--- 2018 Multimaster RSRC VHDL Code

--- Current file name: rsrc.vhd

--- Last Revised: 10/1/2018; 1:40 p.m.

--- Author: WDR

--- Copyright: William D. Richard, Ph.D.

LIBRARY IEEE ;

USE IEEE.STD\_LOGIC\_1164.ALL ;

USE IEEE.STD\_LOGIC\_ARITH.ALL ;

ENTITY rsrc IS

PORT(clk : IN STD\_LOGIC ;

reset\_l : IN STD\_LOGIC ;

d : INOUT STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

address : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

read : OUT STD\_LOGIC ;

write : OUT STD\_LOGIC ;

request : OUT STD\_LOGIC ;

grant : IN STD\_LOGIC ;

done : IN STD\_LOGIC) ;

END rsrc ;

architecture structure of rsrc is

COMPONENT pc

PORT (bus\_in : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

clk : IN STD\_LOGIC ;

pc\_in : IN STD\_LOGIC ;

pc\_out : IN STD\_LOGIC ;

reset\_l : IN STD\_LOGIC ;

bus\_out : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0)) ;

END COMPONENT ;

COMPONENT a

PORT (bus\_in : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

a : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

clk : IN STD\_LOGIC ;

a\_in : IN STD\_LOGIC) ;

END COMPONENT ;

COMPONENT c

PORT (bus\_in : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

clk : IN STD\_LOGIC ;

c\_in : IN STD\_LOGIC ;

c\_out : IN STD\_LOGIC ;

bus\_out : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0)) ;

END COMPONENT ;

COMPONENT alu

PORT (a : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

b : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

add : IN STD\_LOGIC ;

sub : IN STD\_LOGIC ;

andx : IN STD\_LOGIC ;

c\_eq\_b : IN STD\_LOGIC ;

inc4 : IN STD\_LOGIC ;

neg : IN STD\_LOGIC ;

shr : IN STD\_LOGIC ;

shl : IN STD\_LOGIC ;

shc : IN STD\_LOGIC ;

shra : IN STD\_LOGIC ;

orx : IN STD\_LOGIC ;

notx : IN STD\_LOGIC ;

n : IN STD\_LOGIC\_VECTOR(4 DOWNTO 0) ;

c : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0)) ;

END COMPONENT ;

COMPONENT shiftcounter

PORT (rrc : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

ir : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

n : OUT STD\_LOGIC\_VECTOR(4 DOWNTO 0)) ;

END COMPONENT ;

COMPONENT regfile

PORT (bus\_in : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

ir : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

r\_in : IN STD\_LOGIC ;

r\_out : IN STD\_LOGIC ;

ba\_out : IN STD\_LOGIC ;

gra : IN STD\_LOGIC ;

grb : IN STD\_LOGIC ;

grc : IN STD\_LOGIC ;

clk : IN STD\_LOGIC ;

rrc : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

bus\_out : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0)) ;

END COMPONENT ;

COMPONENT ma

PORT (bus\_in : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

address : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

clk : IN STD\_LOGIC ;

grant : IN STD\_LOGIC ;

ma\_in : IN STD\_LOGIC) ;

END COMPONENT ;

COMPONENT md

PORT (bus\_in : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

clk : IN STD\_LOGIC ;

md\_rd : IN STD\_LOGIC ;

md\_wr : IN STD\_LOGIC ;

md\_out : IN STD\_LOGIC ;

md\_bus : IN STD\_LOGIC ;

d : INOUT STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

bus\_out : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0)) ;

END COMPONENT ;

COMPONENT ir

PORT (bus\_in : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

clk : IN STD\_LOGIC ;

c1\_out : IN STD\_LOGIC ;

c2\_out : IN STD\_LOGIC ;

ir\_in : IN STD\_LOGIC ;

bus\_out : OUT STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

ir : INOUT STD\_LOGIC\_VECTOR(31 DOWNTO 0)) ;

END COMPONENT ;

