**Game of Life**

# Overview

This core plays Conway’s Game of Life and provides a two color output of the game’s progress. The core offers control over the speed of the calculations and the size and positioning of the display window.

## Arena

### Size

The default arena for the life game is 256x256 cells. This size may be increased to 512x256 by setting the value of the HCELLS and VCELLS configuration defines. A larger size will increase the number of resources required. For simplicity the arena is torus shaped. The left and right edges are connected as are the top and bottom edges.

### Loading

The arena must be loaded with data before the game can start. Data for one row of the arena is loaded into holding registers (registers 00h to 3Ch). Next the row to update is specified in register 40h. Then the row load bit is pulsed to load the row of data into the arena. Sample initialization software is in the appendix.

## Display

The Game of Life’s progress is displayed in a 256x256 (or 512x256) bitmapped display window.

The position of the display window is controllable relative to external horizontal and vertical sync inputs. The display may be disabled by clearing a bit in the control register.

Note that the display may be used as a simple bitmapped display by disabling the life calculation.

## Calculation

Calculation for the game of life is Conway’s standard calculation. The calculation for a given cell is governed by the state of the surrounding cells.

The game calculates an entire row of cells in parallel every clock cycle. A row counter continuously increments and selects a row for calculation. Three rows (above, current, and below) are simultaneously read from the arena memory. Write-back of the calculated values occurs four cycles after a row has been visited by the calculator so that calculations for subsequent rows aren’t affected by a previous row.

The frequency of the calculation may be controlled by setting the frequency register. This register can range from 0001h to FFFFh corresponding to the lowest possible frequency and highest possible frequency respectively. Frequency control is via a harmonic frequency synthesizer.

# Game Hardware

The game uses block ram resources to store the game data. The block ram is organized as a five port memory; one write port and four read ports. Three read ports are for the calculation and the fourth read port is for the display. Calculation and display occur in parallel and at different clock rates.

## Bus Interface

The core is interfaced to the rest of the system via a 32 bit WISHBONE compatible bus.

## Video Interface

The core is interfaced to the video pipeline via 24 bit RGB input and output buses. External horizontal and vertical sync signals must be provided for timing reference. An externally supplied video clock controls the size of the displayed pixels.

## I/O Port Description

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Port | WB |  | I/O |  |  |
| rst\_i | \* | 1 | I | resets the core default values of registers are set |  |
| clk\_i | \* | 1 | I | core clock input. This clock is shared between the bus interface and for clocking the calculations for the game. |  |
| cyc\_i | \* | 1 | I | bus cycle valid |  |
| stb\_i | \* | 1 | I | data strobe |  |
| ack\_o | \* | 1 | O | transfer acknowledge |  |
| we\_i | \* | 1 | I | write enable |  |
| adr\_i | \* | 32 | I | address input |  |
| dat\_i | \* | 32 | I | data input |  |
| dat\_o | \* | 32 | O | data output – currently data output is not supported. This output is always zero. |  |
| vclk |  | 1 | I | video clock input |  |
| vsync |  | 1 | I | vertical sync reference input |  |
| hsync |  | 1 | I | horizontal sync reference input |  |
| rgb\_i |  | 24 | I | external RGB input. This input is displayed outside the arena or when the arena background is transparent. |  |
| rgb\_o |  | 24 | O | Game display output. |  |

# Register Set Desciption

All registers are write only. There is no read-back provided. Registers are 32 bit aligned in address increments of four.

|  |  |  |
| --- | --- | --- |
| Register | Default Value | Description |
| 00h to 3Ch | --- | Each one of these registers holds part of the pattern to be written to a row of the arena. The first register contains bits 0 to 31 for the row. The next register contains bits 32 to 63 for the row and so on for 16 registers. |
| 40h | --- | The low order eight bits indicate which row of the arena to load with the data in the holding register. This register and the holding registers should be set before setting bit zero of the control register (reg. 44h). |
| 44h | 06h | control register   |  |  | | --- | --- | | Bit |  | | 0 | Set this bit to one to load the row setup by the above registers into the life arena. This bit resets to zero automatically. | | 1 | Calculation enable. If set to zero the life game will cease calculations and remain in a steady state. Useful for loading the arena. | | 2 | Display enable. If set to zero the display will be disabled. Life will continue to run. | |
| 48h | 10h | Bits 0 to 15. Frequency control. Setting this register to a higher value will increase the frequency at which the life game is calculated. The high order 16 bits of this register should be set to zero for proper operation. |
| 4Ch | FFFFFFh | Bits 0 to 23 are a 24 bit RGB color value for the display of “on” pixels in the life game. |
| 50h | 000000h | |  |  | | --- | --- | | Bits |  | | 0 to 23 | 24 bit RGB value for the color of the background for the game. | | 31 | If set to one this makes the game’s background transparent and it will overlay an external RGB input. | |
| 54h | 8F8080E0h | |  |  | | --- | --- | | Bits |  | | 0 to 11 | Position of display window relative to horizontal sync input | | 15 | Scale indicator. 1=2 clocks per pixel, 0=1 clock per pixel | | 16 to 25 | Position of display window relative to vertical sync input. | | 31 | Scale indicator. 1=2 scanlines per pixel, 0 = 1 scanline per pixel. | |

