Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
Secția de Calculatoare și Tehnologia Informației
Anul universitar 2023-2024

ZEDBOARD – COLOR SHOW

LUCRAREA DE DIPLOMĂ

Candidat: Vereș Denisa-Alexandra

Sesiunea: Iunie 2024

CUPRINS

1. INTRODUCERE 5
2. DESIGN 6
2.1. COLOR SHOW (CS) – TOP MODULE 6
2.2. CLOCK DIVIDER (CD) 7
2.3. DEBOUNCERS (DB) 10
2.3.1. DEBOUNCER 10
2.4. UART 11
2.5. COLOR MANAGER (CM) 14
2.5.1. CONFIGURATION MANAGER 16
2.5.2. DECODIFICARE UART 17
2.5.3. CULORI 19
2.5.4. DIALOGUL CU MODULUL EXTERN LM 19
2.5.5. ASSIGN DATA 20
2.5.6. CONFIGURARE CADRAN PRIN UART 22
2.5.7. DIALOGUL CU MODULUL EXTERN LM 22
2.5.8. COUNTER 23
2.6. VGA 23
2.6.1. PARAMETRII REZOLUTIE VGA 25
2.7. LED MANAGER (LM) 26
3. VERIFICARE 29
4. SIMULARE 30
5. IMPLEMENTARE 30
6. CONCLUZII 32
7. BIBLIOGRAFIE 32

*** pune diacritice

LISTA FIGURILOR

Figura 2.1 – Top Module

Figura 2.2 – Top Module simplificat

Figura 2.3 – Modulul Clock Divider

Figura 2.4 – Golden model pentru reconfigurare

Figura 2.5 – Modulul Debouncers

Figura 2.6 – Modulul UART

Figura 2.7 – Golden model pentru o comunicare valida prin UART

Figura 2.8 – Golden model pentru o comunicare invalida prin UART

Figura 2.9 – Modulul Color Manager

Figura 2.10 – Diagrama de stari finite a managerului de configuratii

Figura 2.11 – Codificarea cadranelor

Figura 2.12 – Diagrama de stari finite a managerului de cadrane

Figura 2.13 – Pozitia cadranelor

Figura 2.14 – Golden model pentru un counter din VGA

Figura 2.15 – Modulul VGA

Figura 2.16 – Explicatie SYNC

Figura 2.17 – Parametrii VGA

Figura 2.18 – Parametrii rezolutie

Figura 2.19 – Modulul Led Manager

Figura 3.1 – Design verificare modul UART individual

Figura 3.2 – Design verificare module UART, Cm si CD impreuna

Figura 5.1 – ZedBoard [1]

LISTA TABELELOR

Tabelul 2.1 – Valorile neconfigurabile ale clock dividerului

Tabelul 2.2 – Valorile configurabile ale clock dividerului

Tabelul 2.3 – Codificarea parametrilor configurabili ai modulului UART

Tabelul 2.4 – Descrierea cuvantului de comanda

Tabelul 2.5 – Codificarea modulelor configurabile

Tabelul 2.6 – Codificarea configurarilor

Tabelul 2.7 – Format primit de la UART

Tabelul 2.8 – Formatul statusului de configuratie

Tabelul 2.9 – Formatul LED-URILOR in functionarea normala

Tabelul 2.10 – Formatul LED-URILOR in modul debug

Tabelul 2.11 – Informatiile provenite de la modulul UART

Tabelul 2.12 – Informatiile provenite de la modulul CM

1. INTRODUCERE

Scopul acestui proiect este realizarea unui proiect pe placa de dezvoltare ZedBoard.

Obiectivele propuse sunt ...

Pentru realizarea proiectului se vor urma etapele de documentare, design, implementare, testare și se va finaliza prin testarea pe placă.

Aplicatii folosite in realizarea acestui proiect:

- ModelSim & QuestaSim: compilare si simulare
- Visual Studio Code: editare cod
- Wavedrom: editare modele ideale
- EasyEda: editare design
- Vivado: implementare placa
- Microsoft Excel: verificare
- Microsoft Word: editare text
- Microsoft PowerPoint: editare prezentare

*** fa o introducere a proiectului – descrie pe scurt ce face (nu uita de modurile de debug)

*** la design pune diagramele de la UART

*** la design rescrie FIFO-ul care nu mai e FIFO

*** refa golden model pentru VGA

*** fa golden model pentru DB si LM

!!!!!!!!!!!! Verifica bitul de paritate !!!!!!!!!!!!!!!

2. DESIGN

Designul este compus din 6 module principale, independente si modulare.

2.1. COLOR SHOW (CS) – TOP MODULE

input clk, rst
input btnHS, btnVS, btnUART, btnVGA
input data

output [3:0]RED, GREEN, BLUE
output HSYNC, VSYNC

Figura 2.1 – Top Module

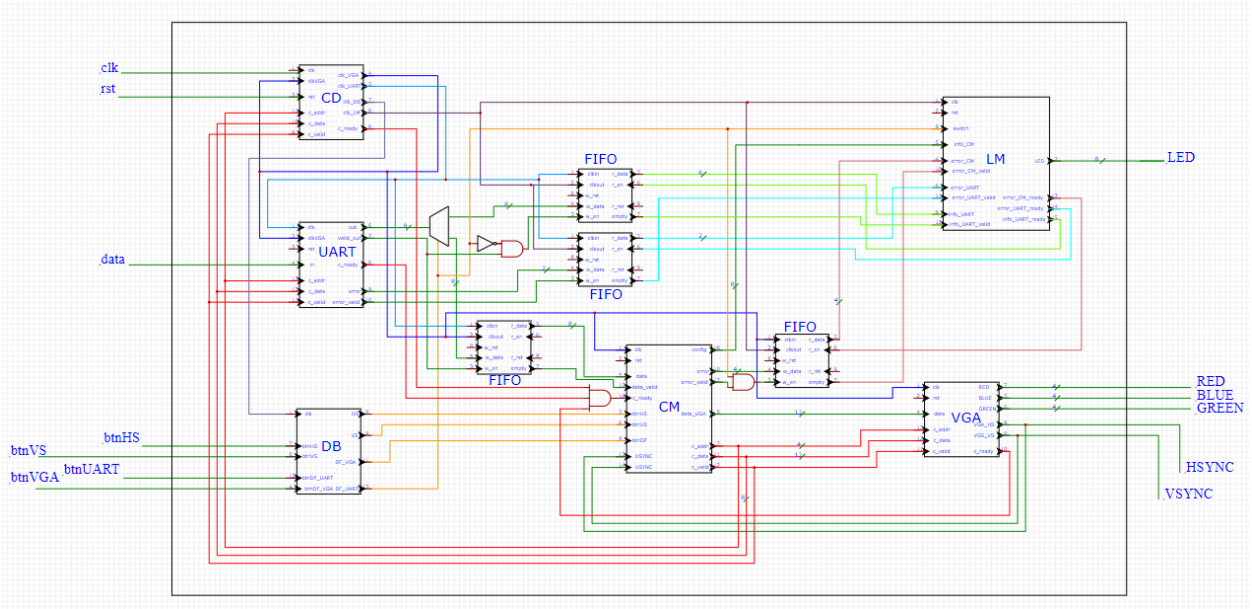
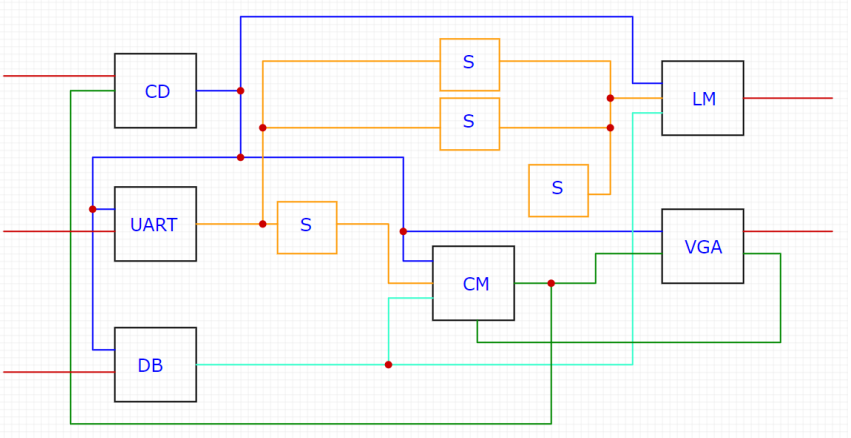