COMPONENT conbit

PORT (bus\_in : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

ir : IN STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

con : OUT STD\_LOGIC) ;

END COMPONENT ;

COMPONENT control

PORT (clk : IN STD\_LOGIC ;

opcode : IN STD\_LOGIC\_VECTOR(4 DOWNTO 0) ;

con : IN STD\_LOGIC ;

done : IN STD\_LOGIC ;

reset\_l : IN STD\_LOGIC ;

grant : IN STD\_LOGIC ;

request : OUT STD\_LOGIC ;

a\_in : OUT STD\_LOGIC ;

c\_in : OUT STD\_LOGIC ;

c\_out : OUT STD\_LOGIC ;

pc\_in : OUT STD\_LOGIC ;

pc\_out : OUT STD\_LOGIC ;

c1\_out : OUT STD\_LOGIC ;

c2\_out : OUT STD\_LOGIC ;

ir\_in : OUT STD\_LOGIC ;

gra : OUT STD\_LOGIC ;

grb : OUT STD\_LOGIC ;

grc : OUT STD\_LOGIC ;

r\_in : OUT STD\_LOGIC ;

r\_out : OUT STD\_LOGIC ;

ba\_out : OUT STD\_LOGIC ;

md\_bus : OUT STD\_LOGIC ;

md\_rd : OUT STD\_LOGIC ;

md\_wr : OUT STD\_LOGIC ;

md\_out : OUT STD\_LOGIC ;

ma\_in : OUT STD\_LOGIC ;

read : OUT STD\_LOGIC ;

write : OUT STD\_LOGIC ;

add : OUT STD\_LOGIC ;

sub : OUT STD\_LOGIC ;

andx : OUT STD\_LOGIC ;

orx : OUT STD\_LOGIC ;

notx : OUT STD\_LOGIC ;

neg : OUT STD\_LOGIC ;

c\_eq\_b : OUT STD\_LOGIC ;

inc4 : OUT STD\_LOGIC ;

shr : OUT STD\_LOGIC ;

shra : OUT STD\_LOGIC ;

shl : OUT STD\_LOGIC ;

shc : OUT STD\_LOGIC) ;

END COMPONENT ;

SIGNAL con : STD\_LOGIC ;

SIGNAL a\_in : STD\_LOGIC ;

SIGNAL c\_in : STD\_LOGIC ;

SIGNAL c\_out : STD\_LOGIC ;

SIGNAL pc\_in : STD\_LOGIC ;

SIGNAL pc\_out : STD\_LOGIC ;

SIGNAL c1\_out : STD\_LOGIC ;

SIGNAL c2\_out : STD\_LOGIC ;

SIGNAL ir\_in : STD\_LOGIC ;

SIGNAL gra : STD\_LOGIC ;

SIGNAL grb : STD\_LOGIC ;

SIGNAL grc : STD\_LOGIC ;

SIGNAL r\_in : STD\_LOGIC ;

SIGNAL r\_out : STD\_LOGIC ;

SIGNAL ba\_out : STD\_LOGIC ;

SIGNAL md\_bus : STD\_LOGIC ;

SIGNAL md\_rd : STD\_LOGIC ;

SIGNAL md\_wr : STD\_LOGIC ;

SIGNAL md\_out : STD\_LOGIC ;

SIGNAL ma\_in : STD\_LOGIC ;

SIGNAL add : STD\_LOGIC ;

SIGNAL sub : STD\_LOGIC ;

SIGNAL andx : STD\_LOGIC ;

SIGNAL orx : STD\_LOGIC ;

SIGNAL notx : STD\_LOGIC ;

SIGNAL neg : STD\_LOGIC ;

SIGNAL c\_eq\_b : STD\_LOGIC ;

SIGNAL inc4 : STD\_LOGIC ;

SIGNAL shr : STD\_LOGIC ;

SIGNAL shra : STD\_LOGIC ;

SIGNAL shl : STD\_LOGIC ;

SIGNAL shc : STD\_LOGIC ;

