



MODUL 6 PROYEK PERANCANGAN RANGKAIAN DIGITAL

Ardian Sudarsono (13217070)

Wilfrid Azariah (13217071)

Asisten: Yoland Nababan/13215053

Tanggal Percobaan: 30/11/2018

EL2102-Praktikum Sistem Digital

Laboratorium Dasar Teknik Elektro - Sekolah Teknik Elektro dan Informatika ITB



Abstrak

Pada praktikum kali ini, praktikan akan membuat sebuah proyek berupa sistem rangkaian digital yang dapat diimplementasikan ke dalam board FPGA. Pada praktikum ini, diharapkan praktikan dapat mendesain suatu sistem digital sederhana dan dapat mengimplementasikannya menggunakan interface pada board FPGA. Praktikum dilakukan dengan mengikuti modul praktikum sistem digital tahun 2018. Kesimpulan dari praktikum ini adalah sistem yang dibuat oleh praktikan telah berjalan dengan baik.

Kata kunci: Proyek, FPGA.

1. PENDAHULUAN

Salah satu tugas seorang insinyur elektro adalah mendesain sistem yang mengandung rangkaian logika kombinasional dan sekuensial. Proses mendesain ini dilakukan mulai dari merincikan spesifikasi sistem, membuat cara kerja sistem, dan lain-lain. Selanjutnya, insinyur akan membuat prototipe sistem tersebut dan menguji sistem tersebut. Pengujian dilakukan agar sistem dapat dievaluasi, diperbaiki, dan dikembangkan hingga menjadi suatu sistem yang baik.

Pada praktikum sebelumnya, praktikan telah mempelajari bagaimana cara kerja VGA Driver dan bagaimana cara menampilkan objek yang diinginkan pada layar LCD. Pada praktikum ini, praktikan akan membuat suatu proyek perancangan sistem digital sederhana yang dapat diimplementasikan menggunakan FPGA, dengan memanfaatkan segala ilmu desain yang praktikan peroleh sepanjang modul-modul praktikum. Tujuan dari proyek ini adalah agar praktikan dapat menspesifikasikan suatu sistem digital sederhana.

Proses desain yang termasuk dalamnya adalah membagi sistem menjadi jalur data dan sistem kendali dan menintegrasikan keduanya. Praktikan juga harus melakukan proses pengujian terhadap sistem dengan mengimplementasikannya pada FPGA berserta komponen tambahan yang diperlukan sistem.

Proyek ini dinilai mulai dari tahapan desain, implementasi, dan pengujian sistem. Praktikan membuat proyek mengikuti kriteria yang tertera pada modul, yaitu interaktif dengan menggunakan interface VGA, mempunyai bagian FSM, dan sedikitnya terdiri dari 3 blok. Kemudian, proyek akan dinilai berdasarkan fungsionalitas, kompleksitas, dan implementasi oleh asisten.

2. PROYEK YANG DIBUAT

Pada proyek ini, praktikan merancang sebuah game bernama "RUNNING BOX." Pengguna akan mengendalikan sebuah kotak yang disebut Kotak Player yang harus menghindari rintangan yang muncul. Kotak tersebut dapat menghindari rintangan dengan melompati rintangan. Pengguna dapat mengendalikan kecepatan lompat kotak untuk menyesuaikan dengan kecepatan gerak rintangan menuju kotak yang semakin lama akan semakin cepat.

Permainan diawali dengan kotak merah pada posisi kiri layar dan rintangan yang berwarna hijau berjalan mendekati kotak. Pengguna harus mengganti nilai switch reset menjadi '0' untuk memulai permainan. Ketika permainan dimulai, pengguna harus membuat kotak merah harus menghindari rintangan yang datang dengan menekan push button lompat. Ketika push button lompat ditekan, kotak akan menjadi lebih tinggi

seakan melompat, lalu kembali pada posisi awal seakan jatuh. Jika kotak mengenai rintangan, layar LCD akan menjadi warna merah tanda pengguna kalah. Pengguna harus mengganti nilai switch reset menjadi '1' untuk mereset game. Selama switch reset masih '1', tidak akan terjadi apa-apa ketika kotak mengenai rintangan. Kotak Player tidak dapat digerakan sampai reset diubah menjadi '0'. Tinggi rintangan berubah secara pseudorandom untuk menambah tantangan. Gerak rintangan juga semakin lama akan semakin cepat, tergantung dengan poin yang user dapatkan. Terdapat bar biru menandakan skor pada bagian atas layar LCD yang akan semakin panjang jika pengguna tidak kalah. Pengguna akan menang ketika bar skor mencapai ujung kanan layar.

Kontrol dari permainan ini adalah menggunakan tombol *push-button* (KEY[n]) serta *switch* (SW[n]) dengan fungsinya masing-masing. KEY[0] membuat kotak melompat, SW[0] sebagai input reset yang akan mereset permainan jika bernilai 1, dan SW[1] untuk mengatur kecepatan lompat kotak.

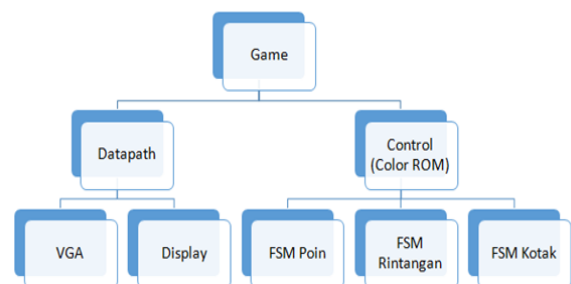
3. PROSES DESAIN SISTEM

3.1 METODOLOGI DESAIN

Metodologi desain sistem digital yang dibuat kali ini adalah menggunakan pendekatan *Bottom-Up*. Praktikan membuat terlebih dahulu system dasar dari system digital tersebut. lalu praktikan mulai mengembangkan desain sistem digital, melalui pembuatan modul dasar dari program, lalu dilanjutkan dengan mengembangkan modul dasar tersebut dengan menambah fitur-fitur pada program dasar.