Figura 2.2 – Top Module simplificat



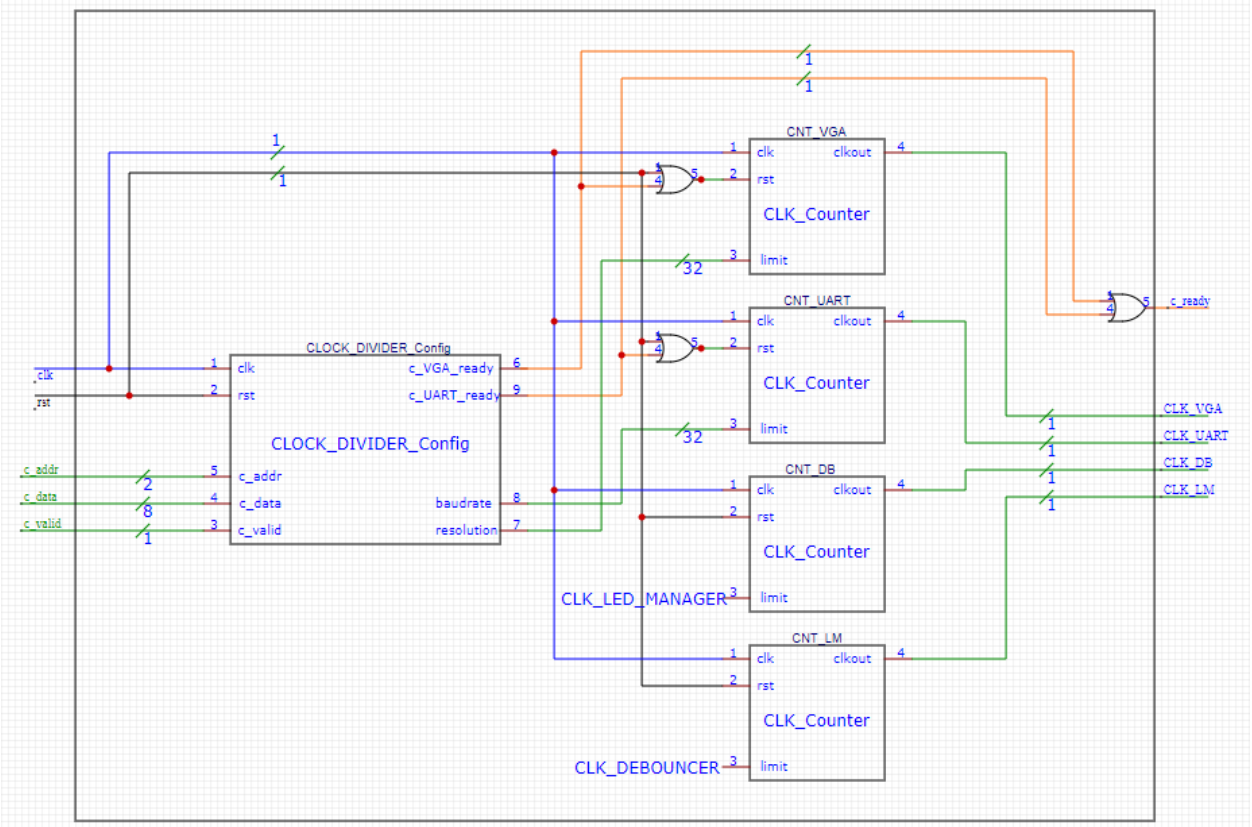
2.2. CLOCK DIVIDER (CD)

input clk, rst
input clkinVGA

input [3:0]c_addr
input [7:0]c_data
input c_valid
output c_ready

output clk_VGA
output clk_UART
output clk_DB
output clk_LM

Figura 2.3 – Modulul Clock Divider



Modulul ClockDivider genereaza impulsuri cu diferite frecvente in functie de frecventa primita la intrare prin clkin.

Iesirile:

- ✓ clkVGA – frecventa: 25Mhz – 65MHz
  - in functie de rezolutia modulului VGA, frecventa este:
    - 640x480 – 25MHz
    - 800x600 – 40MHz
    - 1024x768 – 65MHz
  - frecventa intiala este cea corespunzatoare rezolutiei de 640x480, adica 25MHz.
- ✓ clkUART – frecventa: 38.4kHz – 921.6kHz
  - in functie de baudrate din modulul UART, frecventa este:
    - 2400 –38.4kHz
    - 4800 – 76.8kHz
    - 9600 – 153.6kHz
    - 19200 – 307.2kHz
    - 57600 – 921.6kHz
    - 11200 – 1843.2kHz
  - frecventa initiala este cea corespunzatoare unui baudrate de 9600, adica 9.6kHz
- ✓ clkCM – frecventa: 25MHz – 65MHz
  - in functie de rezolutia modulului VGA, specificata la punctul anterior
- ✓ clkDEB – frecventa 20Hz
- ✓ clkLED – frecventa 1HZ

Modulul este format din:

- 4 numaratoare, unul pentru fiecare frecventa de iesire:
  - CNT_VGA – limitele conform tabelului de mai jos
  - CNT_UART – limitele conform tabelului de mai jos
  - CNT_DB – limita este CLK_DEBOUNCER
  - CNT_LM – limita este CLK_LED_MANAGER

Modulul Clock_Divider_Config, care primeste noile configuratiile si transmite configuratiile actuale ale modulelor UART si VGA spre numaratoare. La primirea unei noi configuratii, numaratorul corespunzator modulului configurat se reseteaza.

Limitele pentru valorile neconfigurabile sunt urmatoarele:

limit	value
clk_DB	11 250 000
clk_LM	225 000 000

Tabelul 2.1 – Valorile neconfigurabile ale clock dividerului

Modificarea configuratiei se intampla doar pentru urmatoarele cazuri:

c_addr	COD	c_data[2:0]	COD	limit	value
UART_BAUDRATE_ADDR	0100	BAUDRATE_2400	000	CLK_BAUDRATE_2400	5859
		BAUDRATE_4800	001	CLK_BAUDRATE_4800	2930
		BAUDRATE_9600	010	CLK_BAUDRATE_9600	1465
		BAUDRATE_19200	011	CLK_BAUDRATE_19200	732
		BAUDRATE_57600	100	CLK_BAUDRATE_57600	244
		BAUDRATE_112000	101	CLK_BAUDRATE_112000	122
VGA_RESOLUTION_ADDR	1000	VGA_640x480	x00	CLK_VGA_640x480	9
		VGA_800x600	x01	CLK_VGA_800x600	6
		VGA_1024x768	x10	CLK_VGA_1024x768	3

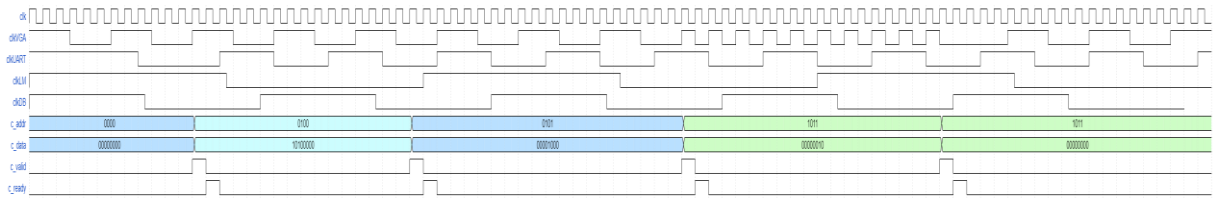
Tabelul 2.2 – Valorile configurabile ale clock dividerului

Pentru a se modifica limitele semnalul c_valid trebuie sa fie active, iar dupa modificarea configuratiei, semnalul c_ready va fi reactivat. Pentru oricare alta adresa limitele numaratoarelor nu se vor modifica si numaratoarele nu vor fi resetate.

In urmatoarea diagrama de semnale se pot observa 4 modificari successive ale configuratiei modulelor UART si VGA:

- BAUDRATE – duce la resetarea semnalului clock_UART si la modificarea limitei numaratorului CNT_UART
- PARITY BIT – nu se modifica nimic, deoarece doar baudrate-ul si rezolutia influenteaza semnalele de clock pentru modulele UART si VGA
- Rezolutie – duce la resetarea semnalului clock_UART si la modificarea limitei numaratorului CNT_UART
- Rezolutie – duce la resetarea semnalului clock_VGA si la modificarea limitei numaratorului CNT_VGA

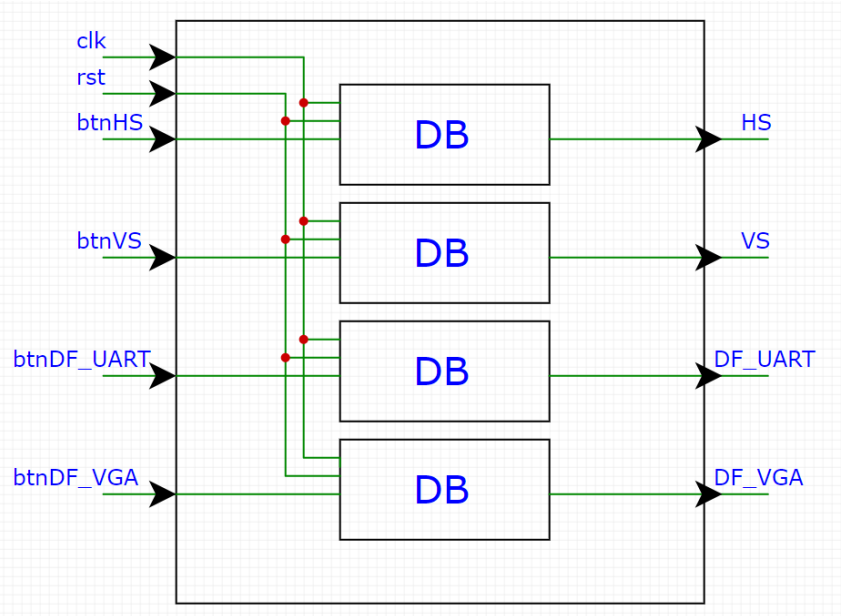
Figura 2.4 – Golden model pentru reconfigurare



2.3. DEBOUNCERS (DB)

input clk
input rst
input btnHS, btnVS, btnDF_UART, btnDF_VGA
output HS, VS, DF_UART, DF_VGA

Figura 2.5 – Modulul Debouncers



Modulul Debouncers este format din 4 module Debouncer independente.

