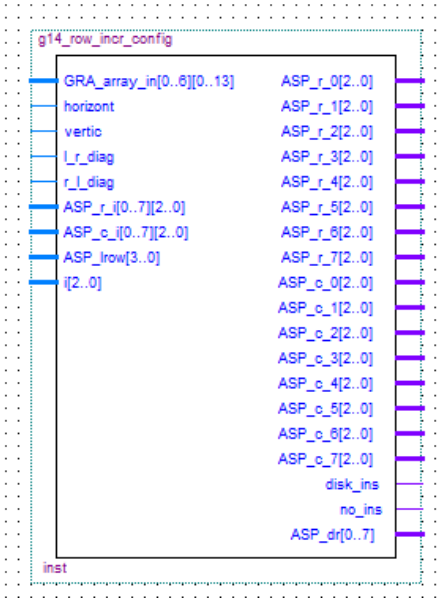
Group 14 -
Jerome Colomb 260349913
Matthew Johnston 260349319

Lab 3 Report

g14_row_increase_config.vhd

Top-level I/O



Inputs

GRA_array_in – gra_array

Horizt, vertic, l_r_diag, r_l_diag – bit

ASP_r_i, ASP_c_i – asp_reg

ASP_lrow – 4-bit std_logic_vector

i – 3 bit std_logic_vector

Outputs

ASP_r_0 through ASP_r_7 – 2-bit std_logic_vector

ASP_c_0 through ASP_c_7 – 2-bit std_logic_vector

Disk_ins, no_ins – bit

ASP_dr – 8-bit std_logic_vector

NB: the gra_array data type is a 7 by 14 array of std_logic_vector and the ASP_reg data type is an 8 by 3 array of std_logic_vector