Pada system digital yang praktikan buat, praktikan menggunakan seluruh file yang digunakan dalam praktikum modul sebelumnya, dengan mengembangkan program dasar kontrol pada modul `color_rom_vhd`. Program dasar kemudian di tes terlebih dahulu, untuk memastikan program dasar sudah sesuai dengan spesifikasi yang praktikan buat. Setelah melakukan pengembangan pada `color_rom_vhd`, praktikan memasukan port-port yang dibutuhkan pada `color_rom_vhd`, lalu melakukan pengembangan untuk fitur-fitur tambahan lainnya `Color_rom_vhd` digunakan untuk mengontrol visualisasi system pada board FPGA, serta

menjadi salah satu bagian dari data path. Namun dalam modul ini, control visualisasi system dan data path dipisah dalam 2 bagian yang berbeda, sehingga data path dan FSM sebagai bagian dari kontrol visualisasi system tidak bercampur. Modul VGA digunakan sebagai control *interface* dari VGA melalui sinyal warna dan sinyal sinkron. Modul Display digunakan sebagai bridge / jembatan untuk mengintegrasikan modul VGA dan modul `color_rom_vhd`.



3.2 METODOLOGI PENGUJIAN

Pengujian dilakukan dengan cara melakukan simulasi-simulasi tertentu untuk mengetes fitur-fitur yang ada dalam desain system digital. Praktikan membuat simulasi dengan cara memodifikasi konstanta-konstanta dalam program, melakukan simulasi untuk fitur tertentu, lalu mengembalikannya sesuai dengan konstanta awal.

Berikut simulasi yang praktikan buat

- Simulasi kalah :
Player tidak digerakan
Praktikan mengamati visual dari sistem
- Simulasi pergerakan rintangan :
Reset dinyalakan
Praktikan mengamati visual dari sistem
- Simulasi pergerakan player :
Variable pergerakan rintangan di set 0
Praktikan mengamati visual dari sistem
- Simulasi poin, dan percepatan pergerakan rintangan :
State loncat untuk player di non-aktifkan
Player diposisikan pada posisi dimana rintangan tidak akan mengenai player
Praktikan mengamati visual dari sistem

- Simulasi reset game
Player sengaja dibuat kalah, lalu reset dinyalakan.
Praktikan mengamati visual dari sistem
Reset dimatikan
Praktikan mengamati visual dari sistem

Dari hasil simulasi, praktikan akan mampu mendapatkan kesalahan-kesalahan pada sistem, dan memperbaikinya agar sesuai dengan spesifikasi yang praktikan buat. Praktikan memperbaiki sistem digital melalui modifikasi pada program control visual, yaitu `color_rom_vhd`

4. PENJELASAN ALGORITMA

Dalam perancangan sistem digital dalam proyek ini, praktikan menggunakan perancangan abstraksi dengan level *behavioral*, yaitu perancangan yang hanya memperhatikan nilai masukan dan keluaran dari desain yang dibuat, tanpa memperdulikan bentuk rangkaian itu sendiri

Berikut adalah algoritma yang dibuat untuk membuat permainan ini:

1. Modul Visual Controller

a. State Assignments

Pada algoritma ini, terdapat variable counter untuk rintangan serta player, dan *state* untuk menjalankan serta mengontrol permainan yaitu state kalah dan state reset. Algoritma ini memproses state-state yang ada sesuai dengan FSM.

b. Posisi

Algoritma ini akan memproses nilai nilai variable posisi dari setiap obyek pada game (rintangan dan player), serta menentukan apakah posisi mereka ada dalam kondisi state-state yang ada dalam FSM

c. Pixel Row and Column

Algoritma ini akan memproses nilai - nilai pixel dimana objek pada game di tampilkan. Pada nilai pixel tersebut akan ditampilkan warna sesuai kode warna masing masing objek. Untuk ukuran rintangan, tinggi rintangan akan berubah secara periodic dengan algoritma pseurandom

d. Kode Warna

Algoritma ini akan memproses warna dari objek yang akan ditampilkan. Kode warna sudah ditentukan terlebih dahulu, berbeda setiap objek, agar objek dapat terlihat dengan jelas

e. Pergerakan Objek

Algoritma ini akan memproses pergerakan masing-masing objek. Pergerakan player dan penambahan skor diproses dengan merubah batas-batas ukuran. Berikut aalgoritma tiap tiap objek pada proyek praktikan

Pergerakan rintangan diatur sehingga otomatis tanpa input dari user. Pergerakan rintangan dipercepat setiap player mendapatkan poin

Pergerakan player diatur dimulai saat ada input dari push button, lalu melakukan state 'loncat' yaitu bergerak dengan menaikan posisi player, lalu turun kembali ke posisi semula. Player tidak berubah posisi selama tidak mendapat input dari push button. Kecepatan player 'loncat' diatur dengan input dari switch SW1

Penambahan bar skor diatur otomatis sesuai poin yang didapat oleh user

2. Modul VGA

Pada modul ini, algoritma yang digunakan adalah algoritma untuk mengontrol *interface* yang ditampilkan pada LCD melalui VGA berupa sinyal warna dan sinyal sinkron. Ukuran piksel layar yang digunakan adalah 640 x 480 pixel persegi. Modul ini didapat dari modul praktikum sebelumnya.

3. Modul Display LCD

Pada modul ini, algoritma yang digunakan adalah algoritma untuk mengintegrasikan modul VGA dan modul control visual. Modul ini didapat dari praktikum sebelumnya.