Frecventa modulului este 20Hz si nu este configurabil.

2.3.1. DEBOUNCER

input clk
input rst
input button
output signal

Modulul Debouncer primeste ca intrare un semnal instabil si il transforma intr-un semnal stabil, folosind 2 bistabile si un numarator cu limită prestabilită.

Semnalul de iesire se modifica instant la primirea valorii ‘0’ si asteapta stabilizarea semnalului pentru o anumita perioada configuratbila in cazul primirii valorii de ‘1’.

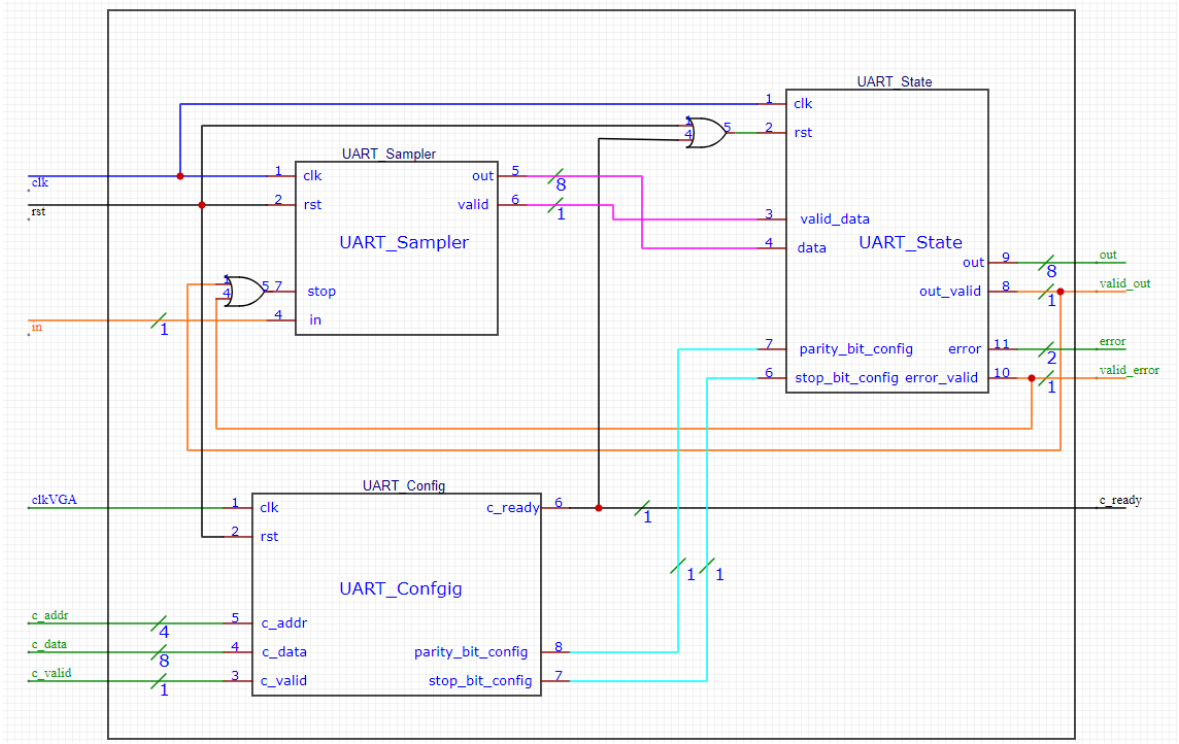
2.4. UART

input clk, rst
input clkVGA
input in

input [3:0]c_addr
input [7:0]c_data
input c_valid
output c_ready

output [1:0]error // datele de trimis la modulul LEDManager
output valid_error // datele sunt gata de trimis la modulul LEDManager
output [7:0]out // datele primite prin UART pe 8 biti
output valid_out // datele primite prin UART sunt gata

Figura 2.6 – Modulul UART



Modulul configurabil UART are in strucutra sa registrii in care sunt salvate valorile configurabile.

Modulul UART este initializat cu BAUDRATE 9600, PARITY BIT none, 1 STOP BIT si 8 DATA BITS si cu frecventa de 153.6kHz.

Parametrii BAUDRATE, PARITY BIT si STOP BITS sunt configurabili, dar DATA BITS nu se poate modifica. Parametrul BAUDRATE influenteaza modulul CD, iar parametrii PARITYBIT si STOPBITS influenteaza modulul UART.

Modificarea configuratiei:

- c_valid trebuie sa fie activ, pentru a semnaliza primirea unei noi configuratii
- c_addr si c_data, formate din 4 si 8 biti sunt codificate conform urmatorului tabel:

c_addr	COD	c_data[1:0]	COD
UART_PARITY_ADDR	0101	PARITYBIT_NONE	00
		PARITYBIT_ODD	11
		PARITYBIT_EVEN	10
UART_STOP_ADDR	0110	STOPBITS_1	x0
		STOPBITS_2	x1

Tabelul 2.3 – Codificarea parametrilor configurabili ai modulului UART

- c_ready – dupa configurare este reactivat
- dupa configurare se reseteaza valorile si orice transmisie in curs de efecuare este intrerupta

Modulul UART este format din 3 module independente:

- UART_Sampler: se ocupa de esantionarea semnalului (un counter care numara cati biti de 0 si 1 sunt esantionati in interiorul semnalului si un comparator care stabileste bit-ul de iesire)
  - Pentru a evita situatia de sincronizare gresita a modulului UART, in care este posibil sa se transmita date gresite si sa ramana nesincronizat, bitii primiti sunt esantionati.
  - Se iau cate 16 esantioane pentru fiecare bit primit, din care primele si ultimele 3 esantioane sunt ignorate, iar din celelalte esantioane se calculeaza valoarea de iesire.

- UART_Config: La primirea unei noi configuratii modulul isi actualizeaza registrele cu valorile parametrilor modificati si reseteaza modulului UART_State pentru a intrerupe orice comunicatie neincheiata

- UART_State: Verifica daca informatia primita respecta protocolul UART:
  - Primul bit esantionat reprezinta bit-ul de START, care trebuie sa fie 0. In caz contrar se considera ca a fost un start fals si se reincepe transmisia.
  - Urmatorii 8 biti esantionati reprezinta bitii de DATE, care pot lua valoare de 0 sau 1.

- In cazul in care paritatea este activata, urmatorul bit reprezinta bit-ul de PARITY, care este verificat. In cazul in care paritatea nu este valida, se semnaleaza eroarea, se considera ca datele transmise au fost corupte si se reincepe transmisia.
- Urmatorul bit/biti reprezinta bit-ul/bitii de STOP, care trebuie sa fie pe 1. In caz contrar se semnaleaza eroarea, se considera ca datele transmise nu au fost esantionate corect si se reincepe transmisia.
- Daca toti bitii verificati (START, PARITY, STOP) sunt valizi, atunci counicaea a avut succes.

Erorile trimise de modulul UART au urmatoarea configuratie:

- 00 – eroare bit de STOP
- 01 – eroare bit de PARITY
- 10 – eroare bit de IDLE
-

Formatul informatiilor trimise la LM prin info sunt detaliate la sectiunea LM.

In urmatoarele diagrame se poate observa o comunicare reusita si o comunicare nereusita datorita nerespectarii protocolului UART pentru bit-ul de STOP.

Figura 2.7 – Golden model pentru o comunicare valida prin UART

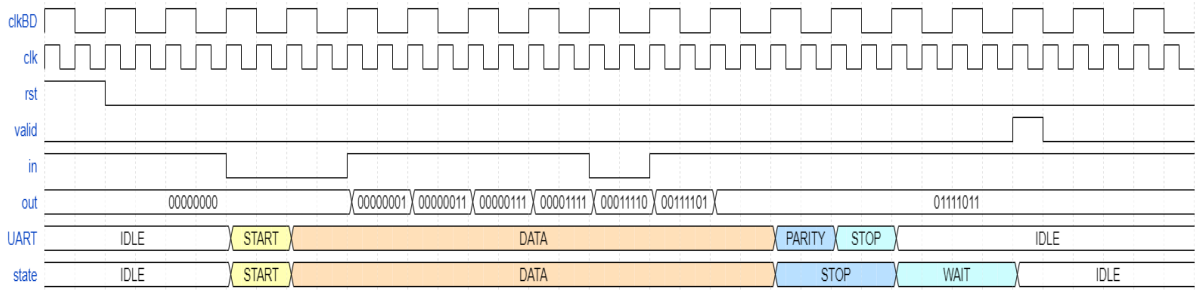
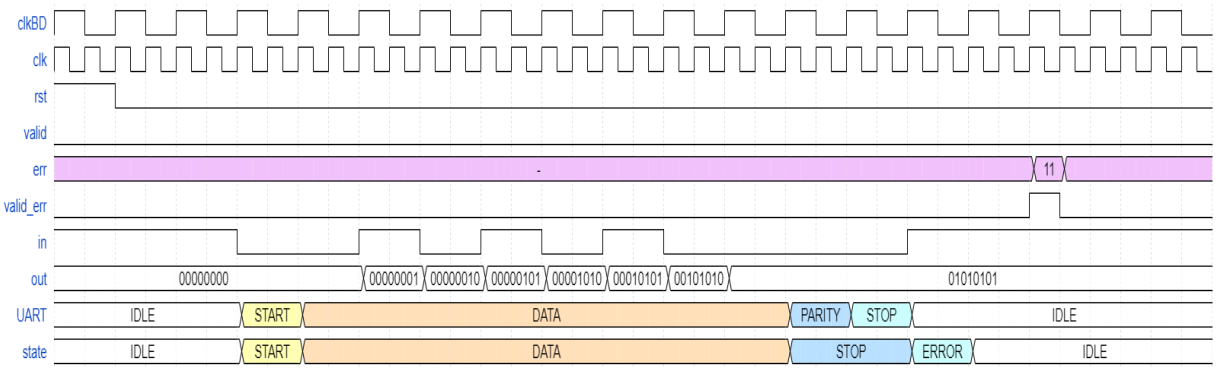