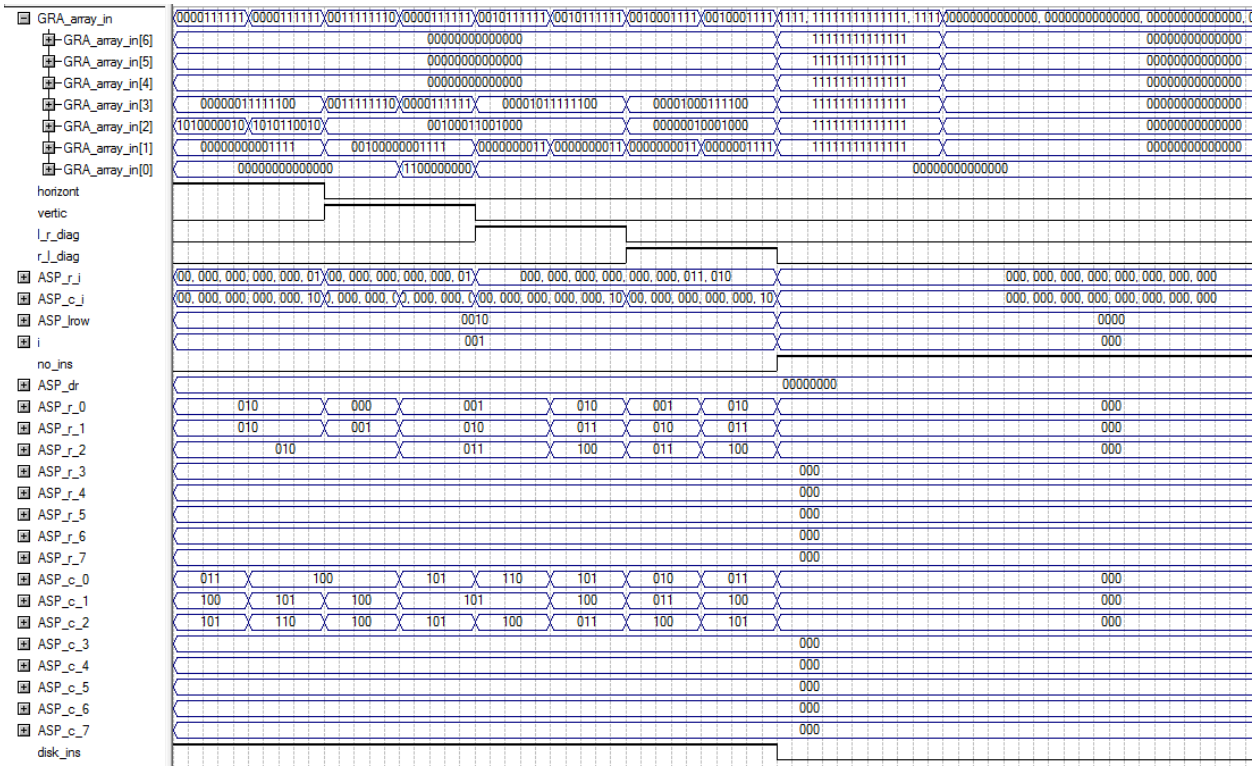
Description of Circuit

This circuit takes in a GRA array representation the state of the playing field prior to a piece being placed, a signal representing the directional configuration of the ASP's longest row, the length of that row, the registers in which the row is stored, and a register index. Based on the directional configuration of the longest row, the circuit determines which end of the row the next piece should be placed by a predetermined formula. If the longest row cannot be extended in either direction, the "no_ins" bit is activated. The output of the circuit is the modified ASP registers containing the location of the new piece placed to extend the longest row. The appropriate ASP_dr bit is activated when there is not space to place a piece.

Group 14 -
Jerome Colomb 260349913
Matthew Johnston 260349319

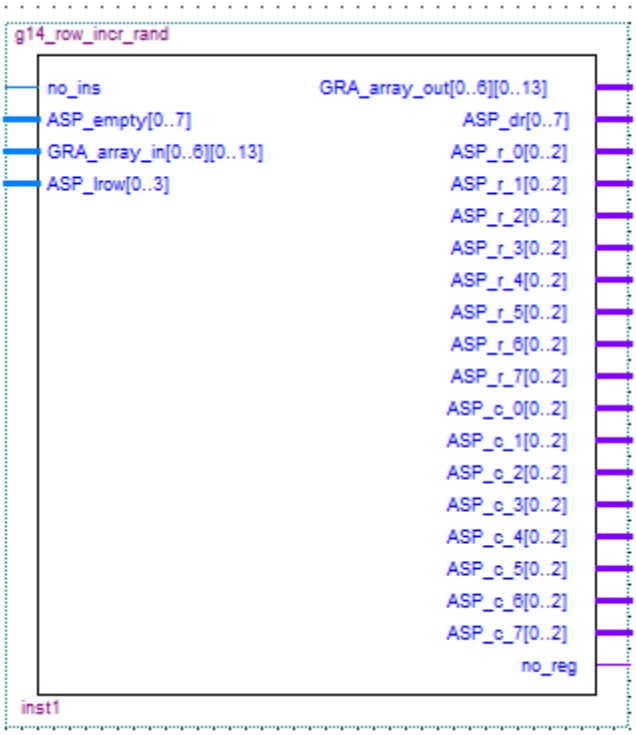
Testing

The circuit was tested by creating sample situations by implementing the GRA_array and ASP_r_i and ASP_c_i configurations described in the Lab 3 Instruction Guide. 2 cases were considered (one for each end of the longest row) for each directional configuration. The outputs were verified by checking the output registers to ensure that the appropriate cells were added to the longest row. Finally, The empty register case was tested to prove that no_ins would be high and disk_ins would be low.



