de habilitação

5.3. Diagrama

Diagrama na próxima página



6. PC_reg

O bloco PC_reg tem um registrador de 13 bits (chamado PC - *program counter*) e um outro registrador, de 8 bits, chamado de PCLATH. Além disso, usa a pilha (*stack*) desenvolvida no item 5. A parte baixa do registrador PC, bits 7 a 0, é chamada de PCL.

6.1. Entradas

nrst	Entrada de *reset* assíncrono. Quando ativada (nível lógico baixo), os registradores PC e PCLATH deverão ser zerados. Esta entrada tem preferência sobre todas as outras.
clk_in	Entrada de *clock* para incremento e carga dos registradores. O incremento ou a carga acontece na borda de subida do *clock*, desde que habilitados.
addr_in[10..0]	Entrada de dados para carga no registrador PC, com habilitação através de load_en.

abus_in[8..0] Entrada de endereçamento para PCL e para o registrador PCLATH. PCL deve ser escrito ou lido quando abus_in[6..0] = “0000010”. O registrador PCLATH deve ser escrito ou lido quando abus_in[6..0] = “0001010”. Dessa forma, nos dois casos, apenas os 7 bits menos significativos de abus_in devem ser usados. Os outros dois bits não importam.

dbus_in[7..0] Entrada de dados para escrita em PCL e PCLATH, com habilitação através de wr_en, e endereçamento por abus_in.

inc_pc Entrada de habilitação para incremento. Quando ativada (nível lógico alto), o registrador PC deverá ser incrementado, síncrono com clk_in. Essa entrada tem preferência sobre load_pc e wr_en.

load_pc Entrada de habilitação para carga. Quando ativada (nível lógico alto), os bits 10 a 0 do registrador PC devem ser carregados com o valor de addr_in e, ao mesmo tempo, os bits 12 a 11 de PC devem ser carregados com o conteúdo de PCLATH[4..3]. Esta entrada tem preferência sobre wr_ena.

wr_en Entrada de habilitação para escrita nos registradores. Quando ativada (nível lógico alto), PCL ou o registrador PCLATH será escrito, síncrono com clk_in, com o valor de dbus_in, desde que o endereço correspondente, conforme especificado acima, esteja presente em abus_in. Caso contrário, nenhuma ação será efetuada. Ao mesmo tempo que uma operação de escrita em PCL, os bits 12 a 8 do registrador PC devem ser escritos com o conteúdo dos 5 bits menos significativos de PCLATH (bits 4 a 0).

rd_en Entrada de habilitação para leitura. Quando ativada (nível lógico alto), a saída dbus_out deverá corresponder ao conteúdo da parte baixa de PC (bits 7 a 0 – chamados de PCL) ou PCLATH, desde que o endereço correspondente, conforme especificado acima, esteja presente em abus_in. Quando desativada ou quando o endereço não corresponder a PCL ou PCLATH, a saída deverá ficar em alta impedância (“ZZZZZZZZ”).

stack_push Entrada de habilitação para colocar valores na pilha. Quando ativada (nível lógico alto), o conteúdo atual de PC deve ser transferido para a primeira posição da pilha. O funcionamento da pilha é independente das entradas inc_pc, load_en e wr_en, podendo, inclusive, ser ativada ao mesmo tempo que a entrada load_pc.

stack_pop Entrada de habilitação para retirar valores da pilha. Quando ativada (nível lógico alto), o registrador PC deve ser carregado com o conteúdo da primeira posição da pilha. Essa entrada tem preferência sobre inc_pc, load_en e wr_en.

6.2. Saídas

nextpc_out[12..0] Saída do valor a ser carregado no contador (entrada do registrador PC). Essa saída está sempre ativa, não dependendo de habilitação. Quando não for ocorrer mudança no valor do contador, ou seja, quando nenhuma das entradas stack_pop, inc_pc, load_en ou wr_en estiver ativada, essa saída deverá corresponder ao valor atual do contador.

