↑ TOP of text, page 2 and following pages,aligned to line↑

12345678901234567890123456789012345678901234
2

IMPLEMENTASI NIOS II SOFT-PROCESSOR untuk decoding FILE IMAGE BMP

1
2

Stefanus Erick Sebastian[1)], Susilo Wibowo[2)]

1) Jurusan Teknik Elektro,Universitas Surabaya, Surabaya 60293, email: stefanus_erick@yahoo.com
2) Jurusan Teknik Elektro,Universitas Surabaya, Surabaya 60293, email: susilo_w@ubaya.ac.id

Abstrak

Pada makalah ini membahas implementasi NIOS II soft-processor pada FPGA untuk decoding file image BMP. File BMP yang tersimpan di SD Card dibaca dan di-decode oleh NIOS II kemudian ditampilkan pada TFT (Thin Film Transistor) LCD (Liquid Crystal Display).

Tahapan pengerjaan yang dilakukan adalah membuat kontroler SD Card, membaca file sistem atau dikenal dengan istilah FAT(File Allocation Table), membuat BMP decoder, mengirim data warna tiap pixel ke TFT-LCD, dan mendeteksi penekanan tombol next dan back. Implementasi yang dikerjakan berhasil mendeteksi file BMP yang tersimpan pada root direktori SD Card ber-FAT16, kemudian menampilkannya pada sebuah TFT LCD.

***Kata kunci**: NIOS II, BMP, FAT, soft-processor.*

1. Pendahuluan

Pada tahun 1980 dikembangkan rangkaian terintegrasi yang dinamakan FPGA (*Field Programmable Gate Arrays*), yaitu perangkat semikonduktor yang dapat dikonfigurasikan oleh konsumen atau desainer setelah diproduksi. IC ini terdiri dari banyak komponen logika terprogram yang disebut *logic block*, koneksi antar *logic block* dapat dikonfigurasi menjadi komponen yang dikenal dengan istilah *core*. Skala desain suatu *core* dapat berupa komponen yang mempunyai fungsi tertentu seperti *core* untuk komunikasi serial hingga komponen yang kompleks seperti prosesor.

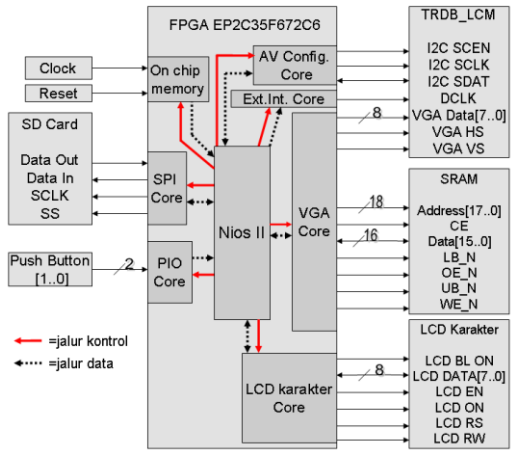
NIOS II merupakan *soft-processor* buatan Altera. Pada desain sistem yang dibahas di makalah ini, digunakan FPGA CycloneII EP2C35F672C6 yang terdapat pada DE2 *Board*, suatu modul pengembangan produksi Altera. Selain FPGA tersebut, modul ini juga dilengkapi banyak perangkat, di antaranya yang digunakan adalah SRAM 512K, sakelar *push button*, SD Card *socket*, karakter LCD 16x2, *oscillator* 50 MHz, dan GPIO (*General Purpose Input Output*) *port*.

NIOS II *soft-processor* digunakan untuk membaca isi dari SD *Card* berkapasitas 1 GB dengan struktur FAT16, kemudian mendeteksi file gambar dengan format BMP yang ber-resolusi maksimum 320x240 dengan jumlah bit per pixel adalah 24 atau 32 bit, menampilkannya pada TFT-LCD, dan *switch* gambar tersebut sesuai penekanan tombol *push button next* atau *back* [6]. Semua IP(*Intellectual Property*) *core* yang digunakan pada desain diperoleh dari Altera. Adapun TFT LCD yang digunakan buatan Terasic, dinamakan TRDB_LCM.