5. CARA KERJA SISTEM

Cara kerja dari rangkaian yang dibuat adalah dengan menggunakan masukan berupa *push-button* pada *board* FPGA, kotak akan melompat jika push-button ditekan. Kecepatan lompatnya akan diatur menggunakan *switch*. Jika kalah, pengguna dapat menggunakan switch reset untuk mereset game. Dengan algoritma pengecil clock, pergerakan objek game dapat diamati pada LCD. Rintangan akan mendekati kotak dan kotak harus melompat untuk menghindari rintangan. Jika

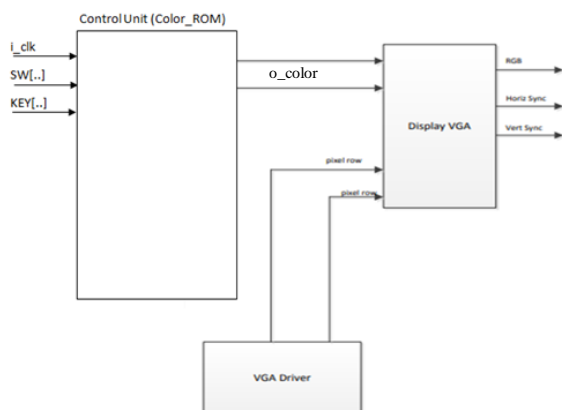
kotak terkena rintangan, layar akan menjadi merah pertanda game over. Pengguna harus mereset game untuk kembali bermain. Setiap kali pengguna sukses menghindari rintangan, bar poin akan bertambah pada bagian atas layar. Jika poin mencapai batas kanan layar maka permainan selesai dan layar akan merah pertanda game over. Semakin lama, pergerakan rintangan akan semakin cepat. Tinggi rintangan juga berubah secara pseudorandom.

6. DATAPATH DAN DIAGRAM FSM SISTEM

6.1. DATAPATH

Dapat dilihat pada gambar, blok *datapath* terdiri dari beberapa modul. Pada permainan ini, Modul Control visual praktikan masukan sebagai data path, sekaligus integrase dengan FSM, agar meghemat file dan mempercepat transmisi. Lalu data yang sudah diolah dikirim pada display VGA, lalu dikeluarkan dalam bentuk siny RGB, Horizontal dan Vertical sync

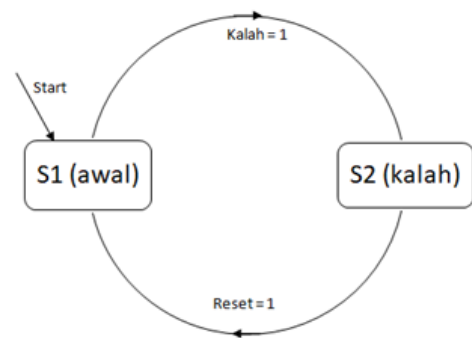
Berikut adalah desain *datapath* dari sistem digital yang praktikan desain.



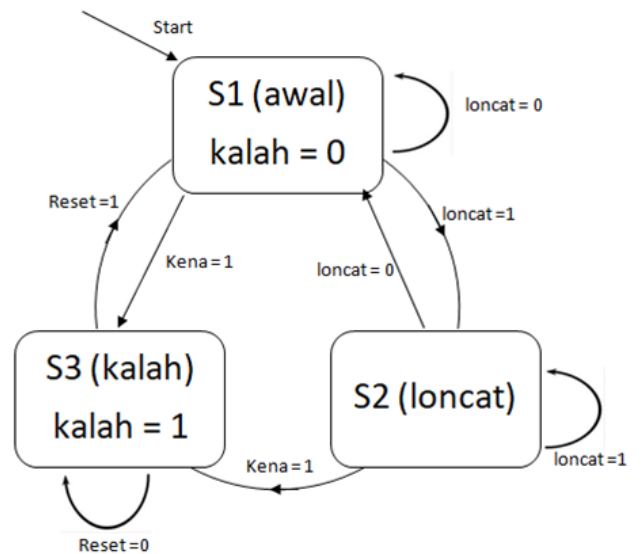
Gambar 6-1 Datapath Sistem

6.2. FSM

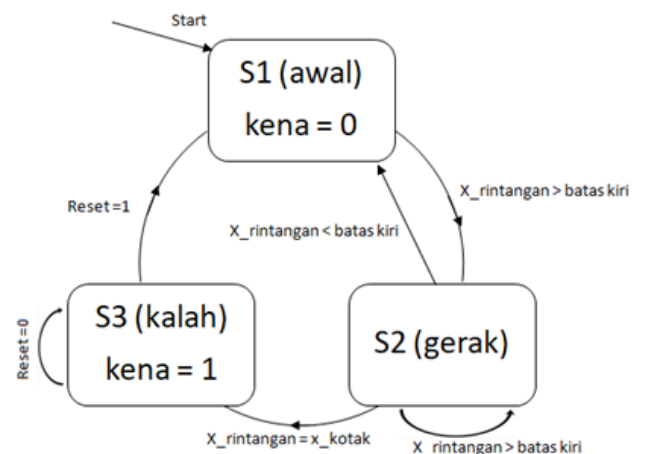
Berikut adalah desain FSM yang praktikan buat untuk sistem digital yang praktikan desain



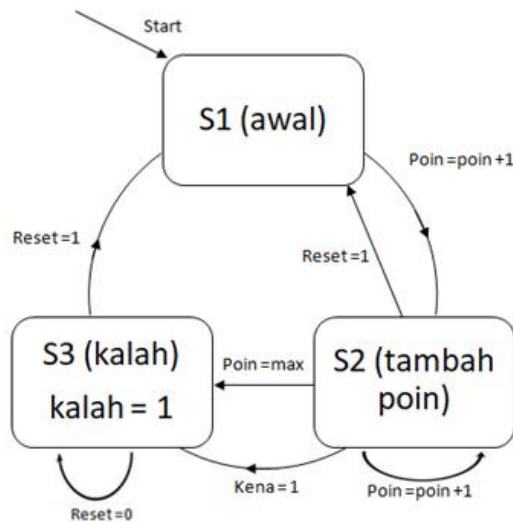
Gambar 6-3 Diagram FSM Game



Gambar 6-3 Diagram FSM Kotak



Gambar 6-4 Diagram FSM Rintangan



Gambar 6-5 Diagram FSM Poin

Deskripsi untuk masing-masing FSM dan *state* adalah sebagai berikut:

Pada FSM game, S1 adalah state awal dimana game dimulai. State ini hanyalah sebagai inisiasi karena selanjutnya, proses akan ditangani oleh subsistem, yaitu FSM Kotak, Rintangan, dan Poin. Ketika mendapat input 'kalah = 1', game akan berpindah state dari S1 ke S2. S2 adalah state kalah dimana semua proses akan terhenti dan game akan memunculkan screen kalah berwarna merah. Harus ada input reset dari user agar game berpindah kembali ke S1. Jika input reset = '0' game tidak akan berjalan, serta Player tidak bias digerakan

Pada FSM Kotak Player, S1 adalah state awal dimana game dimulai. Pada state ini, kotak akan diposisikan pada kiri layar. Ketika mendapat input 'loncat = 1', kotak akan berpindah state dari S1 ke S2. S2 adalah state loncat dimana posisi y dari kotak akan dinaikan selama 10 detik dan turun kembali selama 10 detik. Setelah proses 20 detik ini, state akan otomatis kembali pada S1. Ketika mendapat input 'kena=1' dari FSM rintangan, maka state akan berubah menjadi S3 atau state kalah dimana system berhenti. Ketika mendapat input 'reset' maka state akan kembali pada S1.

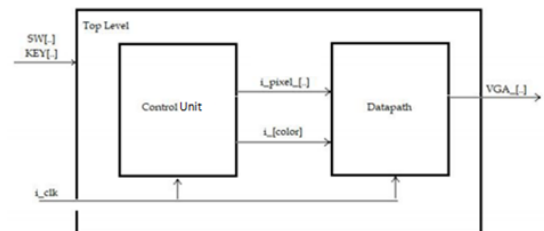
Pada FSM Rintangan, S1 adalah state awal dimana game dimulai. Pada state ini, rintangan akan diposisikan pada kanan layar. Secara otomatis, state akan berpindah state dari S1 ke S2. S2 adalah

state gerak dimana posisi x dari rintangan akan akan berkurang secara otomatis. Ketika posisi x dari rintangan sudah lebih kecil dari batas kiri screen, state akan otomatis kembali pada S1. Ketika $x_{kotak} = x_{rintangan}$, maka state akan berubah menjadi S3 atau state kalah dimana sistem berhenti memberikan output 'kena = 1'. Ketika mendapat input 'reset' maka state akan kembali pada S1.

Pada FSM Poin, S1 adalah state awal dimana game dimulai. Pada state ini, bar poin akan diposisikan pada bagian atas layar dengan panjang = 0. Ketika mendapat input 'poin = poin+1', kotak akan berpindah state dari S1 ke S2. Pada S2 ini, batas kanan bar poin akan bertambah sebanyak 10 piksel. Ketika mendapat input 'kena = 1' atau poin mencapai maks, state akan berpindah state ke S3. Pada S3 ini, sistem akan berhenti. Ketika mendapat input 'reset' maka state akan kembali pada S1.

6.3. TOP LEVEL

Berikut adalah desain *top level* yang memenuhi fungsi yang diinginkan.



Gambar diatas menunjukkan *top level* yang menggabungkan *datapath* dengan *control unit*. Control unit mendapat input, yaitu berupa SW[.] dan KEY[.]. Control unit berfungsi sebagai pengatur sistem *datapath*. Keluaran dari bagian ini adalah berupa *display* pada LCD.

7. HASIL PENGUJIAN SISTEM

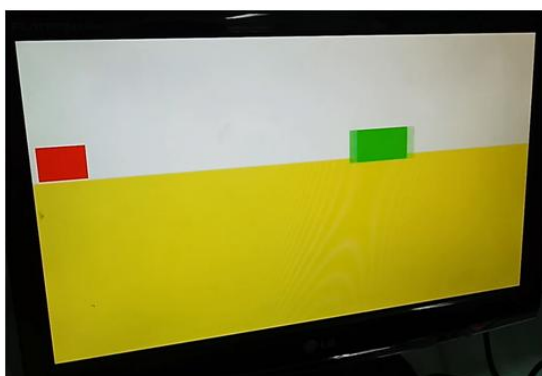
Gambar dibawah merupakan hasil pengujian untuk S1 atau state awal.



Gambar 7-1 Hasil Pengujian State S1

Pada gambar 7-1, terlihat bahwa kotak, rintangan, dan bar poin berada di posisi awal.

Gambar dibawah merupakan hasil pengujian untuk S2 pada rintangan atau state gerak.



(a)

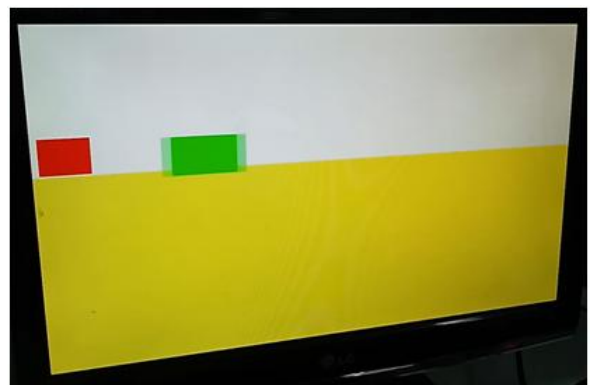


(b)

Gambar 7-2 Hasil Pengujian State Gerak Rintangan

Pada gambar 7-2(a) dan (b) terlihat perpindahan dari rintangan yang bergerak menuju kotak.

Gambar dibawah merupakan hasil pengujian untuk S2 kotak dan poin atau state loncat dan state tambah poin.