Figura 2.8 – Golden model pentru o comunicare invalida prin UART

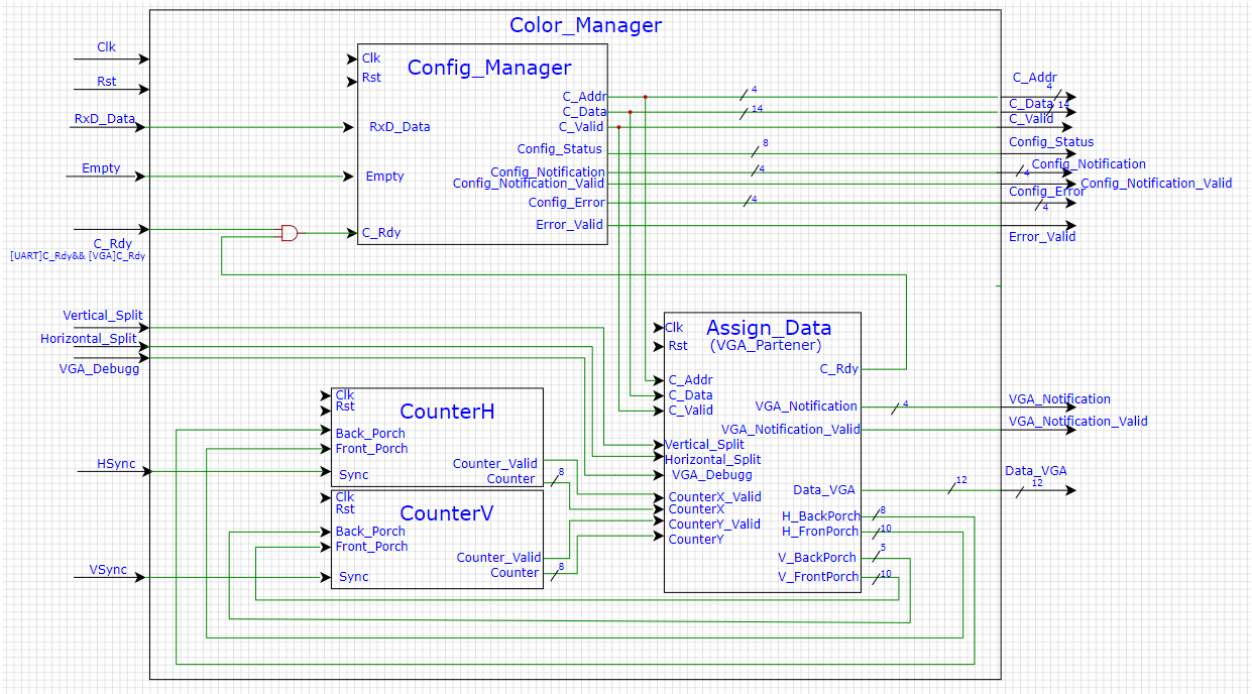


2.5. COLOR MANAGER (CM)

input clk
input rst
input [7:0] RxD_Data
input Empty
input C_Rdy
input Vertical_Split
input Horizontal_Split
input VGA_Debugg
input HSync
input VSync

output [4:0]C_addr
output [14:0]C_Data
ouput C_Valid
output [3:0] CM_Err
output [3:0] Valid_Err
output [3:0]VGA_Not
output [3:0]Valid_VGA_Not
output [11:0]Data_VGA

Figura 2.9 – Modulul Color Manager



Formatul primit de la utilizator:

Pozitie bit	Descriere Configuratie	Configuratie	Descriere Culoare	Culoare	Pozitie bit
15	Selectie Configuratie/ Culoare	1	Selectie Configuratie/Culoare	0	15
14	Adresa unitate configurabila 01 UART 10 VGA	01/10	-	0	14
13			Arata pozitia cadranului sus(0)/jos(1)	0/1	13
12	Adresa registru din unitate *tabel registru	x	Locatia cadranului: stanga(0) /dreapta(1)	0/1	12
11		x	Codul culorii convertit din hexazecimal Site pentru aflarea codului	x	11
10	Codificare configuratie *tabel config	x		x	10
9		x		x	9
8		x		x	8
7	Biti nefolisiti	0		x	7
6		0		x	6
5		0		x	5
4		0		x	4
3		0		x	3
2		0		x	2
1		0		x	1
0		0		x	0

Tabelul 2.4 – Descrierea cuvantului de comanda

Table registru			
Unitate	Valoare		Ce face?
UART	0	0	Configurare BoudRate
	0	1	Configurare Parity
	1	0	Configurare Bit Stop
VGA	0	0	Culoare pentru VGA prin UART
	1	0	Configurare cadran nou VGA
	1	1	Configurare rezolutie pentru VGA

Tabelul 2.5 – Codificarea modulelor configurabile

Tabel configuratii				
Ce face?	Valori			Valori codificate
Configurare BoudRate	0	0	0	2400
	0	0	1	4800
	0	1	0	9600
	0	1	1	19200
	1	0	0	57600
	1	0	1	112000
Configurare Parity	X	0	0	no parity
	X	1	1	odd
	X	1	0	even
Configurare Bit Stop	X	x	0	1 stop bits
	X	x	1	2 stop bits
Configurare rezolutie pentru VGA	X	0	0	640x480 default
	X	0	1	800x600
	X	1	0	1024x768
Configurare cadran nou VGA	X	0	1	vertical split
	X	1	0	horizontal split
	X	1	1	vertical and orizontal split
	*only when split switches are disabled			

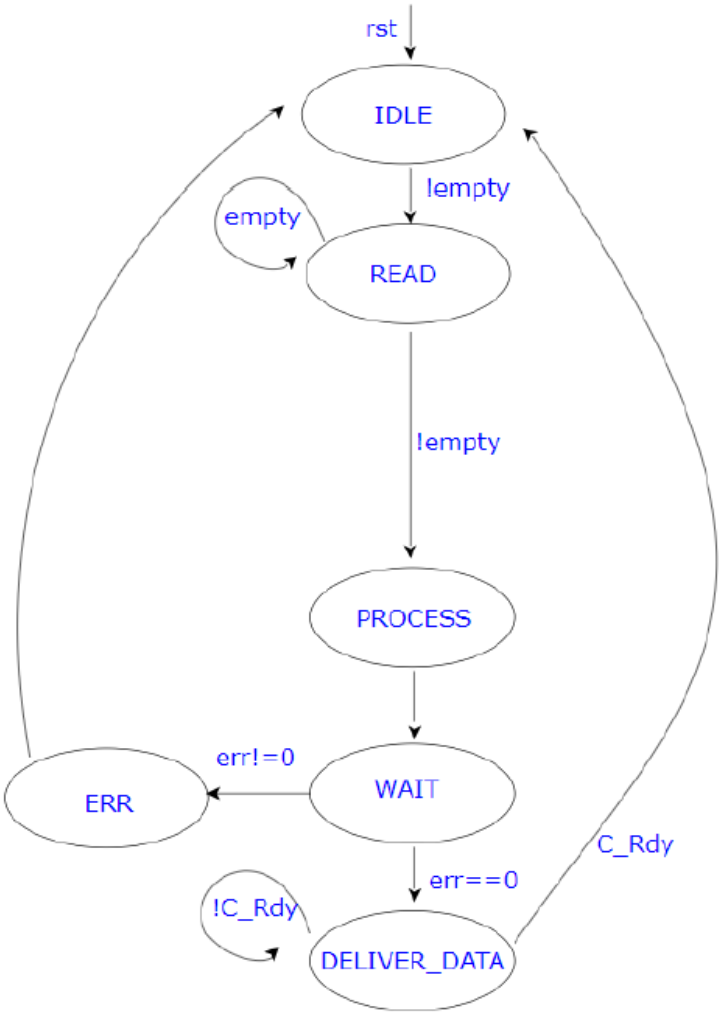
Tabelul 2.6 – Codificarea configurarilor

2.5.1. CONFIGURATION MANAGER

Cel care comanda magistrala de configuratii, semnalizeaza erorile / cuvintele gresite transmise de UART. Tot el verifica date transmise de UART astfel incat sa nu existe configuratii invalide.

Pe langa configuratii gestioneaza modul Debug al UART-ului si transmite culori pentru VGA (impreuna cu cadranul sau).