Group 14 -
Jerome Colomb 260349913
Matthew Johnston 260349319

g14_row_incr_rand.vhd



Top-level I/O

Inputs

No_ins – bit

ASP_empty – 8-bit std_logic_vector

ASP_lrow – 4-bit std_logic_vector

GRA_array_in – gra_array

Outputs

GRA_array_out – gra_array

ASP_dr – 8-bit std_logic_vector

ASP_r_0 through ASP_r_7 – 3-bit std_logic_vector

ASP_c_0 thorugh ASP_c_7 – 3-bit std_logic_vector

No_reg – bit

NB: the gra_array data type is a 7 by 14 array of std_logic_vector

Description of Circuit

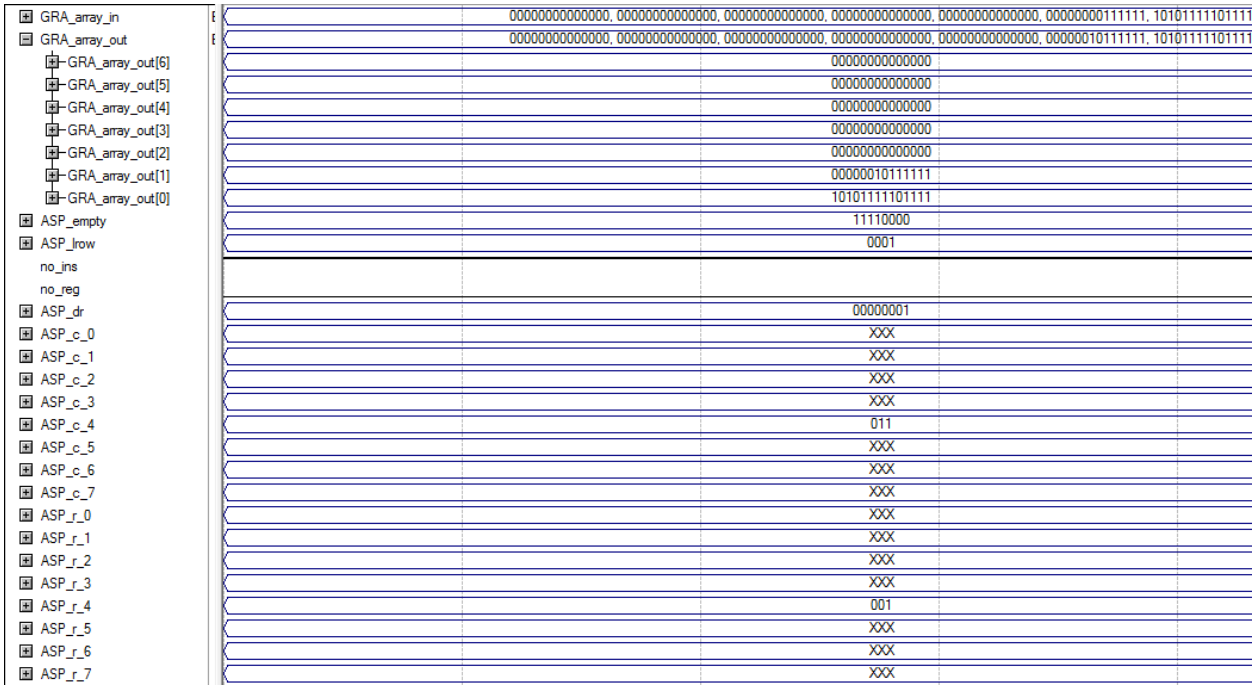
This circuit takes in a GRA array representing the state of the playing field prior to a piece being placed, ASP_empty which indicates the status of empty register spaces, ASP_lrow which indicates the length of the current longest row, and no_ins which means that no disk has yet been inserted. The circuit is used when the longest row cannot be extended due to space limitations on the GRA array (external of this circuit). Thus, the piece is placed in the next available position by "snaking" through the GRA array. Once the next available spot is found, the piece is placed there, and the ASP registers and GRA array are updated to reflect the change.