(a)



(b)

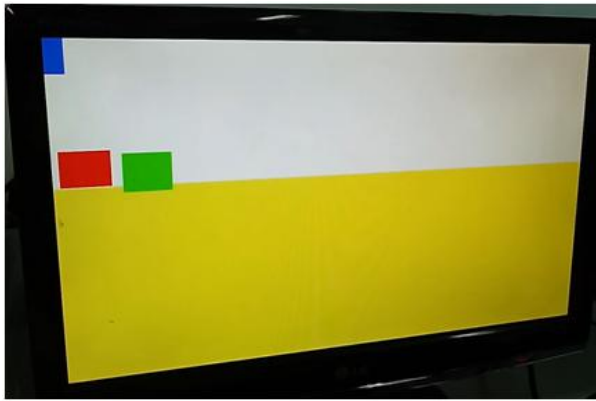


(c)

Gambar 7-3 Hasil Pengujian State S2 loncat dan tambah poin

Pada gambar 7-3(a), terlihat bahwa rintangan hendak mengenai kotak. Pada gambar 7-3(b), terlihat bahwa posisi kotak menjadi lebih tinggi akibat ada input dari push button. Pada gambar 7-3(c), terlihat bahwa kotak kembali lagi kepada S1 dan poin bertambah menjadi S2. Pada saat bersamaan, rintangan juga kembali pada posisi awal, yaitu S1.

Gambar dibawah merupakan hasil pengujian untuk S2 pada game atau S3 pada semua objek atau state kalah.



(a)



(b)

Gambar 7-4 Hasil Pengujian State Kalah

Pada gambar 7-4(a), terlihat bahwa rintangan hendak mengenai kotak dan pengguna tidak menekan push button untuk lompat. Alhasil, rintangan mengenai kotak dan game berubah menjadi S2 atau state kalah. Begitu state pada objek lainnya. Pada *state* ini, permainan selesai atau game over.

DAFTAR PUSTAKA

- [1] Hutabarat, T Mervin, *Petunjuk Praktikum: Praktikum Rangkaian Elektrik*, Laboratorium Dasar Teknik Elektro, Bandung, 2018

LAMPIRAN

Lampiran 1.

Modul Control Visual (color_rom_vhd.vhd)

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_ARITH.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5  use IEEE.NUMERIC_STD.ALL;
6
7  ENTITY color_rom_vhd IS
8  PORT(
9      clockfpga : IN STD_LOGIC;
10     RESET      : IN STD_LOGIC;
11     sp         : IN STD_LOGIC;
12     i_M_US     : IN STD_LOGIC;
13     i_K_US     : IN STD_LOGIC;
14     pushatas   : IN STD_LOGIC;
15     i_pixel_column : IN STD_LOGIC_VECTOR( 9 DOWNTO 0 );
16     i_pixel_row   : IN STD_LOGIC_VECTOR( 9 DOWNTO 0 );
17     o_red        : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 );
18     o_green      : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 );
19     o_blue       : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 );
20     o_poin       : OUT INTEGER;
21 END color_rom_vhd;
22
23 ARCHITECTURE behavioral OF color_rom_vhd IS
24     SIGNAL MUNCUL_KOTAK : STD_LOGIC;
25     SIGNAL MUNCUL_RINTANGAN : STD_LOGIC;
26     SIGNAL MUNCUL_ALAS : STD_LOGIC;
27     SIGNAL MUNCUL_KALAH : STD_LOGIC;
28     SIGNAL MUNCUL_POIN : STD_LOGIC;
29
30     PROCESS(clockfpga, i_pixel_row, i_pixel_column, MUNCUL_KOTAK,
31             MUNCUL_RINTANGAN, MUNCUL_ALAS, MUNCUL_KALAH, RESET)
32     --Kotak
33     VARIABLE ATAS_KOTAK : INTEGER := 149 ;
34     VARIABLE BAWAH_KOTAK : INTEGER := 199;
35     VARIABLE KIRI_KOTAK : INTEGER:= 10;
36     VARIABLE KANAN_KOTAK : INTEGER := 60;
37     --Rintangan
38     CONSTANT Patokan_Atas_Rintangan : INTEGER := 250;
39     VARIABLE ATAS_RINTANGAN : INTEGER := 190;
40     CONSTANT BAWAH_RINTANGAN : INTEGER := 210;
41     VARIABLE KIRI_RINTANGAN : INTEGER:= 500;
42     VARIABLE KANAN_RINTANGAN : INTEGER := 580;
43     --Alas
44     CONSTANT ATAS_ALAS : INTEGER := 200;
45     CONSTANT BAWAH_ALAS : INTEGER := 480;
46     VARIABLE KIRI_ALAS : INTEGER:= 0;
47     VARIABLE KANAN_ALAS : INTEGER := 640;
48     VARIABLE count : INTEGER := 0;
49     --Batas
50     CONSTANT kiri : INTEGER := 0;
51     CONSTANT kanan : INTEGER := 639;
52     CONSTANT atas : INTEGER := 0;
53     CONSTANT bawah : INTEGER := 480;
54     --Poin
55     VARIABLE ATAS_POIN : INTEGER := 1 ;
56     VARIABLE BAWAH_POIN : INTEGER := 50;
57     VARIABLE KIRI_POIN : INTEGER:= 1;
58     VARIABLE KANAN_POIN : INTEGER := 2;
59     -- Variabel loncat
60     variable speed : INTEGER := 1;
61     variable con : INTEGER := 0;
62     -- count adalah variabel penghitung clockfpga yang berlalu
63     -- max adalah konstanta nilai maksimum count dalam 1 detik
64     constant max : INTEGER := 4999999;
65     variable count : INTEGER := 0;
66     -- Variabel mengubah tinggi Rintangan
67     VARIABLE tinggiRintangan : INTEGER := 0;
68     VARIABLE conR : INTEGER :=0;
69     -- STATE
70     VARIABLE loncat : INTEGER := 0;
71     VARIABLE KALAH : INTEGER := 0;
```

FSM