2. Desain

Diagram blok dari sistem yang didesain dapat dilihat pada Gambar 1. Terlihat bahwa ada beberapa *core* dibuat sebagai *peripheral* NIOS II yang dikonfigurasikan di dalam FPGA EP2C35F672C6, tiap *core* menangani tugasnya masing-masing yaitu:

1. SPI (*Serial Peripheral Interface*) *core* sebagai protokol komunikasi untuk akses SD *Card*.
2. PIO (*Parallel Input Output*) *core* sebagai *interface push button next* dan *back*.
3. AV (*Audio Video*) *Configuration core* sebagai protokol komunikasi untuk konfigurasi TFT LCD.
4. *External Interface core* sebagai penghasil sinyal *clock* untuk transfer data ke TFT LCD.
5. VGA (*Video Graphics Array*) *core* sebagai kontrol proses pengiriman data ke TFT LCD serta akses SRAM sebagai memori *buffer*.
6. LCD karakter *core* sebagai kontroller dari karakter LCD untuk menampilkan nama file.
7. NIOS II sebagai *soft-processor* dari sistem untuk kontrol semua *core*.
8. *On chip memory* sebagai tempat penyimpanan instruksi yang akan dieksekusi oleh NIOS II.



Gambar 1. Blok diagram sistem

Gambar 2. Hasil akhir sistem

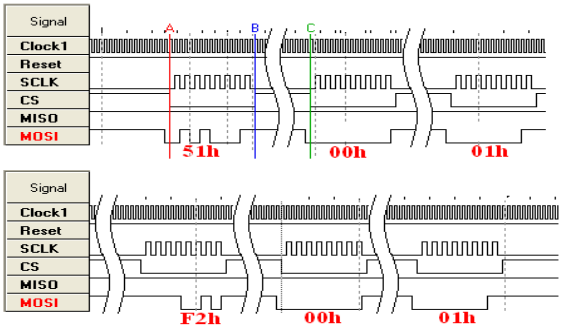
Hasil akhir dari sistem tersebut ditampilkan pada Gambar 2. Terlihat bahwa logo Teknik Elektro Ubaya yang tersimpan di SD *Card* ditampilkan pada TFT_LCD.

2.1. Desain kontroler SD *Card*

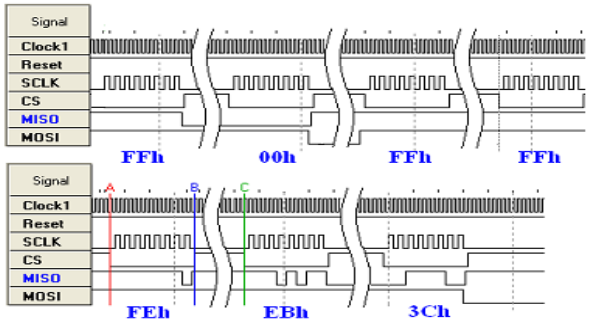
SPI *core* digunakan sebagai *peripheral* antara NIOS II dengan SD *Card*. Data dibaca NIOS II pada *register* rxdata dari SPI core, sedangkan data dikirim ke SD *Card* melalui *register* txdata[1]. *Core* ini bertindak sebagai *master*, parameter di dalamnya dikonfigurasikan sebagai berikut: besar frekuensi SCLK adalah 25MHz, lebar data sebesar 8 bit dengan arah *shifting* MSB terlebih dulu, dan *timing clock polarity* maupun *clock phase* berlogika *low*. Konfigurasi ini sesuai dengan spesifikasi SD *Card*.

Pada awalnya dilakukan inisialisasi sehingga SD *Card* aktif pada mode SPI. CMD0 dikirim untuk memulai proses inisialisasi, kemudian dikirim CMD1 secara berulang hingga diperoleh respon yang menyatakan SD *Card* telah memasuki mode SPI dan siap menerima perintah lainnya seperti baca data [5].

Perintah CMD17 digunakan untuk membaca isi dari SD *Card* sebesar 1 blok atau sektor (512 byte). Sinyal pengiriman perintah tersebut dapat dilihat pada Gambar 3. Kode 51h merupakan kode untuk perintah CMD17, kemudian 00 01 F2 00h adalah alamat sektor 249 (dinyatakan dalam byte) dari SD *Card* yang datanya akan dibaca. Terakhir kode 01h menyatakan akhir dari pengiriman perintah, kemudian dilanjutkan respon dari SD *Card*.



