DESIGN SPECIFICATION REPORT
A4

November 24, 2017

Jippe van Dunné
4573137

Lars Gelling
4590600

Daan de Groot
4607414

Nick Hogendoorn

Pim Jansen
4637165

Koen Lam
4608844

Emiel van der Meijs
4567528

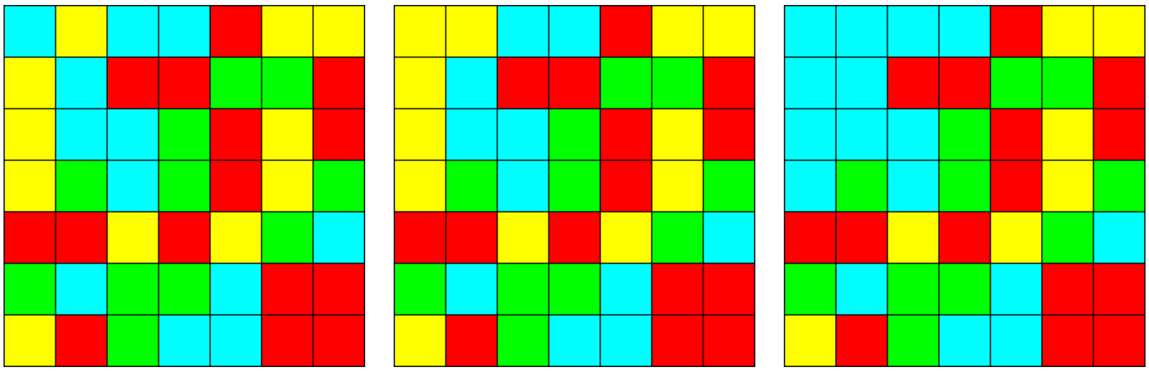
Mirza Mrahorović
4596536

Contents

1 INTRODUCTION

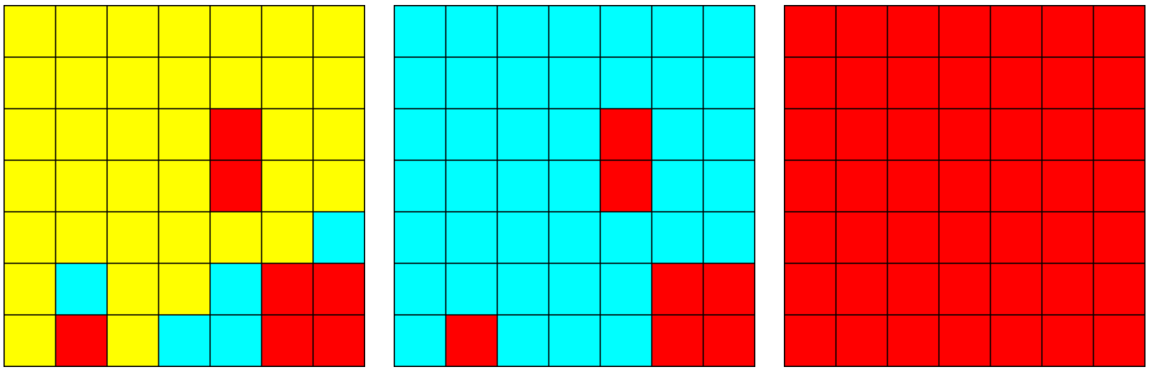
This report contains the the major outlines of the EPO 3 project by group A4. In the following weeks we will design and implement an integrated circuit based on the layout specified here. It will therefore focus on defining and describing the operations required by main components making up the system.

The design that is going to be implemented is the game Flood. The game consist of a square matrix. Each element of this matrix consist of a random color. The player can input a color and the most left element of the matrix will turn into that color which can be seen in figure 1a. Then when the player chooses another color the most left element and all the neighbouring elements with the same color will turn into the new color as in figure 1b. The goal of this game is to get the whole matrix the same color in the least amount of turns. The process of the beginning and end of a Flood game can be found in figure 1 and 2.