```
73 BEGIN
74     IF (clockfpga'event and clockfpga = '1') THEN
75         -- FSM
76         -- Program mengecek input STATE RESET
77         IF RESET='1' THEN
78             -- Jika reset = 1, maka permainan tidak dimulai
79             -- Kotak Player tidak bisa bergerak
80             ATAS_KOTAK := 149;
81             BAWAH_KOTAK := 199;
82             KIRI_KOTAK := 10;
83             KANAN_KOTAK := 60;
84             conR:=0;
85         END IF;
86         -- mendelay pergerakan pada visual
87         IF count < max THEN
88             count := count + 1;
89             -- jika count belum mencapai max (count masih di interval 0 sampai
90             -- 9999999), count akan terus ditambah 1
91         ELSE
92             count := 0;
93             -- RINTANGAN
94             -- mengubah tinggi RINTANGAN
95             conR := conR + 1;
96             -- Mereset counter
97             if conR >20 then
98                 conR := 0 ;
99             -- Algoritma pseudorandom
100             elsif conR > 10 then
101                 ATAS_RINTANGAN:= ATAS_RINTANGAN- 3;
102             else
103                 ATAS_RINTANGAN:= ATAS_KOTAK + 5;
104             end if;
105             -- Pergerakan RINTANGAN
106             KIRI_RINTANGAN := KIRI_RINTANGAN - 10 - (KANAN_POIN/5);
107             KANAN_RINTANGAN := KANAN_RINTANGAN - 10 - (KANAN_POIN/5);
108
109             --Pergerakan kotak Player
110             IF pushatas = '0' THEN
111                 loncat := 1;
112             end if;
113             -- Menentukan faktor kecepatan LONCAT
114             if sp = '1' then
115                 speed := 15;
116             else
117                 speed := 10;
118             end if;
119             -- STATE LONCAT
120             if loncat = 1 then
121                 -- counter loncat
122                 con := con + 1;
123                 if con >20 then
124                     -- setelah counter habis, reset counter
125                     -- state loncat dikembalikan 0 secara otomatis
126                     con := 0 ;
127                     LONCAT := 0;
128                 elsif con > 10 then
129                     -- kotak naik
130                     ATAS_KOTAK := ATAS_KOTAK + speed;
131                     BAWAH_KOTAK := BAWAH_KOTAK + speed;
132                 else
133                     -- lalu turun
134                     ATAS_KOTAK := ATAS_KOTAK - speed;
135                     BAWAH_KOTAK := BAWAH_KOTAK - speed;
136                 end if;
137             END IF;
138
139             --BATAS KOTAK PLAYER
140             --Batas, jika melewati, akan tetap tertahan
141             IF BAWAH_KOTAK > bawah THEN
142                 ATAS_KOTAK := bawah-50;
143                 BAWAH_KOTAK := bawah;
144             END IF;
145             IF ATAS_KOTAK < atas THEN
146                 ATAS_KOTAK := atas;
147                 BAWAH_KOTAK := atas + 50;
148             END IF;
149
150             IF KANAN_KOTAK < kiri THEN
151                 ATAS_KOTAK := kiri;
152                 BAWAH_KOTAK := kiri + 50;
153             END IF;
154             IF KIRI_KOTAK > kanan THEN
155                 ATAS_KOTAK := kanan - 50;
156                 BAWAH_KOTAK := kanan ;
157             END IF;
158
159             -- BATAS RINTANGAN
160             -- batas tinggi rintangan, agar tinggi rintangan tidak
161             -- lebih tinggi dari tinggi maksimal kotak meloncat
162             IF ATAS_RINTANGAN < 140 THEN
163                 ATAS_RINTANGAN := 190;
164             END IF;
165             -- batas agar rintangan kembali pada posisi semula saat
166             -- sudah sampai pinggir kiri, lalu poin bertambah
167             IF KIRI_RINTANGAN < kiri THEN
168                 KIRI_RINTANGAN := kanan - 50;
169                 KANAN_RINTANGAN := kanan;
170                 KANAN_POIN:= KANAN_POIN + 10;
171             END IF;
172
173             -- Mekanisme Kena Rintangan
174             IF (KANAN_KOTAK>KIRI_RINTANGAN) AND (BAWAH_KOTAK>ATAS_RINTANGAN) THEN
175                 -- screen KALAH
176                 KANAN_KOTAK := kanan;
177                 BAWAH_KOTAK := bawah;
178                 KIRI_KOTAK := kiri;
179                 ATAS_KOTAK := atas;
180                 KANAN_POIN:= 2;
181                 KIRI_RINTANGAN:= 500;
182                 KANAN_RINTANGAN:= 580;
183             END IF;
184         END IF;
185         -- jika poin full, reset kembali ke awal game
186         IF KANAN_POIN=640 THEN
187             KANAN_KOTAK := kanan;
188             BAWAH_KOTAK := bawah;
189             KIRI_KOTAK := kiri;
190             ATAS_KOTAK := atas;
191             KANAN_POIN:= 2;
192             KIRI_RINTANGAN:= 500;
193             KANAN_RINTANGAN:= 580;
194         END IF;
```


DATA PATH