Gambar 3. Sinyal pengiriman CMD17



Gambar 4. Sinyal respon CMD17

Sinyal respon dari perintah tersebut dapat dilihat pada Gambar 4. Kode FFh yang pertama menyatakan NCR (*Command Response Time*), kemudian respon 00h menyatakan bahwa perintah berhasil diterima. Setelah itu SD *Card* mengirim FFh secara terus menerus selama menyiapkan pengiriman data pada sektor 249. Respon FEh menyatakan data siap dikirim, dilanjutkan EB 3Ch yang merupakan 2 byte pertama dari sektor 249. Pengiriman dilanjutkan untuk 510 byte selanjutnya.

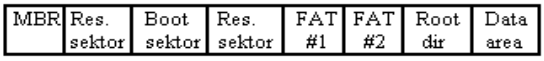
Berikut instruksi yang dijalankan NIOS II untuk membaca satu blok data pada alamat address, kemudian data disimpan di read_sect:

```
void read_single_block(alt_u32 address, alt_u8 * read_sect)
{
alt_u8 addr[4]; //membuat array addr dengan besar masing-masing  8 bit
alt_u8 baca_data[850];//membuat array baca_data dengan besar masing-masing  8 bit
int i=0;//membuat variabel i dengan tipe integer
for(i=0;i<4;i++)
addr[i]= address>>(24-8*i); //memecah data address(32bit) menjadi 4 data yang tersimpan di array addr(8 bit)
```

```
alt_u8 cmd17[6]= {0x51,addr[0],addr[1],addr[2],addr[3],
0x01}; //perintah pada SD card untuk read single blok
alt_avalon_spi_command (SPI_SD_CARD_BASE, 0, 6,
cmd17,850, baca_data,0x02);  //menggunakan fungsi
alt_avalon_spi_command untuk transfer data dengan SD Card
i=0;
while(baca_data[i]!=0xFE) i++;          //pointer baca_data
ditambahkan sampai mendapatkan data 0xFE (respon yang
mnytakan  data siap dibaca)
int start_read=i+1;// data pertama yang akan dibaca
for(i=0;i<512;i++)
*read_sect++ =baca_data[start_read+i]; //baca data sebesar
512 byte atau 1 blok, disimpan pada read_sect
}
```

2.2. Struktur FAT16

Untuk mengetahui alamat daerah penyimpanan file maka perlu diperhatikan struktur FAT. Pada FAT16, urutan strukturnya dapat dilihat pada Gambar 5.



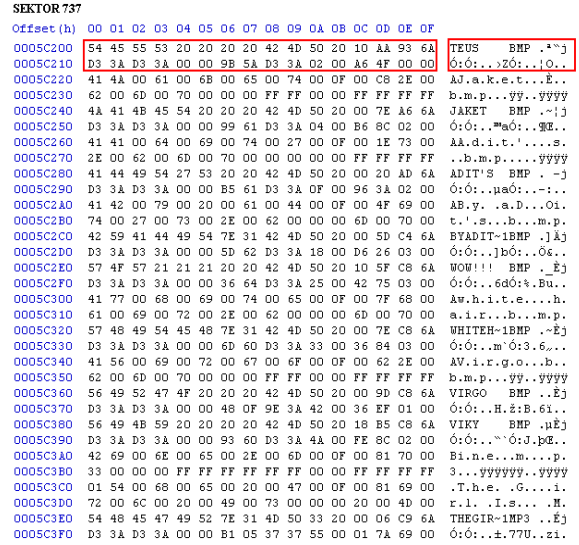
Gambar 5. Urutan struktur FAT16

Tiap bagian menyimpan informasi sebagai berikut:

- MBR = Informasi partisi dan alamat *boot* sektor.
- *boot* sektor = Informasi FAT16, digunakan untuk mengetahui alamat FAT *table* dan *root directory*.
- FAT *table* = Informasi penggunaan *cluster*.
- *root directory* = Informasi umum file/ folder.
- data area = Area penyimpanan data.

Format penyimpanan menggunakan *little endian*, atau MSB disimpan pada alamat yang lebih tinggi [3].

Untuk tujuan deteksi file BMP maka pembahasan dikonsentrasikan pada bagian *root directory*, ditunjukkan pada Gambar 6.