(a) Map after generation **(b)** Map after an input of yellow **(c)** Map after an input of blue

Figure 1: Flood game



(a) Map almost finished **(b)** Map after an input of blue **(c)** Map after an input of red

Figure 2: Flood game after a couple of moves

2 Game requirements and chip layout

For this design there are both functional requirements and boundary conditions. The block diagram of the system can be found in 3.

2.1 Functional requirements

2.1.1 Main function

The product should be able to display the game Flood it on a monitor and a player should be able to play the game as well.

2.1.2 Side function

The player has to have the ability to replay a map.

2.2 Boundary conditions

- The design has use a maximum of 40, 000 transistor pairs.
- The design should use no more than 200-250 flip-flops
- It has to work on a 6.144 MHz clock

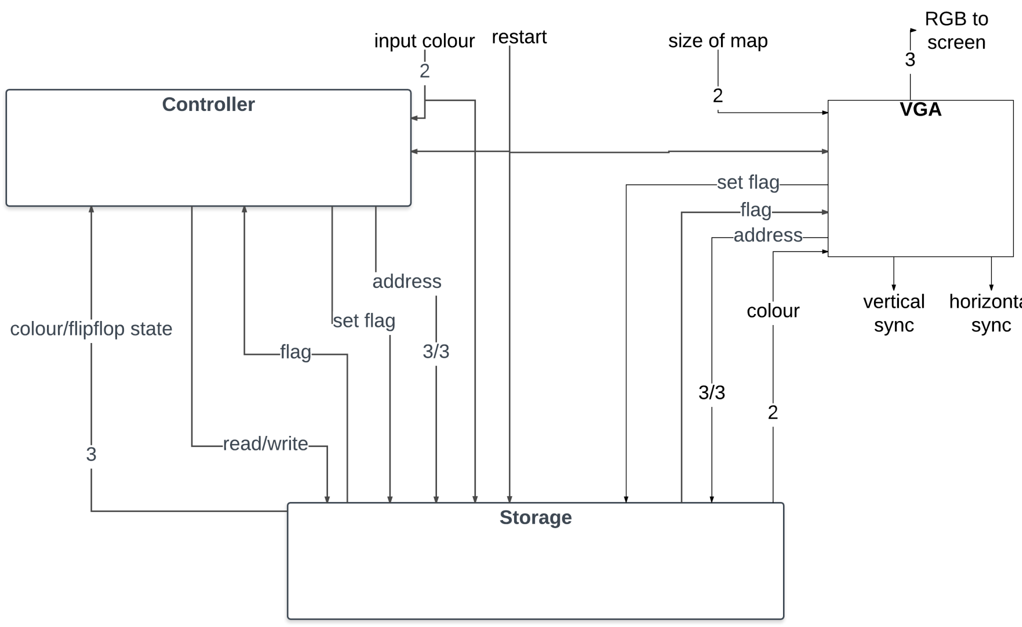


Figure 3: General overview of the chip layout

3 Storage

The storage module is designed to store information about the the map and its current state. It can be divided into three different parts, a storage controller (not to be confused with the main controller that will be discussed in Section 4), an array of flip-flops and a random number generator which will be discussed in Section 6.

