Draw Block Documentation

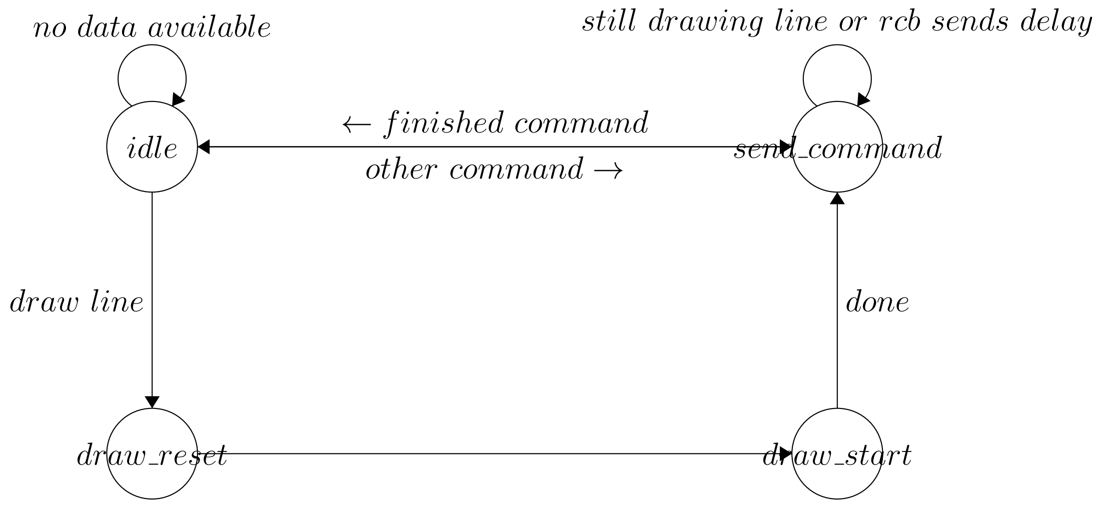
Jake Humphrey

March 10, 2014

1 Overview

The Draw Block recieves draw commands from the host processor and re-encodes them for the RAM control block (RCB). In the case of a draw line command, it first uses the `draw_any_octant` entity to convert the line into a stream of pixels, before sending individual draw commands to the RCB for each pixel.

2 Finite State Machine



The Finite State Machine schedules the operation of the Draw Block. When there are no commands, the block is in the `idle` state. When a command comes in, the block goes to one of two states. The simplest is a `move_pen` or a `clear_screen` command. In this case, the block goes to the `send_command` state to re-encode and forward the command to the RAM Control Block.

In the case of a `draw_line` command, the Draw Block must work out the stream of pixels to be drawn. For this it uses the `draw_any_octant` sub-entity. first it sets the start point in the `draw_reset` state, and the endpoint in the `draw_start` state. Then it moves to the `send_command` state, where it remains, outputting pixel values until the line is complete.

In either case, when the command is over, the Draw Block returns to the `idle` state to await another command.

3 Other processes

3.1 read_new_command

Type Clocked

Inputs `hdb, dav, command`

Function When a command is available from the host processor, and the draw block is ready to process the command, this process reads the command into a register to be available throughout the execution of the command. The previous command in this register is shifted to another register, such that the current and previous commands are always available.

Outputs `command, prev_command.`

3.2 set_dao_inputs

Type Combinational

Inputs `command, prev_command.`

Function This process takes the x and y values of the current and previous commands (the previous command giving the initial pen position), and works out dx and dy values for the current command. This process also works out the correct octant to be drawn based on the values of dx and dy.

Outputs `xin, yin, negx, negy, swapxy, xbias.`

3.3 send_rcb_inputs

Type Combinational

Inputs `state, command, prev_command, dao_xout, dao_yout`

Function This process drives the communications to the RAM control block. If a line is being drawn, the output x and y values are added to the current pen position to give the canvas co-ordinates forwarded to the RCB. For other commands, the x and y values are simply forwarded as-is. This process also re-encodes the operation and pen signals into a single command signal for the RCB.

Outputs `dbb_bus`

3.4 finished

Type Combinational

Inputs `state, dav`

Function This process indicates when the draw block has finished working and has no outstanding commands. It then asserts the `db_finish` line.

Outputs `db_finish`

Table 1: FSM state table

State	hdb_busy	dao_draw	dao_reset
idle	0	0	0
draw_reset	1	0	1
draw_start	1	1	0
send_command	1	0	0