Figura 2.10 – Diagrama de stari finite a managerului de configuratii



2.5.2. DECODIFICARE UART

Pozitie	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Valoare	1	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0

Tabelul 2.7 – Format primit de la UART

Addr_Module – id unitate configurabila, trimis prin C_Addr[3:2]

Addr_Reg – id registru din unitatea configurabila (doar UART); trimis tot prin C_Addr, adica C_Addr[1:0]

C_Data – noua configuratie codificata: 00 0000 0000 0xxx

Fiecare unitate configurabila se codifica prin Addr_Module. Cand o unitate sesiseaza codul sau pe C_Addr va presupune ca C_Data si C_Valid sunt adresate sale.

- VGA_Control – 10
- UART_Config – 01

Cand vine vorba de un registrul ce tine de un aspect configurabil din UART sau de configuratia care ajunge la VGA, vorbim despre Addr_Reg:

- 10 00 – Culoare pentru VGA
- 10 10 – Configurare cadran nou VGA
- 10 11 – Configurare rezolutie pentru VGA
- 01 00 – Configurare BoudRate
- 01 01 – Configurare Parity
- 01 10 – Configurare Stop

C_Data se foloseste astfel:

- Pentru UART - C_Data[2:0]:
  - BoudRate 00
    - 000 – 2400
    - 001 – 4800
    - 010 – 9600
    - 011 – 19200
    - 100 – 57600
    - 101 – 112000
  - Parity 01
    - x00 – no parity
    - x11 – odd
    - x10 – even
  - StopBit 10
    - xx0 – 1 stop bit
    - xx1 – 2 stop bit
- Pentru VGA
  - Config 11 – C_Data[1:0] :
    - 00 – 640x480 – defaul
    - 01 – 800x600
    - 10 – 1024x768
  - Configurare cadran nou 10 – C_Data[1:0] (luata in cosiderare doar in cazul ambele switch-uri de split <intrari Assign_Color> sunt dezactivate, adica ambele pe 0):
    - 01 – ecran impartit vertical
    - 10 – ecran impartit orizontal
    - 11 – ecran impartit orizontal si vertical

2.5.3. CULORI

Format primit de la UART: 00xx xxxx xxxx xxxx

Format trimis:

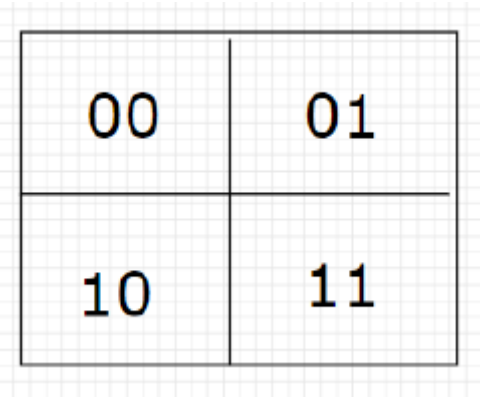
- C_addr: 10 00
- C_Data: xx xxxx xxxx xxxx

Pentru un cuvant de date, ultimii bitii, C_Data[11:0] semnifica bitii de culoare.

In cazul unui cadran:

- bitul 14 arata pozitia cadranului sus(0)/jos(1)
- bitul 13 arata pozitia cadranului: stanga(0) /dreapta(1)

Figura 2.11 – Codificarea cadranelor



2.5.4. DIALOGUL CU MODULUL EXTERN LM

Trimite Configuratia actuala in mod permanent prin Config_Status.

Pozitia bitilor	**7 6 5**	**4 3**	**2**	**1 0**
Continut	Boudrate	Parity	Stop Bit	VGA Rezolution

Tabelul 2.8 – Formatul statusului de configuratie

Parametrul pentru reset este DEFAULT_CONFIG = 8'b01000000, BoudRate default 9600, No Parity, Un bit de stop si rezolutia 640x480.

Config_Notification este trimis doar in cazul se primeste o configuratie pe care exista codificare, adica se afla in intervalul definit pentru Codificarea datelor primite de la UART.

Parametrii acestuia sunt:

- parameter CORRECT_BAUDRATE_CONFIGURATION = 4'b1100
- parameter CORRECT_PARITYBIT_CONFIGURATION = 4'b1101
- parameter CORRECT_STOPBIT_CONFIGURATION = 4'b1110
- parameter CORRECT_RESOLUTION_CONFIGURATION = 4'b1111

Config_Error este trimis doar in cazul se primeste o configuratie pe care nu exista codificare, adica nu se afla in intervalul definit pentru Codificarea datelor primite de la UART.

Parametrii acestuia sunt:

- parameter FAILED_CONFIGURATION_ADDRESS = 4'b0011
- parameter FAILED_BAUDRATE_CONFIGURATION = 4'b0100
- parameter FAILED_PARITYBIT_CONFIGURATION = 4'b0101
- parameter FAILED_QUADRAN_CONFIGURATION = 4'b0110
- parameter FAILED_RESOLUTION_CONFIGURATION = 4'b0111

2.5.5. ASSIGN DATA

Primeste counter-ul de pixel, configuratia vga actuala, starea culorii si transmite culoare pentru VGA_Control, dar si configuratia VGA actuala spre Counterele interne.

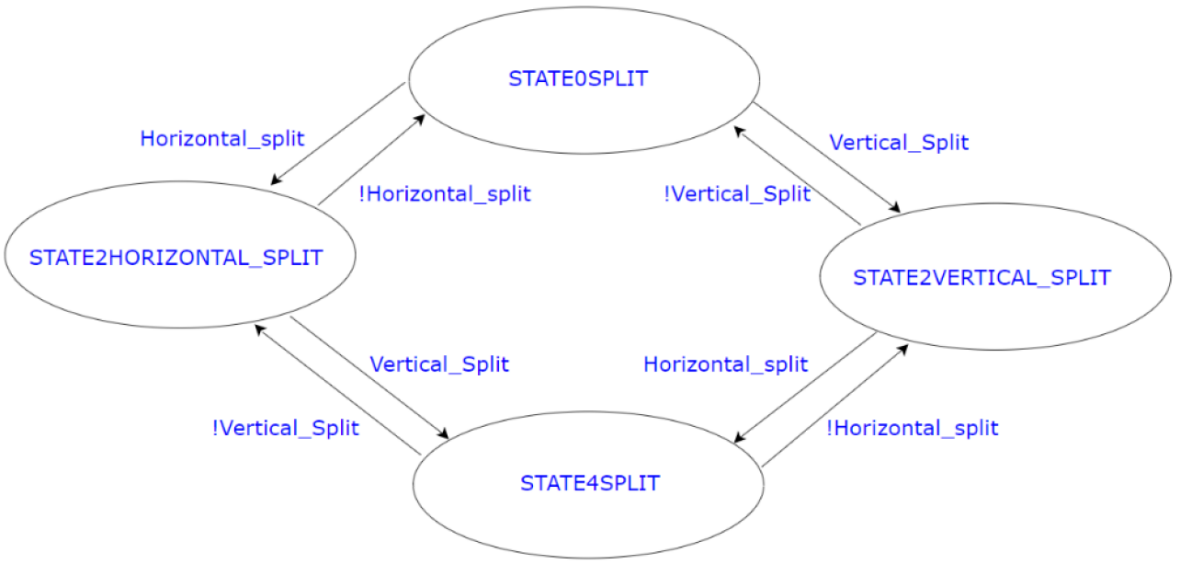
Daca nu este activat switch-ul de VGA_Debugg, modulul verifica datele primite de la UART si le salveaza. Primirea unei culori pentru un cadran inactiv se rezuma la salvarea culorii respective- nu este afectata comandarea VGA_Control.

In cazul in care se primeste o configuratie noua, aceasta intra in viguare instant. La primirea unei culori noi aceasta este afisata imediat, iar ochiul uman nu va observa trecerea dintre culori.

Starile tin de intrarea Horizontal_split si Vertical_Split (comanda) astfel:

- STATE0SPLIT - 00
- STATE2VERTICAL_SPLIT - 01
- STATE2HORIZONTAL_SPLIT - 10
- STATE4SPLIT – 11

Figura 2.12 – Diagrama de stari finite a managerului de cadrane



Modulul este initializat cu State0Split-00, care semnifica o singura culoare pe ecran. In aceasta stare asteapta sa primeasca o culoare in Left_UP.

Cererea de impartire pe verticala State2VSplit-01 va afisa culoarea din Left_UP si Right_UP.

Cererea de impartire pe orizontala State2HSplit-10 va afisa culoarea din Left_UP si Left_DOWN.

Starea State4Split-11 se va folosi de toate cele 4 culori din toate registrele dedicate.

Figura 2.13 – Pozitia cadranelor

Left_UP 00	Right_UP 01
Left_DOWN 10	Right_DOWN 11

Transmiterea culorilor pentru VGA cand acesta este in zona activa se face in functie de rezolutie si coordonatele CounterX si CounterY.

- Left_UP –> Counter_X < zona activa orizonatala/2 si Counter_Y < zona activa verticala/2;
- Right_UP –> Counter_X >= zona activa orizonatala/2 si Counter_Y < zona activa verticala/2;
- Left_DOWN –> Counter_X < zona activa orizonatala/2 si Counter_Y >= zona activa verticala/2;
- Right_DOWN–> Counter_X >= zona activa orizonatala/2 si Counter_Y >= zona activa verticala/2.