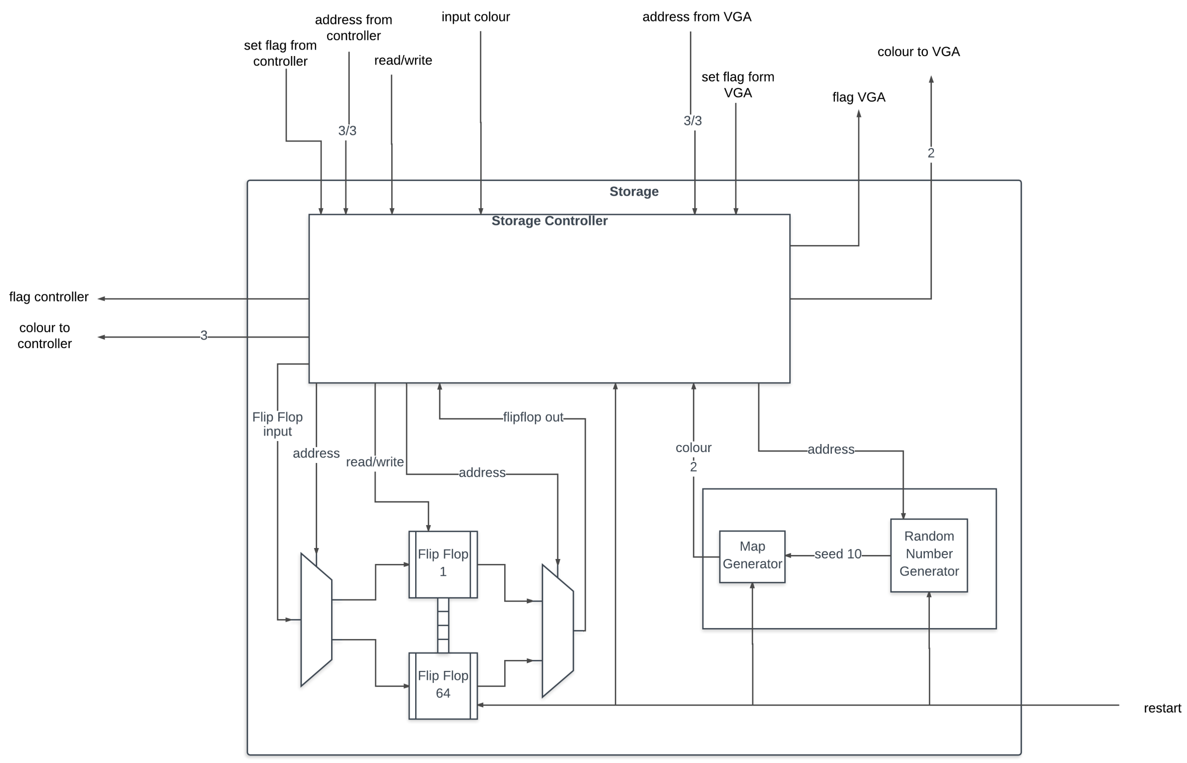


Figure 4: Block diagram of the storage component

3.1 Storage components

3.1.1 Storage controller

As can be seen in Figure 4, all in- and outputs of the storage component are connected to the storage controller, while the RNG and the flip-flop array are only connected internally. This is because all requests from the VGA or the controller should first be interpreted by the storage controller, before any output can be given to the respective component.

3.1.2 Flip-flop array

For each of the 64 parts of the map, one flip-flop is reserved. However, each part can get four different colours assigned, which would logically need two bits to store the information. This would require twice as many flip-flops, for which there is no space. So instead of saving the colour of a part, its ’originality’ is saved. These two states of originality are a ’0’, which means that since the start of the game, the colour of the part has not been changed, and a ’1’, which means that the colour has been changed. This is because the game functions in such a way that, if the colour of a part has ever been changed, it will change at every turn to the colour the player chooses.

3.2 Operations

The storage controller can get three kinds of requests or ’operations’, the controller can ask it to reset, to read from a flip-flop, or to write to a flip-flop, but the VGA can only ask it to read from a flip-flop.

3.2.1 Read

When a read request is received, a multiplexer selects the output from the required flip-flop which tells whether the part still has its original colour that was assigned to it by the RNG, or that it has changed in the past. This output, combined with a multiplexer selects either the colour the player has selected if the flip-flop gives a ’1’, or the original colour of the part is calculated by the RNG and that output is selected by the multiplexer if the flip-flop gives a ’0’.