Gambar 6. Kode heksa *root directory*

Tabel 1. Arti kode heksa di *root directory* untuk file TEUS.BMP

Offset	Deskripsi	Ukuran	Nilai
00h	Nama file	8 byte	"TEUS"
08h	Ekstension	3 byte	"BMP"
1Ah	Cluster awal	1 word	00 02h

Pada Gambar 6 diketahui bahwa informasi tiap file maupun folder pada *root directory* sebesar 32 bytes atau kelipatannya untuk LFN (*Long File Names*). Pada Tabel 1 menunjukkan arti kode heksa yang terdapat di dalam kotak. Berdasarkan tabel tersebut, diketahui bahwa file BMP dapat dideteksi melalui *ekstension*nya yang tersimpan di alamat offset 08h, dengan alamat melalui perhitungan:

Alamat sektor dari file=((cluster awal-1)*32)+737 (1)

Setelah didapatkan alamat dari file BMP maka tahap dilanjutkan dengan membaca struktur dari BMP tersebut di data area.

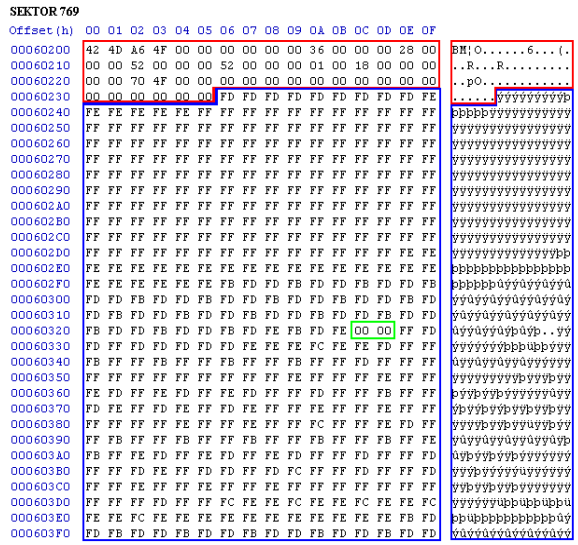
Berikut instruksi yang dijalankan NIOS II untuk mendeteksi file BMP:

```
for(i=0;i<=0x1F;i++)header_file[i]=root_dir[j+i];//copy
informasi file di root direktori
rootfile=rootdirfunct(header_file);//memilah informasi ke tiap
bagian
if
((rootfile.Extension==0x424D50)&(rootfile.Filename[0]!=0xE
5)) //ekstension file BMP=42 4D 50, byte pertama pd filename
=0xE5 menandakan file telah dihapus
{
show_filename(lcd_char,rootfile.Filename);//cetak nama file
//tampilkan gambar ke TRDB_LCM
result= read_BMP_file(RGBstruct,rootfile.Startcluster);
if (result==1)   //jika output dari fungsi read_BMP_file =1 atau
ukuran gambar lebih besar dari 320 x 240
{
blank(vga);//layar diputihkan
alt_up_character_lcd_set_cursor_pos(lcd_char, 0, 2);//kursor di
baris 2
alt_up_character_lcd_write(lcd_char, overpix, 12);//tampilkan
tulisan "size too big"
}
}
else result=2;//jika belum menemukan file BMP,proses
pencarian dilanjutkan
```

2.3. Struktur file BMP

File TEUS.BMP mempunyai struktur yang ditunjukkan Gambar 7. Kode heksa di dalam kotak atas pada gambar tersebut merupakan *header* dari file TEUS.BMP yang berukuran 54 byte. Sisa byte di dalam

kotak bawah merupakan data warna dengan urutan Blue, Green, Red.



Gambar 7. Kode heksa file TEUS.BMP di data area

Tabel 2. Arti kode heksa dari *header* file TEUS.BMP

Offset	Deskripsi	Ukuran	Nilai
12h	Lebar (kolom)	1Dword	52h=82
16h	Tinggi (baris)	1Dword	52h=82
1Ch	Jumlah bit per pixel	1 word	18h=24

Beberapa informasi yang digunakan pada desain sistem ditunjukkan Tabel 2. Ukuran maksimum untuk lebar dan tinggi dari gambar yang ditampilkan ke TFT LCD adalah 320 dan 240, sedangkan syarat jumlah bit per *pixel* adalah 24 atau 32 bit. *Pixel* disimpan dengan urutan per baris untuk kolom 1 hingga kolom 82, dilanjutkan baris ke 2, untuk kolom 1 hingga kolom 82, begitu seterusnya. Baris 1 dimulai dari baris paling bawah dan kolom 1 dimulai dari kolom paling kiri. *End Of file* berada pada pixel baris ke 82 dan kolom ke 82.