# Appendix

## Sample Initialization Software (DSD9 Assembler Code)

|  |
| --- |
| init\_lifegame:    FFFCDA90 3000014009 ldi r5,#$FFD30000 ; control register set  FFFCDA95 000000FFCD  ; first clear out the environment  FFFCDA9A 0000008030 mov r2,r0  FFFCDAA0 000000C030 mov r3,r0  .life1:  FFFCDAA5 00080085B4 stt r0,[r5+r2\*4]  FFFCDAAA 0000108204 add r2,r2,#1  FFFCDAB0 FFF504025C bltu r2,#16,.life1  .life2:  FFFCDAB5 000400C594 stt r3,64[r5] ; set the row to load  FFFCDABA 0000110009 ldi r4,#1  FFFCDAC0 0004410594 stt r4,68[r5] ; pulse load bit  FFFCDAC5 000010C304 add r3,r3,#1  FFFCDACA FFEB00035C bltu r3,#256,.life2 ; 256 rows  FFFCDAD0 00000000C1  ; now add some random data  .life3:  FFFCDAD5 F308003F50 call gen\_rand[pc]  FFFCDADA 000000C130 mov r3,r1,#$FF  FFFCDAE0 F307503F50 call gen\_rand[pc]  FFFCDAE5 0000008130 mov r2,r1  FFFCDAEA F306B03F50 call gen\_rand[pc]  FFFCDAF0 0000F04108 and r1,r1,#$F  FFFCDAF5 000400C594 stt r3,64[r5] ; set the row (random)  ; zero out all the row  FFFCDAFA 0000018030 mov r6,r0  .life4:  FFFCDB00 00080185B4 stt r0,[r5+r6\*4]  FFFCDB05 0000118604 add r6,r6,#1  FFFCDB0A FFF604065C bltu r6,#16,.life4  FFFCDB10 00082045B4 stt r2,[r5+r1\*4] ; set data word within row (random)  FFFCDB15 0000108009 ldi r2,#1  FFFCDB1A 0004408594 stt r2,68[r5] ; pulse row load bit  FFFCDB20 0000110404 add r4,r4,#1  FFFCDB25 FFB005045C bltu r4,#20,.life3 ; 20 random samples  FFFCDB2A 0000608009 ldi r2,#6  FFFCDB30 0004408594 stt r2,68[r5] ; turn on display and calculation  FFFCDB35 0000000AEF ret |

## WISHBONE Compatibility Datasheet

The LifeGame core may be directly interfaced to a WISHBONE compatible bus.

|  |  |  |
| --- | --- | --- |
| WISHBONE Datasheet  WISHBONE SoC Architecture Specification, Revision B.3 | | |
|  |  | |
| Description: | Specifications: | |
| General Description: | Conway’s Game of Life | |
| Supported Cycles: | MASTER, READ / WRITE  MASTER, BLOCK READ / WRITE | |
| Data port, size:  Data port, granularity:  Data port, maximum operand size:  Data transfer ordering:  Data transfer sequencing | 32 bit  32 bit  32 bit  Little Endian  any (undefined) | |
| Clock frequency constraints: |  | |
| Supported signal list and cross reference to equivalent WISHBONE signals | Signal Name:  ack\_o  adr\_i(31:0)  clk\_i  dat\_i(31:0)  dat\_o(31:0)  cyc\_i  stb\_i  wr\_i | WISHBONE Equiv.  ACK\_O  ADR\_I()  CLK\_I  DAT\_I()  DAT\_O()  CYC\_I  STB\_I  WE\_I |
| Special Requirements: |  | |