*Counter_X si Counter_Y sunt cu un ciclu de clock in fata celor din VGA

2.5.6. CONFIGURARE CADRAN PRIN UART

In cazul in care Vertical_Split si Horizontal_Split sunt inactive, Assign_Data tine cond de C_Addr 1010 care inseamna o noua configuratie de cadran.

Configurare cadran nou C_Addr 1010 , iar C_Data [1:0] poate fi:

- 01 – ecran impartit vertical
- 10 – ecran impartit orizontal
- 11 – ecran impartit orizontal si vertical

2.5.7. DIALOGUL CU MODULUL EXTERN LM

Comunicarea consta in semnalele VGA_Notification si VGA_Notification_Valid.

VGA_Notification_Valid se activeaza pentru un ciclu de clock cand exista o noua notificare.

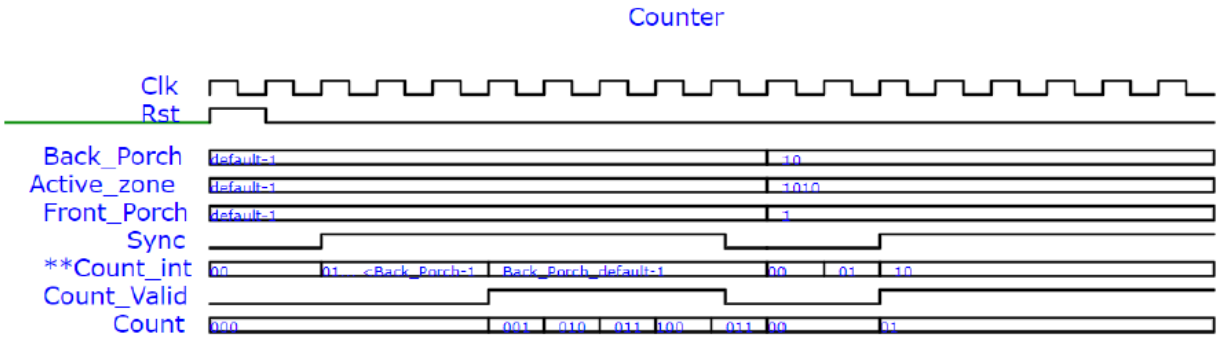
VGA_Notification are urmatoarii parametrii:

- parameter STATE0SPLIT_CHANGE = 4'b1000
- parameter STATE2VERTICAL_CHANGE = 4'b1001
- parameter STATE2HORIZONTAL_CHANGE = 4'b1010
- parameter STATE4SPLIT_CHANGE = 4'b1011

2.5.8. COUNTER

Primeste semnalul Sync corespunzator (dintre HSync si VSync), Active zone, Back_Porch-ul si Front-Porch-ul – toti parametrii unei configuratii VGA pentru configuratia actuala transmite 2 semnale, Count_Valid, care semnalizeaza ajungerea Pixel Counter in zona activa, si Count_X/Y care tine de coordonata Pixelului in plan. Counter trimite coordonatele cu un ciclu de clock mai repede decat VGA pentru ca acesta sa primesca culorile la timp.

Figura 2.14 – Golden model pentru un counter din VGA



2.6. VGA

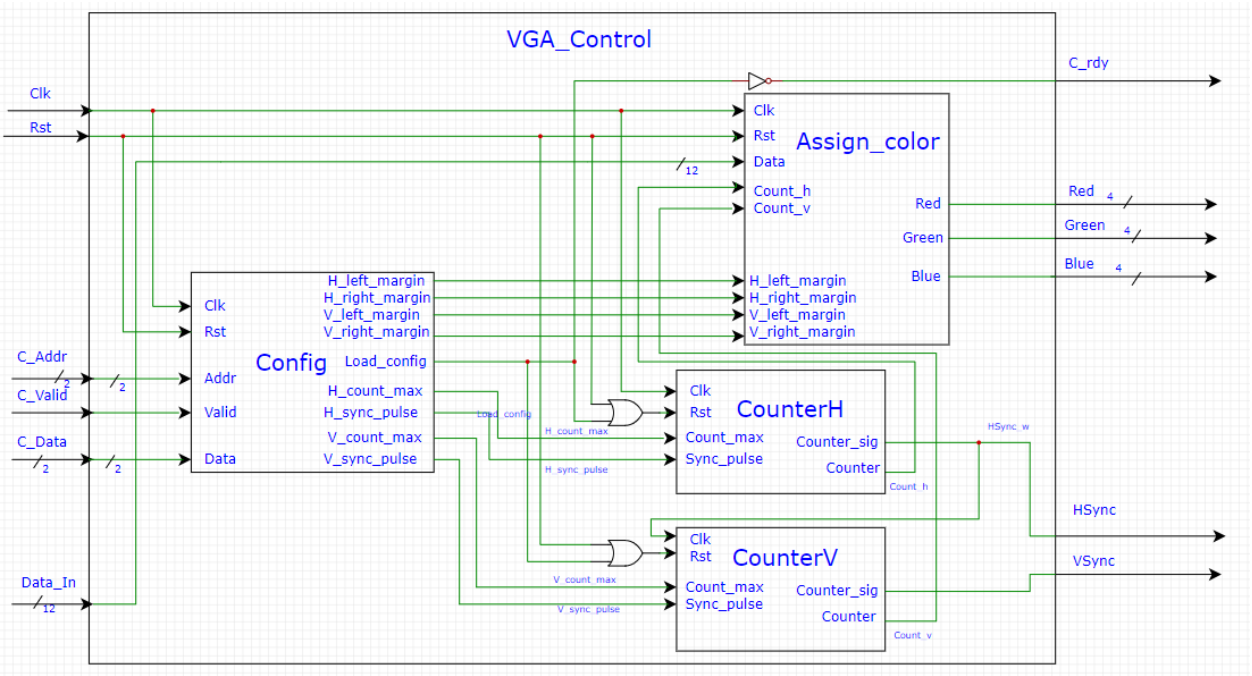
input clk
input rst
input [1:0] C_addr
input C_valid
input [1:0] C_data
input [11:0] Data_in

output [3:0]Red
output [3:0]Green
output [3:0]Blue
ouput HSync
ouput Vsync

Acest modul comanda iesirile pentru portul VGA.

VGA_Control controleaza ieșirea pentru VGA astfel încât avem cele 3 culori principale care vor forma culoarea finală – dată prin intrarea Data_in[11:0].

Figura 2.15 – Modulul VGA



Module interne:

- Config - distribuie mai departe informatiile primite de la magistrala de configuratii.
- Counter - 2 instantieri: una pentru portiunea verticala, alta pentru portiunea orizonatala. Un numarator care mai furnizeaza si semnalele HSync si VSync.
- Assign_color - determina daca numaratorul pixelului se afla in zona activa, iar in acest caz trimite mai departe culorile primite la intrarea modulului VGA_Control.

Se pot alege dintre rezolutiile:

- 640x480 (00- default)
- 800x600 (01)
- 1024x768 (10)
- Cazul 11 este nevalid, deci ignorat.

HSync spune ecranului când să se miște la o nouă coloană de pixeli, iar VSync transmite ecranului când să înceapă un nou cadru- momentele sunt salvate in V_State si H_State, ele sunt definite de cel mai semnificaiv bit, iar restul de 12 biti tin de coorodonatele pixelului curent(in cazul default nu sunt folositi toti bitii).

VSync

- 0 – moment inoportun pentru mutare cursor la noua coloana.
- 1 – moment oportun pentru noua coloana.

HSync

- 0 – moment inoportun pentru mutare la un nou frame.
- 1 - moment oportun pentru un nou frame.