Testing

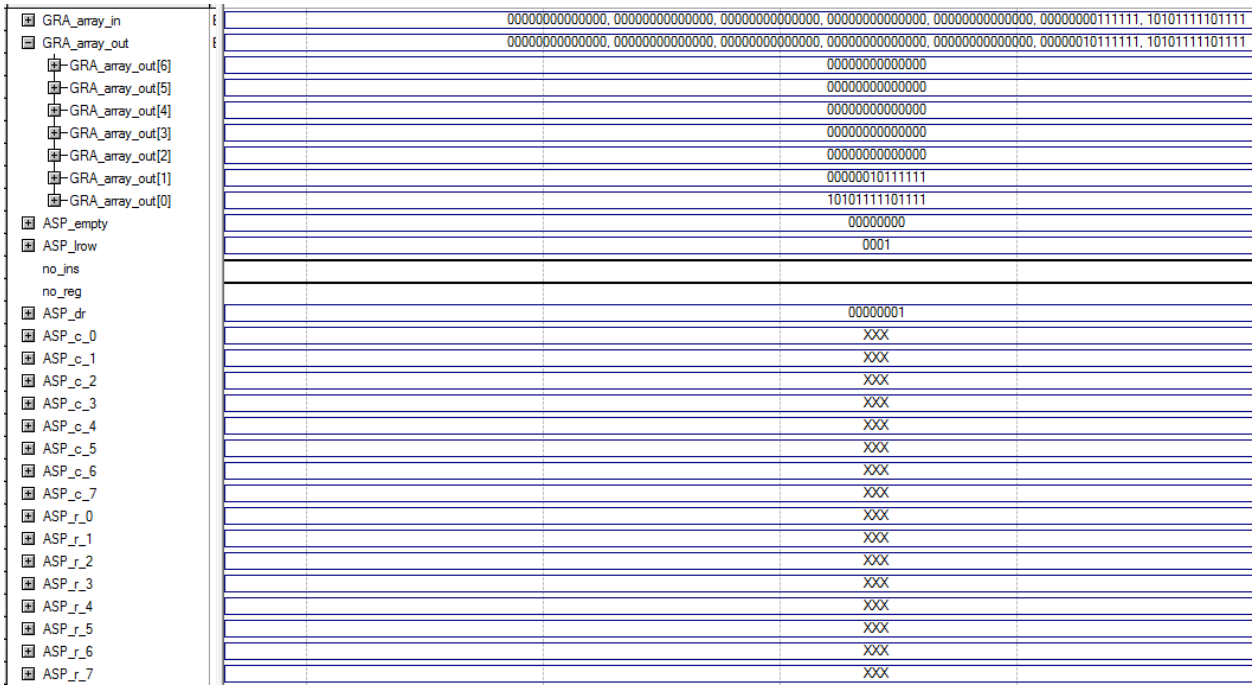
The circuit was tested by creating a sample GRA array and register configurations and observing the next available cell chosen along the the postion in the register where the cell's address was

saved. Output verification requires the inspection of GRA_array_out to show the correct location and of the output registers to show correct address and saving those values in the proper register location.

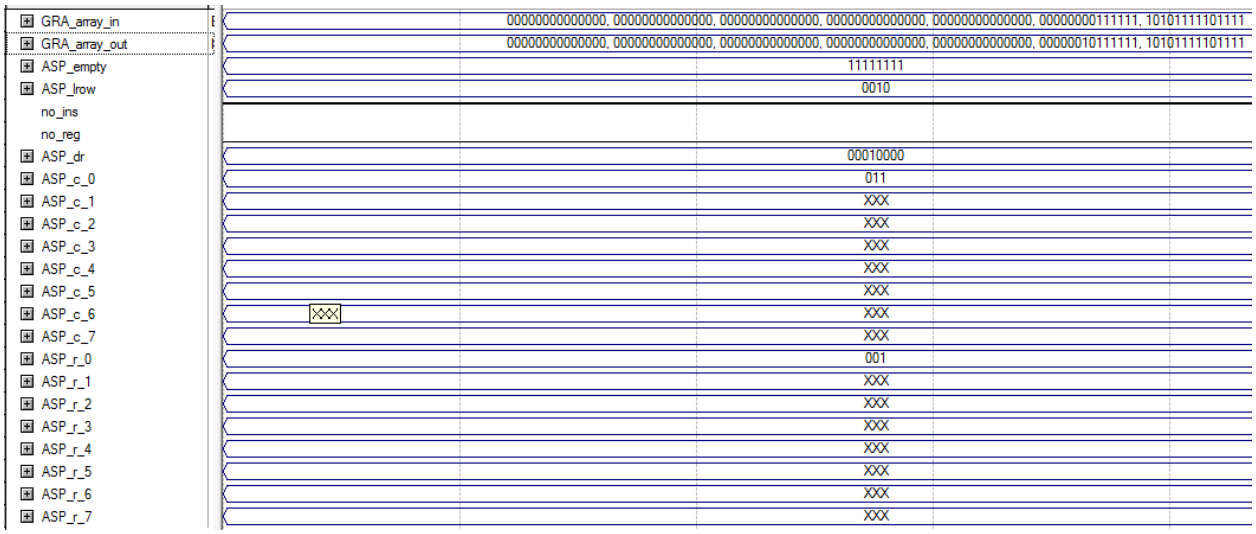
Save to register 4



All registers are full



Group 14 -
Jerome Colomb 260349913
Matthew Johnston 260349319



next piece in row 2