Terdapat kotak kecil berisi 00 00h yang merupakan *byte-padding* dari file BMP. Salah satu karakteristik file ini adalah jumlah byte tiap barisnya merupakan kelipatan 4 byte. Jika tidak memenuhi syarat tersebut, akan ditambahi *byte-padding* pada tiap akhir baris [4].

2.4. Pengiriman data ke TFT-LCD

TFT LCD diinisialisasi terlebih dahulu. Proses inisialisasi ini dijalankan secara otomatis oleh AV *config core* tiap terjadi reset [1]. Nilai register setelah diinisialisasi ditunjukkan Tabel 3. Penggunaan register dan isinya dapat dilihat di [2]

Tabel 3. Nilai register TRDB_LCM setelah inisialisasi

Alamat register	Nilai
0x02	00h
0x03	01h
0x04	3Fh
0x05	17h
0x06	18h
0x07	08h
0x08	00h
0x09	20h
0x0A	20h
0x0B	20h
0x0C	10h
0x10	3Fh
0x11	3Fh
0x12	2Fh
0x13	2Fh
0x14	98h
0x15	9Ah
0x16	A9h
0x17	99h
0x18	08h

Setelah proses inisialisasi, maka TFT LCD mampu menerima data yang dikirim NIOS II melalui VGA *core*. Data RGB tiap *pixel* yang ditulis ke VGA *core*, akan disimpan di SRAM. Kemudian, VGA *core* membaca data warna dari SRAM untuk ditampilkan ke TFT-LCD. Adanya memori *buffer* ini dikarenakan TFT LCD membutuhkan *refresh*.

Selama proses pengiriman data, TFT LCD membutuhkan *clock* sebesar 25 MHz yang dihasilkan oleh *External Interface Core*. Sinyal *Vertical Synch* dan *Horizontal Synch* yang dihasilkan VGA *core* secara otomatis disesuaikan dengan spesifikasi TFT LCD. Input warna pada VGA *core* berukuran 16 bit RGB per *pixel* [1] dengan komposisi ditunjukkan Gambar 8. NIOS II bertugas dalam proses menjalankan instruksi agar tampilan gambar sesuai dengan yang tersimpan di SD *Card* seperti komposisi warna, posisi peletakan *pixel*, serta konversi jumlah bit per *pixel*.

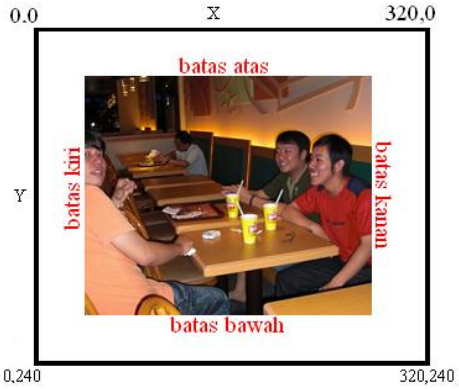
Gambar 8. Komposisi 16 bit RGB pada VGA *core*

Berikut instruksi untuk konversi 24 bit RGB ke 16 bit RGB:

```
readRGBstruct read_BGR (readRGBstruct RGBstruct,alt_u8
*data_BMP, alt_u16 bitcount)
{
```

```
RGBstruct.B=data_BMP[RGBstruct.offset]>>3;//konversi komponen blue dari 8 bit  ke 5 bit
RGBstruct.G=data_BMP[RGBstruct.offset]>>2;//konversi komponen Green dari 8 bit ke 6 bit
RGBstruct.R=data_BMP[RGBstruct.offset]>>3;//konversi komponen Res dari 8 bit  ke 5 bit
RGB=(RGBstruct.R<<11)+(RGBstruct.G<<5)+(RGBstruct.B);//menggabungkan tiap komponen R,G,B
}
```

Apabila gambar yang ditampilkan mempunyai resolusi di bawah 320 x 240, gambar ditampilkan di tengah layar TFT LCD, seperti terlihat pada Gambar 9.