SIGNAL n : STD\_LOGIC\_VECTOR(4 DOWNTO 0) ;

SIGNAL cpu\_bus : STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

SIGNAL a\_bus : STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

SIGNAL c\_bus : STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

SIGNAL ir\_bus : STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

SIGNAL rrc\_bus : STD\_LOGIC\_VECTOR(31 DOWNTO 0) ;

BEGIN

rsrcpc:pc

PORT MAP(bus\_in => cpu\_bus ,

clk => clk ,

pc\_in => pc\_in ,

pc\_out => pc\_out ,

reset\_l => reset\_l ,

bus\_out => cpu\_bus) ;

rsrcareg:a

PORT MAP(bus\_in => cpu\_bus ,

a => a\_bus ,

clk => clk ,

a\_in => a\_in) ;

rsrccreg:c

PORT MAP(bus\_in => c\_bus ,

clk => clk ,

c\_in => c\_in ,

c\_out => c\_out ,

bus\_out => cpu\_bus) ;

rsrcalu:alu

PORT MAP(a => a\_bus ,

b => cpu\_bus ,

add => add ,

sub => sub ,

andx => andx ,

c\_eq\_b => c\_eq\_b ,

inc4 => inc4 ,

neg => neg ,

shr => shr ,

shl => shl ,

shc => shc ,

shra => shra ,

orx => orx ,

notx => notx ,

n => n ,

c => c\_bus) ;

rsrcshiftcounter:shiftcounter

PORT MAP(rrc => rrc\_bus ,

ir => ir\_bus ,

n => n) ;

rsrcregfile:regfile

PORT MAP(bus\_in => cpu\_bus ,

ir => ir\_bus ,

r\_in => r\_in ,

r\_out => r\_out ,

ba\_out => ba\_out ,

gra => gra ,

grb => grb ,

grc => grc ,

clk => clk ,

rrc => rrc\_bus ,

bus\_out => cpu\_bus) ;

rsrcmareg:ma

PORT MAP(bus\_in => cpu\_bus ,

address => address ,

clk => clk ,

grant => grant ,

ma\_in => ma\_in) ;

rsrcmdreg:md

PORT MAP(bus\_in => cpu\_bus ,

clk => clk ,

md\_rd => md\_rd ,

md\_wr => md\_wr ,

md\_out => md\_out ,

md\_bus => md\_bus ,

d => d ,

bus\_out => cpu\_bus) ;

rsrcirreg:ir

PORT MAP(bus\_in => cpu\_bus ,

clk => clk ,

c1\_out => c1\_out ,

c2\_out => c2\_out ,

ir\_in => ir\_in ,

bus\_out => cpu\_bus ,

ir => ir\_bus) ;

rsrcconbit:conbit

PORT MAP(bus\_in => rrc\_bus ,

ir => ir\_bus ,

con => con) ;

rsrccontrol:control

PORT MAP(clk => clk ,

opcode => ir\_bus(31 DOWNTO 27) ,

con => con ,

done => done ,

reset\_l => reset\_l ,

grant => grant ,

request => request ,

a\_in => a\_in ,

c\_in => c\_in ,

c\_out => c\_out ,

pc\_in => pc\_in ,

pc\_out => pc\_out ,

c1\_out => c1\_out ,

c2\_out => c2\_out ,

ir\_in => ir\_in ,

gra => gra ,

grb => grb ,

grc => grc ,

r\_in => r\_in ,

r\_out => r\_out ,

ba\_out => ba\_out ,

md\_bus => md\_bus ,

md\_rd => md\_rd ,

md\_wr => md\_wr ,

md\_out => md\_out ,

ma\_in => ma\_in ,

read => read ,

write => write ,

add => add ,

sub => sub ,

andx => andx ,

orx => orx ,

notx => notx ,

neg => neg ,

c\_eq\_b => c\_eq\_b ,

inc4 => inc4 ,

shr => shr ,

shra => shra ,

shl => shl ,

shc => shc) ;

END structure;