3.2.2 Write

If a write request is received, a demultiplexer is used to send a ’1’ to the requested flip-flop. This is always a ’1’, because the only case in which the controller wants to write to a flip-flop during the game is when a part of the map is absorbed by the changing colour, and thus the originality changes from ’0’ to ’1’.

3.2.3 Restart

In case of a restart request, all the flip-flops in the array are set to ’0’ directly, without interaction with the storage controller Additionally, it causes the RNG to generate a new seed for the creation of a new map.

3.3 Description of inputs and outputs

flag controller, set flag from controller & address form controller If the controller has set an address on the input address from controller, it is possible that the storage controller is not able to directly answer the request. For this reason a set flag from the controller and "flag" to controller is set to one. When the storage controller did answer the request, the flag to the controller is set to zero.

Input colour The input colour is the colour that the user selected.

read/write The controller has the opportunity to write or to read to a specified address and so to the flipflops. An one will result in writing to the flipflops and a zero will result in reading from the flipflops.

colour to controller If the controller requests the information about a specified address, the colour and the state of the flipflop is send back. The state of the flipflop is determined by the fact of the address corresponding with this flipflop ever changed from the original colour to the input colour.

flag VGA, set flag from VGA & address form VGA Also if the VGA has set an address on the input address of the VGA, it is possible that the storage controller is not able to directly answer the request. Again the flag mechanism is used as a check for the VGA component to recognise if the storage controller has answered the request of the VGA.

Colour to VGA The output to the VGA is a colour of the address that is specified at the address from the VGA input.

Restart There can be a restart signal of the game, this should set all the flipflops to zero and the random number generator should generate a new seed. This will be explained in section 6

4 CONTROLLER

The controller is a module of the design which is engaged with updating the map. This means that in the controller the inputs, which can be 1 of the 4 available colours or the reset button, are read by the input buffer. The controller will start a algorithm which will work in the following way, there is a 7 by 7 matrix with 2 vectors, a x-vector and a y-vector. In the controller there is a separate part which is accountable which keeps track for which 'coordinate-block' (x,y) has to be calculated if it should change or stay the same, this block is called the controller. The flag-fall section will be responsible for keeping track when the counter has to go to the next coordinate, this is a separate block because it is not possible to use the command $fallingedge$ on a 'regular' data line, this can only be used with the clock. The data of each block saying which colour it is and if the block has changed colour already is stored in the storage section. The controller and storage section will thus communicate trough data lines. There will be a set-flag line towards the storage from the controller which will be high when there is data available for the storage, this can be a address from which the colour data is asked from (this is when the read/write line is low) or when the read/write line is high this means that the colour of the address has to be changed to the current colour, this means that when the block has to stay the same colour there will be send nothing from the controller to the storage unit. By coding it this way we are able to save space on the chip and make the process faster too.

4.1 Description of inputs and outputs

The block diagram of the section controller can be seen in figure 5. Where the in and output description is listed below.

Input colour The 'input colour' signal stands for the colour that is currently active, the colour button which has been pushed last.

Address If the controller needs information about a specific address, this address is outputted to the storage. This signal is 2 times 3 bits in parallel.

flag, set flag & colour When an address is set, the 'set flag' will be set high. The 'flag' signal will be high when the storage section is busy doing calculations and finding the requested data. The signal will become 0 when the storage unit is done searching and the data is ready on the 'colour' signal, here a 3 bit vector will be put on, 2 bits for the colour and 1 for stating whether the colour of the block has been changed already.