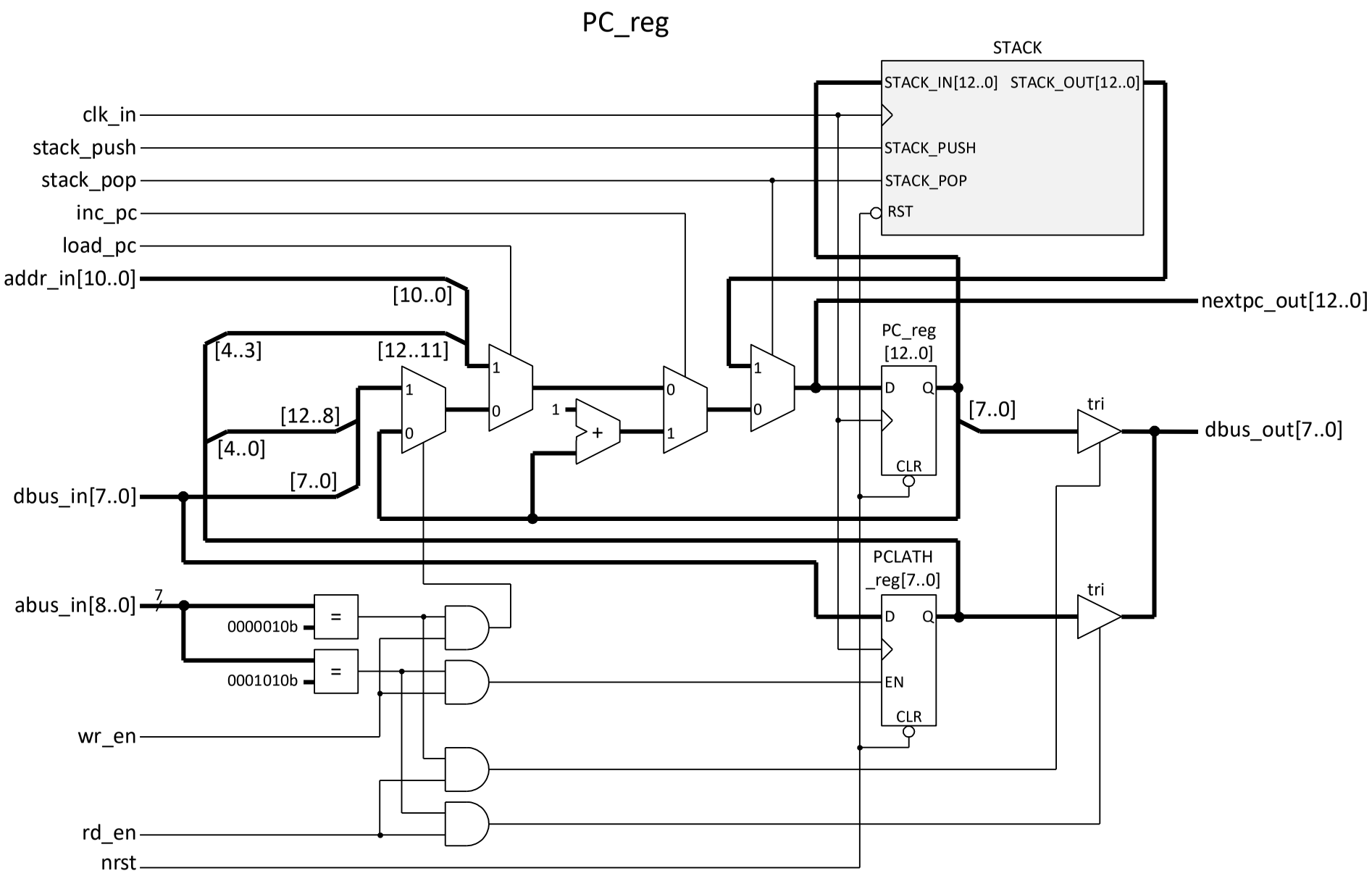
dbus_out[7..0] Saída de dados lidos com habilitação através de rd_en e endereçamento por abus_in. Quando não usada, deverá ficar em alta impedância (“ZZZZZZZZ”).

6.3. Diagrama

Diagrama na próxima página

Obs.: Embora não seja obrigatório, sugere-se organizar esse projeto em quatro PROCESS:

a) Lógica combinacional para nextpc
b) Lógica sequencial para PC_reg
c) Lógica sequencial para PCLATH
d) Lógica combinacional para dbus_out


PC_reg
STACK
STACK_IN[12..0]
STACK_OUT[12..0]
STACK_PUSH
STACK_POP
RST
clk_in
stack_push
stack_pop
inc_pc
load_pc
addr_in[10..0]
[10..0]
[4..3]
[12..11]
[12..8]
[4..0]
[7..0]
dbus_in[7..0]
nextpc_out[12..0]
PC_reg
[12..0]
D
Q
CLR
[7..0]
tri
dbus_out[7..0]
PCLATH
_reg[7..0]
EN
abus_in[8..0]
7
0000010b
0001010b
=
wr_en
rd_en
nrst
1
0
+