Gambar 9. Tampilan gambar di TFT LCD

Pada Gambar 9 ditunjukkan koordinat x,y (kolom, baris) untuk akses *pixel* di TFT LCD yang telah ditentukan oleh VGA *core*. Proses pengiriman data warna per *pixel* sesuai dengan urutan file BMP yaitu dimulai dari (0,240) hingga (320,240), dilanjutkan baris di atasnya yaitu (0,239) hingga (320, 239), begitu seterusnya hingga baris paling atas yaitu (0,0) hingga (320,0).

Agar tampilan gambar di tengah maka ditentukan batas-batas dari gambar tersebut, melalui perhitungan:

$$\text{Batas atas} = \frac{240 - tinggi}{2} \quad (2)$$

$$\text{Batas bawah} = \text{tinggi} + \text{batas atas} \quad (3)$$

$$\text{Batas kiri} = \frac{320 - lebar}{2} \quad (4)$$

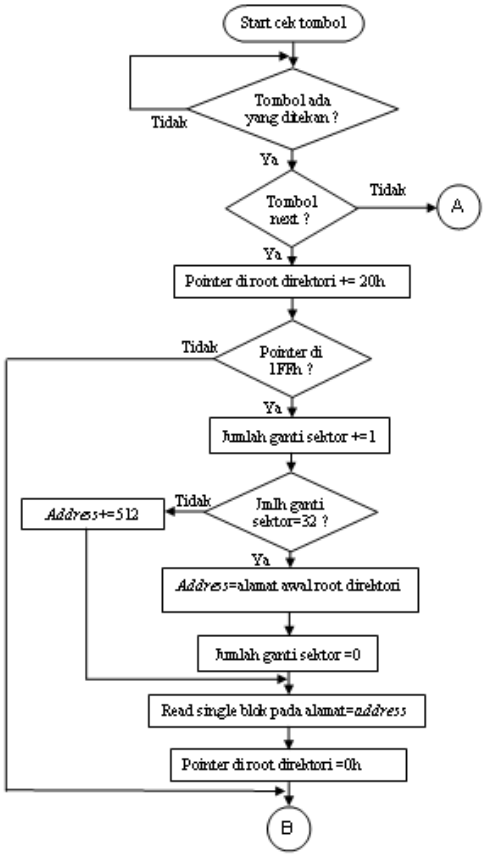
$$\text{Batas kanan} = \text{lebar} + \text{batas kiri} \quad (5)$$

NIOS II mengirim warna putih (FF FF h) ke VGA *core* pada koordinat di luar batas-batas tersebut. Setelah satu gambar telah ditampilkan, NIOS II akan mengecek adanya penekanan tombol *next* atau *back*.

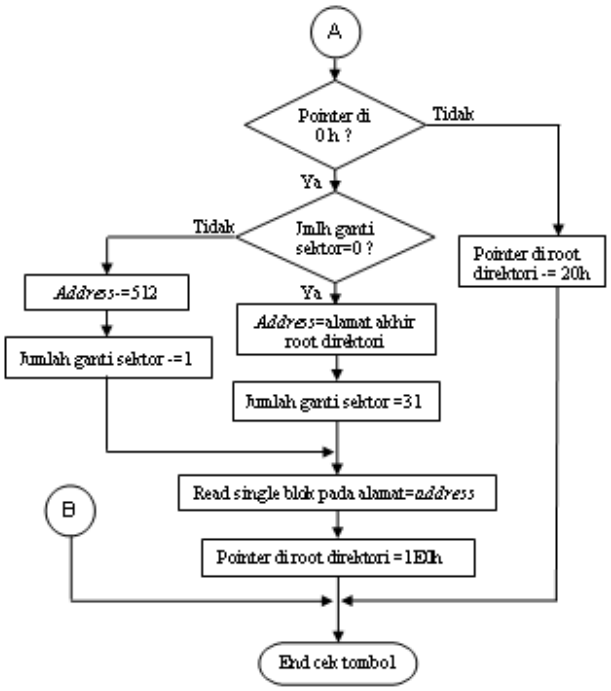
2.5. Desain *switch* gambar *next* dan *back*

Diagram alir untuk proses pengecekan tombol dan proses yang terjadi jika tombol *next* ditekan ditunjukkan Gambar 10. Jika Tombol *next* ditekan maka variabel yang digunakan sebagai *pointer* di *root directory* ditambah dengan 20h. Kemudian dicek apakah saat ini *pointer* berada di 1FFh (511) atau posisi terakhir dari satu sektor. Jika tidak, maka program untuk bagian ini selesai. Sedangkan jika ya, maka variabel yang menyimpan jumlah pergantian sektor ditambah dengan 1.