Figura 2.16 – Explicatie SYNC

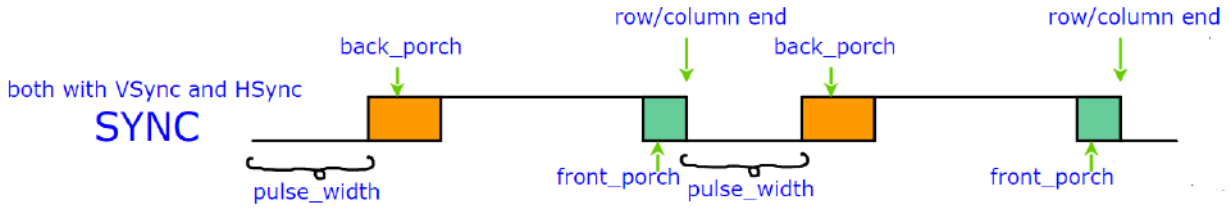
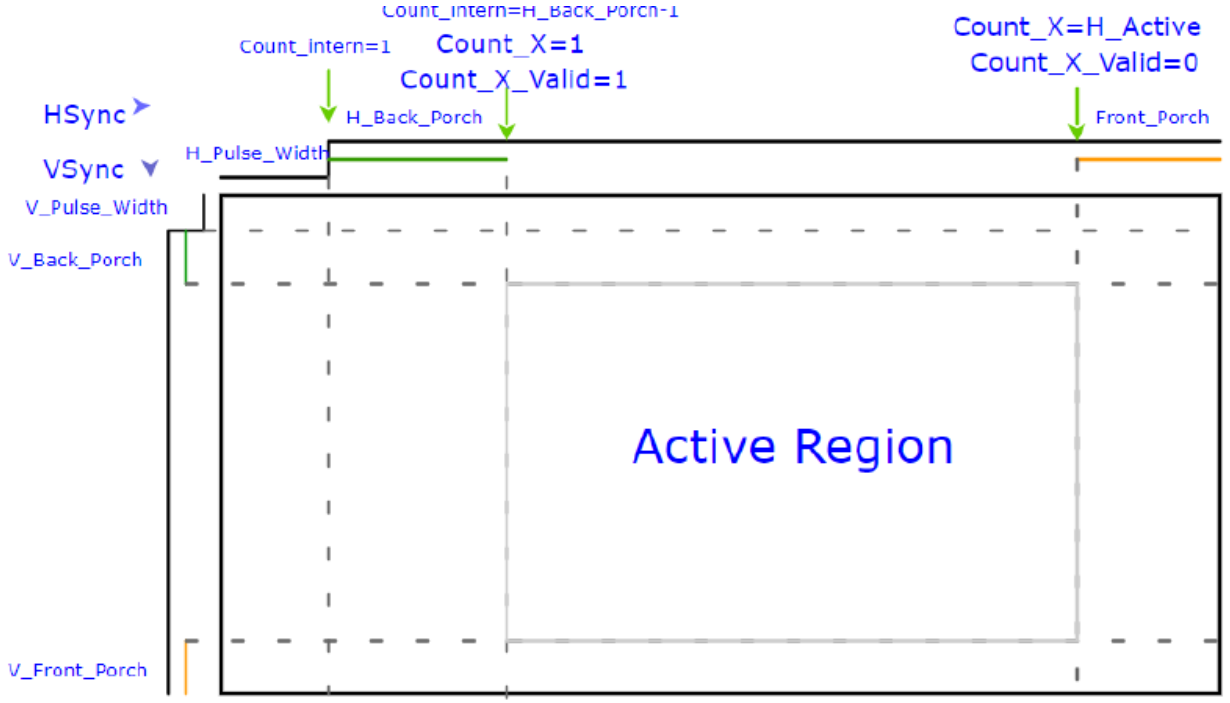


Figura 2.17 – Parametrii VGA



2.6.1. PARAMETRII REZOLUTIE VGA

Pixelul are sau nu culoarea data prin intrare si in functie de Sync Pulse, care inseamna perioada in care HSync sau VSync sunt inactive(0 logic), dar si de Front Porch si Back Porch.

Figura 2.18 – Parametrii rezolutie

Format	Horizontal (in Pixels)				Vertical (in Lines)			
	Active Video	Front Porch	Sync Pulse	Back Porch	Active Video	Front Porch	Sync Pulse	Back Porch
640x480, 60Hz	640	16	96	48	480	11	2	31
800x600, 60Hz	800	40	128	88	600	1	4	23
1024x768, 60Hz	1024	24	136	160	768	3	6	29

In cazului unei rezolutii noi, contoarele reincep de la 0, dar limitele acestora sunt schimbate in functie de tabelul de mai sus. O rezolutie noua presupune un reset local, dar pastrarea noii configuratii. Resetul local al contoarelor este asigurat de combinatie logica dintre semnalul extern de Reset si semnalul intern Load_Config.

2.7. LED MANAGER (LM)

input clk, rst
input UART_data_debug_switch
input [7:0]UART_data // erorile provenite de la modulul UART
input [1:0]UART_errors // datele provenite de la modulul UART
input [3:0]CM_errors // erorile provenite de la modulul CM
input UART_data_valid, UART_errors_valid, CM_errors_valid
input UART_info_empty, UART_error_empty, CM_error_empty
input [7:0] config_notification

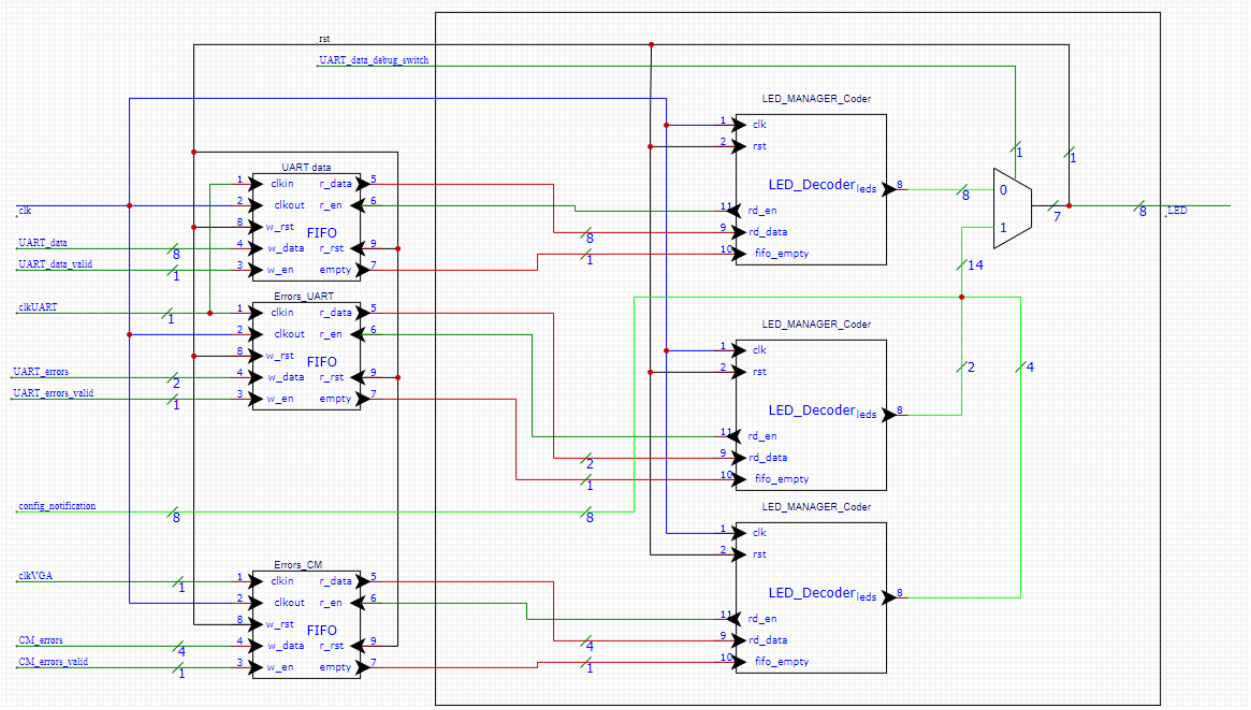
output [7:0]leds //8 LED-uri

Modulul LEDManager functioneaza la o frecventa de 1Hz.

Modulul primeste informatii de la celelalte module (actualizari, erori) si le transpune in output vizual, pe LED-uri.

Modulul primeste date de la 3 surse diferite: modulul principal CS, UART si CM. Fiecare informatie activeaza o anumita combinatie de LED-uri.

Figura 2.19 – Modulul Led Manager



Pentru cazul in care UART_data_debug_switch nu este activ, LED-urile vor fi aprinse conform urmatorului tabel. LED-urile vor afisa constant configuratia actuala a modulelor UART si VGA. LED-urile folosite pentru afisarea notificarilor si a erorilor, se vor aprinde timp de 1 secunda, dupa care se vor stinge.

LED	VALOARE
U14	RESET (1)
U19	UART_data_debug_switch (1)
W22 V22	EROARE UART
U21 U22 T21 T22	EROARE CM

Tabelul 2.9 – Formatul LED-URILOR in functionarea normala

Pentru cazul in care UART_data_debug_switch este activ, LED-urile vor fi aprinse conform urmatorului tabel. LED-urile corespunzatoare valorii UART_data se vor aprinde timp de 1 secunda, dupa care se vor stinge.

LED	VALOARE
U14	UART_data[7]
U19	UART_data[6]
W22	UART_data[5]
V22	UART_data[4]
U21	UART_data[3]
U22	UART_data[2]
T21	UART_data[1]
T22	UART_data[0]

Tabelul 2.10 – Formatul LED-URILOR in modul debug

EROARE	CODIFICARE
NO_ERRORS	00
FAILED_STOP_BITS	11
FAILED_PARITY_BIT	01
FAILED_IDLE_BITS	10

Tabelul 2.11 – Informatiile provenite de la modulul UART

EROARE	CODIFICARE
NO_ERRORS	0000
FAILED_CONFIGURATION_ADDRESS	0011
FAILED_BAUDRATE_CONFIGURATION	0100
FAILED_QUADRAN_CONFIGURATION	0101
FAILED_STOPBITS_CONFIGURATION	0110
FAILED_RESOLUTION_CONFIGURATION	0111
CORRECT_BAUDRATE_CONFIGURATION	1100
CORRECT_PARITYBIT_CONFIGURATION	1101
CORRECT_STOPBITS_CONFIGURATION	1110
CORRECT_RESOLUTION_CONFIGURATION	1111
WRONG_QUADRANT	0010
STATE0SPLIT_CHANGE	1000
STATE2VERTICAL_CHANGE	1001
STATE2HORIZONTAL_CHANGE	1010
STATE4SPLIT_CHANGE	1011

Tabelul 2.12 – Informatiile provenite de la modulul CM

3. VERIFICARE

Designul realizat la pasul anterior trebuie sa fie functional la nivel de unitate si cluster. Astfel au fost realizate teste de baza pentru modulele simpliste si un environment de testare randomizata pentru modulele complexe si pentru cluster.