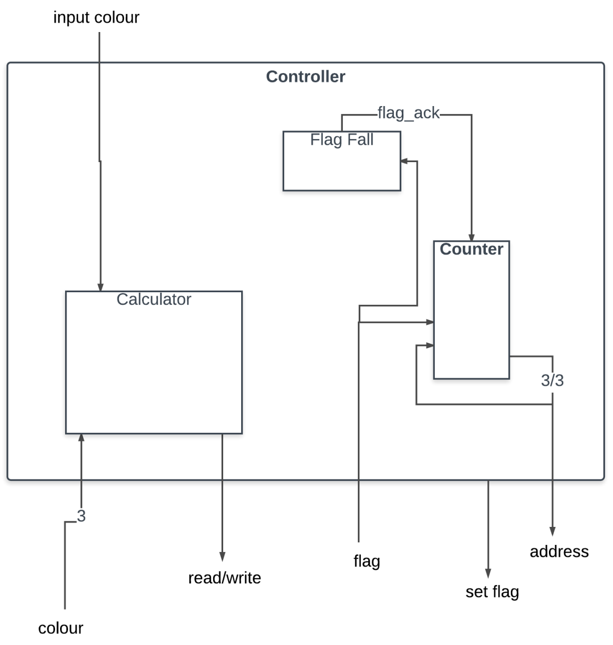


Figure 5: VBlock diagram of controller

5 VGA

To display the playing field of the game, a monitor is used which is connected to the designed chip via VGA cable. To function properly, both RGB signals for the colours as well as synchronisation signals need to be sent to the monitor's control chip.

5.1 VGA protocol

The VGA or video graphics array standard is a display communication protocol used for driving relatively low resolution displays but ideal for our use case due to it's simple and small implementation. Data gets transmitted to the display as an analogue signal over three wires representing the the amount of red green and blue in the respective pixel. The positioning of pixels gets controlled by the horizontal en vertical sync signals originating from the CRT monitor which used an electron beam deflected by coils to 'draw' an image line by line on a fluorescent screen. At the end of each line the beam would have to track back to the left of the screen and start again, this time a pixel lower. At the end of each line the horizontal sync signal has to be lowered and at the end of the frame the vertical one. A visual representation of this is given in figure 6 and a table 1 contains the corresponding timing values for the standard resolution of 640 by 480 at a refresh rate of 60 Hz. Today's digital monitors have controllers that use these two signals to associate the incoming data to the pixels on the screen.

5.2 Task description

The pixel rate for the 640 by 480 resolution at 60 Hz is 25.175 Mpixels/second. However the circuit will only have access to a 6.144 MHz clock after production and therefor the colour will only be able to change about every 4th pixel, creating an effective resolution of about 156 by 480. The VGA module's primary objective is to print out