```

196 -- DATA PATH
197 -- pixelrow dan colomn
198 -- KOTAK PLAYER
199 IF ((i_pixel_row > ATAS_KOTAK) AND (i_pixel_row < BAWAH_KOTAK) )
200 AND ((i_pixel_column >= KIRI_KOTAK) AND (i_pixel_column < KANAN_KOTAK) ) THEN
201     MUNCUL_KOTAK <= '1';
202 ELSE MUNCUL_KOTAK <= '0';
203 END IF;
204 -- RINTANGAN
205 IF ((i_pixel_row > ATAS_RINTANGAN) AND (i_pixel_row < BAWAH_RINTANGAN))
206 AND ((i_pixel_column >= KIRI_RINTANGAN) AND (i_pixel_column < KANAN_RINTANGAN) ) THEN
207     MUNCUL_RINTANGAN <= '1';
208 ELSE MUNCUL_RINTANGAN <= '0';
209 END IF;
210 -- ALAS
211 IF ((i_pixel_row > ATAS_ALAS) AND (i_pixel_row < BAWAH_ALAS))
212 AND ((i_pixel_column >= KIRI_ALAS) AND (i_pixel_column < KANAN_ALAS) ) THEN
213     MUNCUL_ALAS <= '1';
214 ELSE MUNCUL_ALAS <= '0';
215 END IF;
216 -- POIN
217 IF ((i_pixel_row > ATAS_POIN) AND (i_pixel_row < BAWAH_POIN))
218 AND ((i_pixel_column >= KIRI_POIN) AND (i_pixel_column < KANAN_POIN) ) THEN
219     MUNCUL_POIN <= '1';
220 ELSE MUNCUL_POIN <= '0';
221 END IF;
222
223 -- KODE WARNA
224 -- KOTAK PLAYER
225 IF (MUNCUL_KOTAK = '1' ) THEN
226     o_red <= X"FF"; o_green <= X"00"; o_blue <= X"00";
227 -- RINTANGAN
228 ELSIF (MUNCUL_RINTANGAN = '1' ) THEN
229     o_red <= X"00"; o_green <= X"ff"; o_blue <= X"00";
230 -- ALAS
231 ELSIF (MUNCUL_ALAS = '1' ) THEN
232     o_red <= X"FF"; o_green <= X"ff"; o_blue <= X"00";
233 -- POIN
234 ELSIF (MUNCUL_POIN = '1' ) THEN
235     o_red <= X"00"; o_green <= X"00"; o_blue <= X"FF";
236 -- Selain itu, beri warna putih
237 ELSE o_red <= X"ff"; o_green <= X"ff"; o_blue <= X"ff";
238 END IF;
239 END IF;
240 END PROCESS;
241 END behavioral;

```

Lampiran 2 Modul Top Level

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_ARITH.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  ENTITY top_level_vhd IS
7  PORT(
8      CLOCK_50 : IN STD_LOGIC;
9      SW       : IN STD_LOGIC_VECTOR( 9 DOWNTO 0 );
10     sp       : IN STD_LOGIC;
11     RESET    : IN STD_LOGIC;
12     pushatas : IN STD_LOGIC;
13     VGA_R    : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
14     VGA_G    : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
15     VGA_B    : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
16     VGA_HS   : OUT STD_LOGIC;
17     VGA_VS   : OUT STD_LOGIC;
18     VGA_CLK  : OUT STD_LOGIC;
19     VGA_BLANK : OUT STD_LOGIC;
20     GPIO_0   : OUT STD_LOGIC_VECTOR( 35 DOWNTO 0 );
21     LEDR     : OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 );
22
23 END top_level_vhd;
24
25 ARCHITECTURE behavioral OF top_level_vhd IS
26 COMPONENT display_vhd IS
27 PORT(
28     i_clk      : IN STD_LOGIC;
29     i_M_US     : IN STD_LOGIC;
30     i_K_US     : IN STD_LOGIC;
31     RESET      : IN STD_LOGIC;
32     sp         : IN STD_LOGIC;
33     pushataso  : IN STD_LOGIC;
34     VGA_R      : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
35     VGA_G      : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
36     VGA_B      : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
37     VGA_HS     : OUT STD_LOGIC;
38     VGA_VS     : OUT STD_LOGIC;
39     VGA_CLK    : OUT STD_LOGIC;
40     VGA_BLANK  : OUT STD_LOGIC;
41     poin       : OUT INTEGER);
42 END COMPONENT;
43
44 BEGIN
45 module_vga : display_vhd
46 PORT MAP (
47     i_clk      => CLOCK_50,
48     i_M_US     => M_US,
49     i_K_US     => K_US,
50     sp         => sp,
51     RESET      => RESET,
52     pushataso  => pushatas,
53     VGA_R      => VGA_R,
54     VGA_G      => VGA_G,
55     VGA_B      => VGA_B,
56     VGA_HS     => VGA_HS,
57     VGA_VS     => VGA_VS,
58     VGA_CLK    => VGA_CLK,
59     VGA_BLANK  => VGA_BLANK,
60     poin       => poin
61 );
62
63 END behavioral;

```

Lampiran 3 Modul Display

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_ARITH.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  ENTITY display_vhd IS
7  PORT(
8      i_clk      : IN STD_LOGIC;
9      i_M_US     : IN STD_LOGIC;
10     i_K_US     : IN STD_LOGIC;
11     RESET      : IN STD_LOGIC;
12     sp         : IN STD_LOGIC;
13     pushataso  : IN STD_LOGIC;
14     VGA_R      : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
15     VGA_G      : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
16     VGA_B      : OUT STD_LOGIC_VECTOR( 5 DOWNTO 0 );
17     VGA_HS     : OUT STD_LOGIC;
18     VGA_VS     : OUT STD_LOGIC;
19     VGA_CLK    : OUT STD_LOGIC;
20     VGA_BLANK  : OUT STD_LOGIC;
21     poin       : OUT INTEGER);
22 END display_vhd;
23
24 ARCHITECTURE behavioral OF display_vhd IS
25
26 SIGNAL red      : STD_LOGIC_VECTOR( 5 DOWNTO 0 );
27 SIGNAL green    : STD_LOGIC_VECTOR( 5 DOWNTO 0 );
28 SIGNAL blue     : STD_LOGIC_VECTOR( 5 DOWNTO 0 );
29 SIGNAL red_color : STD_LOGIC_VECTOR( 7 DOWNTO 0 );
30 SIGNAL green_color : STD_LOGIC_VECTOR( 7 DOWNTO 0 );
31 SIGNAL blue_color : STD_LOGIC_VECTOR( 7 DOWNTO 0 );
32 SIGNAL pixel_row : STD_LOGIC_VECTOR( 9 DOWNTO 0 );
33 SIGNAL pixel_column : STD_LOGIC_VECTOR( 9 DOWNTO 0 );
34 SIGNAL red_on    : STD_LOGIC;
35 SIGNAL green_on  : STD_LOGIC;
36 SIGNAL blue_on   : STD_LOGIC;
37
38
39 COMPONENT vga IS
40 PORT(
41     i_clk      : IN STD_LOGIC;
42     i_red      : IN STD_LOGIC;
43     i_green    : IN STD_LOGIC;
44     i_blue     : IN STD_LOGIC;
45     o_red      : OUT STD_LOGIC;
46     o_green    : OUT STD_LOGIC;
47     o_blue     : OUT STD_LOGIC;
48     o_horiz_sync : OUT STD_LOGIC;
49     o_vert_sync : OUT STD_LOGIC;
50     o_pixel_row : OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 );
51     o_pixel_column : OUT STD_LOGIC_VECTOR( 9 DOWNTO 0 );
52 END COMPONENT;
53
54 COMPONENT color_rom_vhd IS
55 PORT(
56     clockfpga : IN STD_LOGIC;
57     sp        : IN STD_LOGIC;
58     i_M_US    : IN STD_LOGIC;
59     i_K_US    : IN STD_LOGIC;
60     RESET     : IN STD_LOGIC;
61     pushatas  : IN STD_LOGIC;
62     i_pixel_column : IN STD_LOGIC_VECTOR( 9 DOWNTO 0 );
63     i_pixel_row   : IN STD_LOGIC_VECTOR( 9 DOWNTO 0 );
64     o_red        : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 );
65     o_green      : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 );
66     o_blue       : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 );
67     o_poin       : OUT INTEGER);
68 END COMPONENT;

```