Astfel, modulele DB si FIFO au fost testate in cateva cazuri predefinite.

Pentru modulele UART, VGA, CM, LM si CD este nevoie de cate un mediu de verificare individual, care apoi poate fi inclus la nivelul de cluster. De specificat ca este necesara si testarea simplificata si a modulelor componente. Ordinea ideala de verificare este UART, CM si CD impreuna, dupa care LM si VGA separat, deoarece scopul proiectului face ca primele 3 module sa fie strans legate unul de celalalt. Testarea independenta este posibila (necesara in cazul protocolului UART), dar nu este necesara.

Scenariile si cazurile de test si coverage-ul au fost realizate pentru a se asigura functionarea robusta a proiectului.

Figura 3.1 – Design verificare modul UART individual

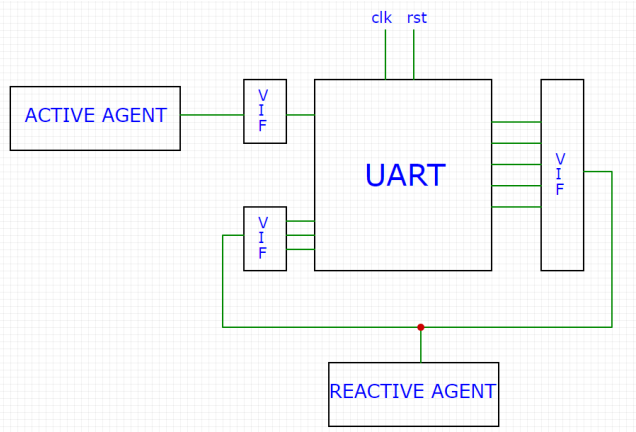
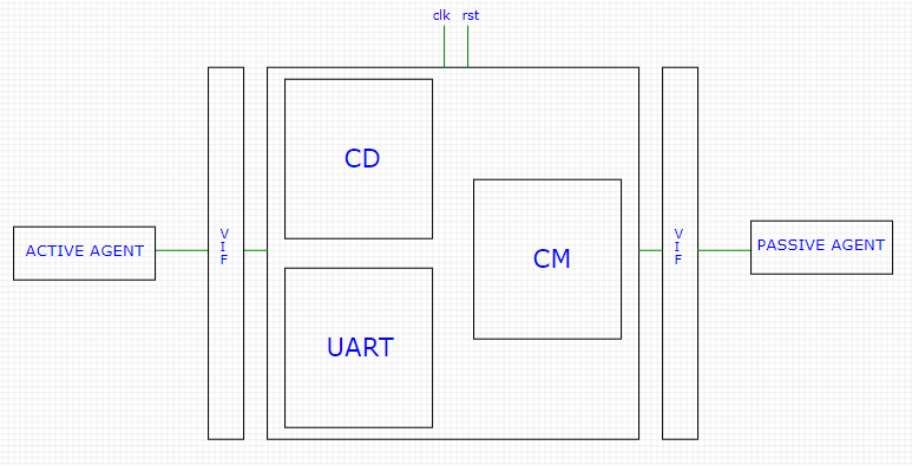


Figura 3.2 – Design verificare module UART, CM si CD impreuna



4. SIMULARE

*** pune poze din questasim

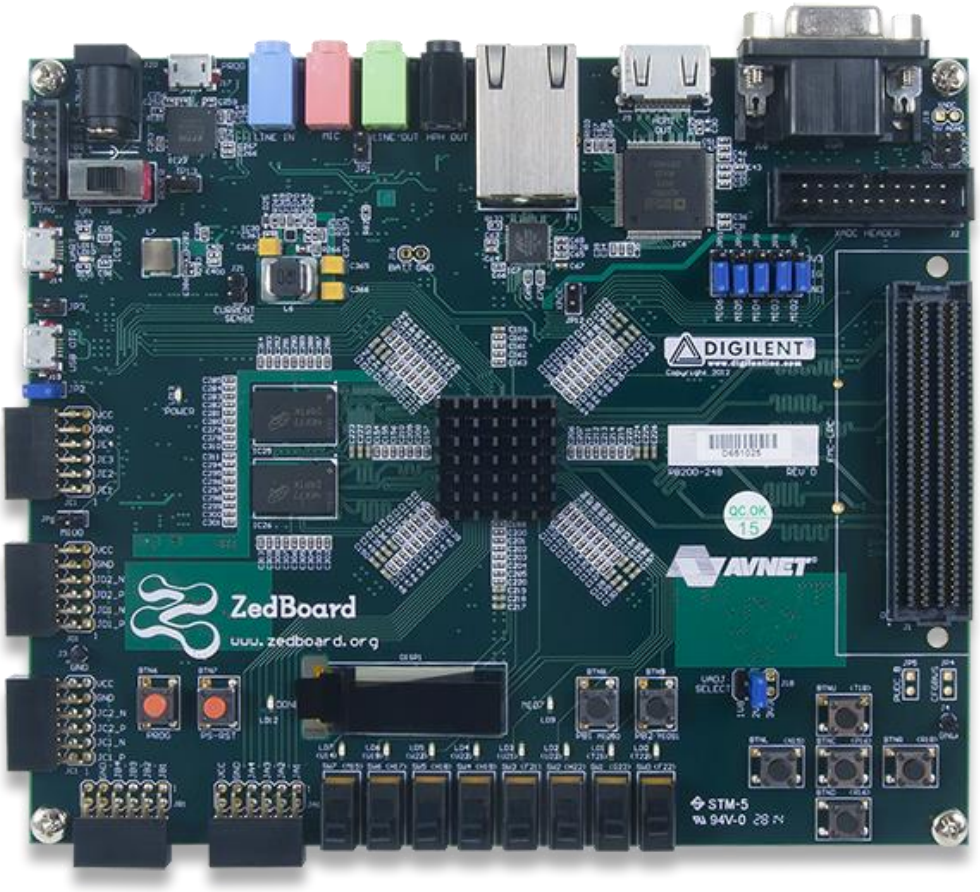
*** descrie cum compilezi/simulezi/rulezi/faci coverage/masori asertiile si scoreboardul

*** pune poze cu logurile pentru coverage si asertii

5. IMPLEMENTARE

Pentru implementare se va folosi placa de dezvoltare ZedBoard.

Figura 5.1 – ZedBoard [1]



Pentru a conecta intrarile si iesirile la pinii de pe placa am folosit un fisier de constrangeri:

```
#set_property PACKAGE_PIN T22 [get_ports {LD0}];  # "LED[0]"
#set_property PACKAGE_PIN T21 [get_ports {LD1}];  # "LED[1]"
#set_property PACKAGE_PIN U22 [get_ports {LD2}];  # "LED[2]"
#set_property PACKAGE_PIN U21 [get_ports {LD3}];  # "LED[3]"
#set_property PACKAGE_PIN V22 [get_ports {LD4}];  # "LED[4]"
#set_property PACKAGE_PIN W22 [get_ports {LD5}];  # "LED[5]"
#set_property PACKAGE_PIN U19 [get_ports {LD6}];  # "LED[6]"
#set_property PACKAGE_PIN U14 [get_ports {LD7}];  # "LED[7]"

#set_property PACKAGE_PIN Y21  [get_ports {VGA_B1}];  # "BLUE[0]"
#set_property PACKAGE_PIN Y20  [get_ports {VGA_B2}];  # "BLUE[1]"
#set_property PACKAGE_PIN AB20 [get_ports {VGA_B3}];  # "BLUE[2]"
#set_property PACKAGE_PIN AB19 [get_ports {VGA_B4}];  # "BLUE[3]"
#set_property PACKAGE_PIN AB22 [get_ports {VGA_G1}];  # "GREEN[0]"
#set_property PACKAGE_PIN AA22 [get_ports {VGA_G2}];  # "GREEN[1]"
#set_property PACKAGE_PIN AB21 [get_ports {VGA_G3}];  # "GREEN[2]"
#set_property PACKAGE_PIN AA21 [get_ports {VGA_G4}];  # "GREEN[3]"
#set_property PACKAGE_PIN AA19 [get_ports {VGA_HS}];  # "HS"
#set_property PACKAGE_PIN V20  [get_ports {VGA_R1}];  # "RED[0]"
#set_property PACKAGE_PIN U20  [get_ports {VGA_R2}];  # "RED[1]"
#set_property PACKAGE_PIN V19  [get_ports {VGA_R3}];  # "RED[2]"
#set_property PACKAGE_PIN V18  [get_ports {VGA_R4}];  # "RED[3]"
#set_property PACKAGE_PIN Y19  [get_ports {VGA_VS}];  # "VS"

#set_property PACKAGE_PIN F22 [get_ports {SW0}];  # "btnHS"
```

#set_property PACKAGE_PIN G22 [get_ports {SW1}]; # "btnVS"

#set_property PACKAGE_PIN H22 [get_ports {SW2}]; # "btnUART"

#set_property PACKAGE_PIN F21 [get_ports {SW3}]; # "btnVGA"

*** nu lasa aici codul

*** pune poze din vivado si din realitate

6. CONCLUZII

7. BIBLIOGRAFIE

1. https://digilent.com/reference/programmable-logic/zedboard/start
2. https://github.com/Digilent/digilent-xdc/blob/master/Zedboard-Master.xdc
3. https://web.mit.edu/6.111/www/s2004/NEWKIT/vga.shtml