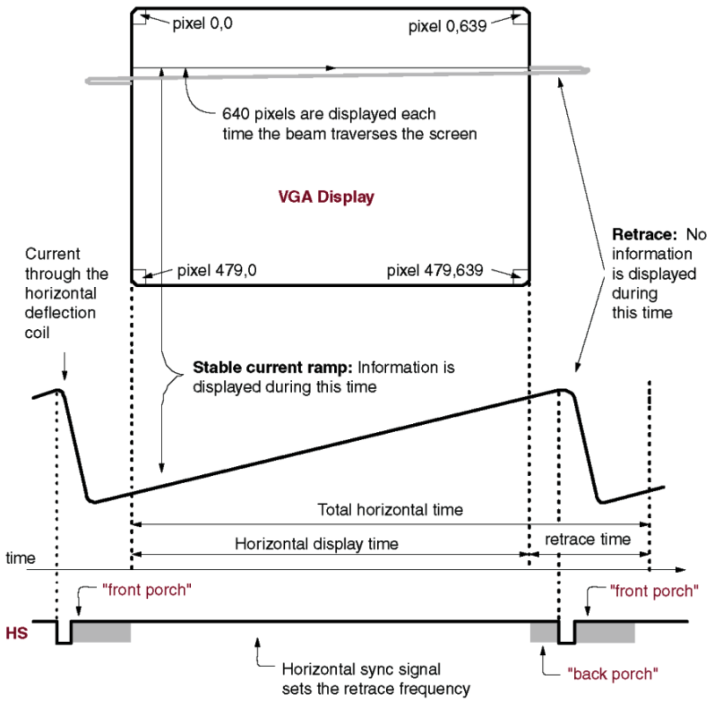

pixel 0,0
pixel 0,639
640 pixels are displayed each time the beam traverses the screen
VGA Display
Retrace: No information is displayed during this time
Current through the horizontal deflection coil
pixel 479,0
pixel 479,639
Stable current ramp: Information is displayed during this time
Total horizontal time
Horizontal display time
retrace time
time
"front porch"
"front porch"
HS
Horizontal sync signal sets the retrace frequency
"back porch"

Figure 6: Visualisation of VGA protocol *from [1]*

	Horizontal timing		Vertical timing	
Scanline part	Pixels	Time [μs]	Pixels	Time[ms]
Visible area	640	25.42	480	15.25
Front porch	16	0.64	10	0.32
Sync pulse	96	3.81	2	0.064
Back porch	48	1.91	33	1.05
Whole line	800	31.78	525	16.68

Table 1: VGA timings (640x480@60Hz) *from [2]*

the playing field. The colours of the blocks making up this playing field are stored in the chip's storage module, from which individual colours stored behind addresses can be requested. Since these colours are stored as the binary numbers 0 through 3 the VGA module has to be able to convert these codes to the corresponding bits for R, G and B. This colour decoder can't be provided with pixel colours directly by the central memory, as the fast rate at which data would be requested would flood the storage module and so limit its accessibility for the controller, discussed in Section 4. To solve this problem rows of block colours have to be temporarily stored in the VGA module itself during the time they are active on the screen, after which the row will be filled with the next row.

5.3 Description of inputs and outputs

To achieve the described operations we are going to need the connections to other modules a described below and shown in figure 7.

Clock The VGA module will use the clock signal to synchronise with the monitor standard refresh rate.

Flag, set flag, address & colour These connections are used for communicating reliably with the storage module. By pulsing the set flag we indicate we want to read the colour corresponding to the address on the bus. This will also force the flag high. The memory module can then clear the flag when the correct data is available on the colour line.

RGB This three bit vector serially sends a colour value to the monitor synchronised to the horizontal and vertical sync signals.

horizontal & vertical sync These signals need to comply with the VGA standard timings as specified in table 1.

Size select As an extra feature, the board size will be able to vary in size, which can be controlled by the size select signal.