```

70 BEGIN
71
72 vga_driver0 : vga
73 = PORT MAP (
74     i_clk          => i_clk,
75     i_red           => '1',
76     i_green         => '1',
77     i_blue          => '1',
78     o_red           => red_on,
79     o_green         => green_on,
80     o_blue          => blue_on,
81     o_horiz_sync    => VGA_HS,
82     o_vert_sync     => VGA_VS,
83     o_pixel_row     => pixel_row,
84     o_pixel_column  => pixel_column);
85
86 color_rom0 : color_rom_vhd
87 = PORT MAP (
88     i_M_US          => i_M_US,
89     i_K_US          => i_K_US,
90     sp              => sp,
91     RESET           => RESET,
92     clockfpga       => i_clk,
93     pushatas        => pushataso,
94     i_pixel_column  => pixel_column,
95     i_pixel_row     => pixel_row,
96     o_red           => red_color,
97     o_green         => green_color,
98     o_blue          => blue_color,
99     o_poin          => poin);
100
101 red  <= red_color  (7 DOWNTO 2) ;
102 green <= green_color(7 DOWNTO 2) ;
103 blue <= blue_color (7 DOWNTO 2) ;
104


```

```

106 = PROCESS(red_on,green_on,blue_on,red,green,blue)
107 BEGIN
108
109 = IF (red_on = '1' ) THEN VGA_R <= red;
110 = ELSE VGA_R <= "000000";
111 END IF;
112
113 = IF (green_on = '1' ) THEN VGA_G <= green;
114 = ELSE VGA_G <= "000000";
115 END IF;
116
117 = IF (blue_on = '1' ) THEN VGA_B <= blue;
118 = ELSE VGA_B <= "000000";
119 END IF;
120
121
122 END PROCESS;
123
124
125
126 END behavioral;

```

Lampiran 4 Penilaian Rekan



Penilaian Rekan Praktikum

EL2102 Praktikum Sistem Digital

Rekan Penilai : Wafiq Anwar

NIM : 3119077

Judul Proyek Akhir : Penyusunan Rux

Rekan Dinilai : Arda Subarna

NIM : 3119090

Evaluasi Keterlibatan Rekan dalam Tim Kerja

- Keterlibatan dalam perancangan, implementasi, dan debugging proyek.**

30	Terlibat pada semua bagian perancangan, implementasi dan debugging proyek.	Nilai <u>30 /30</u>
25	Terlibat pada lebih dari satu bagian perancangan, implementasi atau debugging proyek.	
20	Terlibat sekurangnya pada perancangan atau implementasi atau debugging proyek.	
10	Tidak terlibat dalam pekerjaan tim.	
- Partisipasi dalam desain, demo dan presentasi.**

10	Memberikan kesempatan rekan berbicara untuk saling menguatkan/melengkapi.	Nilai <u>10 /10</u>
7	Memberi kesempatan rekan berbicara namun beberapa kali menginterupsi.	
4	Tidak memberi kesempatan rekan berbicara.	
- Partisipasi dalam pengerjaan proyek.**

10	Sering hadir (diatas 80% kehadiran) dan terlibat dalam diskusi.	Nilai <u>10 /10</u>
7	Kadang-kadang tidak hadir (50-80% kehadiran).	
4	Jarang hadir dalam diskusi (kurang dari 50%) kehadiran.	
- Sikap dalam pengerjaan proyek.**

10	Menata kembali semua peralatan serta merapikan area kerja dan melaporkan kepada penanggung jawab Lab.Dasar setelah selesai melakukan percobaan.	Nilai <u>10 /10</u>
7	Cukup terlibat dalam merapikan area kerja.	
4	Tidak terlibat dalam penataan dan merapikan area kerja.	
- Kesiapan untuk menyelesaikan proyek akhir.**

10	Terlibat dalam pengerjaan laporan akhir dan pembuatan video	Nilai <u>10 /10</u>
7	Terlibat pada salah satu pekerjaan (laporan atau video)	
4	Tidak terlibat pada pembuatan laporan dan video	

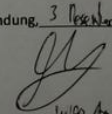
Jumlah Total	<u>70 /70</u>
--------------	---------------

Komentar untuk rekan praktikum :

Sangat Bro!

Saya menyatakan bahwa penilaian rekan praktikum ini diisi dengan sebenarnya.

Bandung, 3 Desember 2018



Wafiq Anwar