Root directory FAT16 terbatas hanya sampai 32 sektor (0 sampai 31), maka dilakukan pengecekan apakah saat ini sudah melebihi batas tersebut. Jika tidak, maka variabel address berisi alamat sektor berikutnya, yaitu dengan menambahkan 512 (1sektor=512 byte). Jika ya, berarti address diisi alamat awal *root directory*. NIOS II membaca data di sektor dengan alamat yang disimpan di variabel address. Kemudian pointer direset menjadi 0.



Gambar 10. Diagram alir jika tombol *next* ditekan



Gambar 11. Diagram alir jika tombol *back* ditekan

Diagram alir untuk proses yang terjadi jika tombol *back* ditekan ditunjukkan Gambar 11. Jika tombol *back* ditekan maka dilakukan pengecekan terlebih dahulu, apakah *pointer* berada di 0h. Jika tidak, maka *pointer* dikurangi dengan 20h dan program untuk bagian ini selesai. Sedangkan jika ya, berarti *pointer* harus pindah sektor maka dilakukan pengecekan lagi apakah sektor sekarang adalah alamat awal *root directory*.

Andaikata sektor sekarang bukan sektor pertama *root directory*, maka variabel address dikurangi dengan 512 berarti pindah ke sektor sebelumnya, dan variabel jumlah ganti sektor dikurangi dengan 1. Sedangkan jika ya, maka address harus diisi alamat akhir dari *root directory*, dan variabel jumlah ganti sektor diisi 31. Pada dua kemungkinan tersebut, diakhiri proses pembacaan data pada sektor dengan alamat yang disimpan di variabel address. Mengingat tombol *back* yang ditekan, maka *pointer* sekarang berada di 1E0h, atau mengacu pada file terakhir dari sektor.

3. **Kesimpulan**

Dari percobaan dan pengujian yang telah dilakukan, dapat disimpulkan bahwa:

1. Implementasi NIOS II beserta IP *core* berhasil menampilkan file BMP di SD *Card* pada sebuah TFT-LCD.
2. SD *Card* dalam mode SPI harus diinisialisasi terlebih dahulu. Kemudian file yang tersimpan di SD *Card* dapat dibaca dengan mengirim perintah CMD17.
3. Pada FAT16 terdiri dari MBR, *boot* sektor, tabel FAT, *root directory*, dan data area. Proses deteksi file BMP dan lokasi penyimpanan file dapat diketahui dari *root directory*.
4. *Decoder* file BMP dilakukan dengan membaca data RGB yang tersimpan di area *bitmap* dengan *pixel* pertama terletak di kiri bawah dari gambar.

Referensi

[1] Altera Corporation. (2008,November). IP Core. Altera Corp., San Jose. [Online]. Tersedia: ftp://ftp.altera.com/up/pub/University_Program_IP_Cores/UP_IP_Library.exe.

[2] Datasheet TPG051 RGB Driver/Timing Controller IC For LTPS TFT LCD

[3] Jack Dobiash (2008, November 10). FAT16 Structure Information.[Online]. Tersedia: http://home.teleport.com/~brainy/fat16.htm

[4] Paul Debono(2008, Desember 12).BMP FILE HANDLING.[Online]. Tersedia: //www.um.edu.mt/__data/assets/pdf_file/0019/53263/Ing._Paul_Debono_1.pdf

[5] SanDisk Corporation. (2003,Desember). San Disk Secure Digital Card : Product Manual. SanDisk Corp., Sunnyvale.[Online].Tersedia: http://www.cs.ucr.edu/~amitra/sdcard/ProdManualSDCardv1.9.pdf

[6] S. E. Sebastian, "Pengaplikasian Altera DE2 Board sebagai Decoder File Image BMP yang Tersimpan di SD Card", Tugas Akhir, Jurusan Teknik Elektro, Universitas Surabaya, 2009.