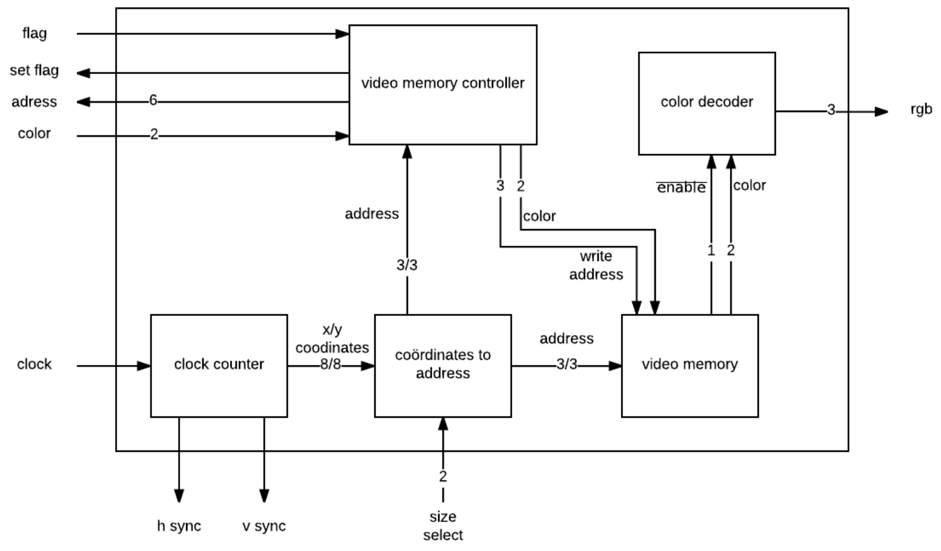


Figure 7: Block diagram of the VGA

6 RANDOM NUMBER GENERATION

In order for the game to be playable and fun, the colour of the blocks should be as random as possible. This ensures that the player will most likely play different maps. In order to assure this, a random number generator has to be made. The random number obtained will be used as an input for the map generator, which will in turn generate the map. The block diagram of the map generator can be found in figure 8.

6.1 Description of inputs and outputs

In the block diagram the following inputs and outputs can be found: restart, game reset, clk, seed, address and colour.

Restart signal The restart signal is used to turn on the random number generator and set the initial value equal to $'1'$. This is the case when the game is started and the map generator requires a random number in order to generate the map. In other words, the random number generator generates an output when enable = '1'.

Clk signal The clk or clock signal ensures that the system is synchronised with the rest of the components of the design.

Seed signal The seed is a 10 bit signal. It is generated by the random number generator and used by the map generator to make a map. Each seed generates a different map. Therefore when the same seed is used the map will be the same, allowing the player to play a map multiple times.

Address signal The address signal represents the coordinates of an element in the map matrix. The map generator combines this signal with the seed into a colour.

Colour signal The colour signal contains the colour information needed for the map. The colour signal is 2 bits long, which stands for four different colours.

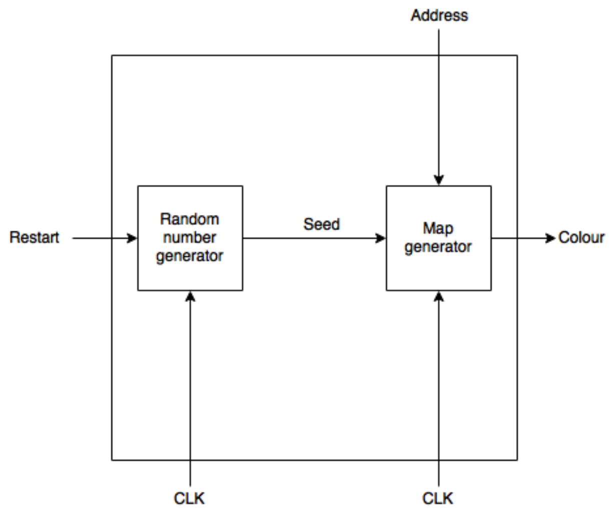


Figure 8: Block diagram of random number generator and map generator

6.2 Functionality

6.2.1 Random number generator

The goal of the random number generator is to generate a 10 bits random vector, called the seed. This vector represents the amount of different maps that can be played, because the seed is used as input for the map generator. Therefore, there are $2^{10} - 1 = 1023$, if "0000000000" is excluded, combinations for the map generator possible. The random number generator should also use the least amount of space/flip-flops, because these are limited.

6.2.2 Map generator

The main goal of the map generator is to avoid creating a pattern of four different colours. Another requirement is that the map generator should not take too much space/memory, hence contain simple logic without the use of flip-flops. It should be also fast enough, because the VGA uses this map generator to get the information needed for the display. Therefore, the map generator needs to faster than 1/60 seconds. The random map pattern is achieved by combining the address vector with the seed vector with a logic gate. This will result in different input combinations for each element of the matrix, because each address corresponds to another bit vector. This can be done several times until the logic combination leads to a two bits vector. This vector represents the four colours. Also, if the game is completed, the current map should be replaced by a new, random generated map.

7 Conclusion

References

[1] Mark Bowers and Michael Prechel. Asic vga controller. `https://www.markbowers.org/asic-vga/`, April 2013.

[2] Vga signal timing. `http://tinyvga.com/vga-timing`, November 2017.