Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2023106993
Vol. 10, No. 4, Agustus 2023, hlm. 799-806 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

RAM-ROM SEBAGAI PENDUKUNG ALGORITMA *ZIGZAG SCAN* MENGGUNAKAN METODE PEMETAAN PADA KOMPRESI CITRA *REAL-TIME*

Robby Candra[*1]

[1]Universitas Gunadarma, Jakarta
Email: [1]robb13.c7@gmail.com
[*]Penulis Korespondensi

(Naskah masuk: 24 Januari 2023, diterima untuk diterbitkan: 25 Juli 2023)

Abstrak

Konsep pengiriman informasi yang meliputi berbagai macam format data dengan proses yang cepat (*real-time*) semakin dibutuhkan untuk berbagai kebutuhan, hal ini harus didukung dengan infrastruktur yang memadai baik saluran komunikasi, perangkat keras maupun aplikasi yang digunakan. Guna mendukung konsep *real-time* diperlukan 2 komponen *Random Access Memory* (RAM) yang dapat menyimpan data sehingga tidak terdapat antrian data. Hal ini bertujuan agar proses scan dapat berlangsung dengan cepat karena proses penulisan dan pembacaan koefisien *Discrete Cosine Transform* (DCT) dapat dilakukan secara bersamaan di kedua RAM tersebut. Proses yang dilakukan yaitu pada siklus clock yang sama RAM 1 melakukan proses penulisan dan RAM 2 melakukan proses pembacaan, pada siklus clock berikutnya RAM 1 melakukan proses pembacaan dan RAM 2 melakukan proses penulisan, sehingga RAM 1 dan RAM 2 dapat bekeja secara paralel, yaitu melakukan proses penulisan dan pembacaan secara bersamaan agar proses *scan* koefisien DCT dapat berlangsung dengan cepat. Hasil perancangan prototipe IC RAM dan ROM menggunakan FPGA dapat diimplementasikan untuk proses kompresi citra secara *real-time.*

Kata kunci: *Citra, Kompresi, RAM, ROM, Zigzag Scan*

RAM-ROM AS SUPPORT FOR ZIGZAG SCAN ALGORITHM USING MAPPING METHOD IN REAL-TIME IMAGE COMPRESSION

Abstract

The concept of sending information which includes various data formats with fast (real-time) processing is increasingly needed for various needs, this must be supported by adequate infrastructure, both communication channels, hardware and applications used. In order to support the real-time concept, 2 components of Random Access Memory (RAM) are needed which can store data so that there is no data queue. It is intended that the scanning process can take place quickly because the process of writing and reading the Discrete Cosine Transform (DCT) coefficients can be done simultaneously in both RAMs. The process that is carried out is that in the same clock cycle RAM 1 carries out the writing process and RAM 2 performs the reading process, in the next clock cycle RAM 1 carries out the reading process and RAM 2 carries out the writing process, so RAM 1 and RAM 2 can work in parallel, i.e. the process of writing and reading simultaneously so that the DCT coefficient scan process can take place quickly. The results of the IC RAM and ROM prototype design using FPGA can be implemented for real-time image compression process.

Keywords: *Compression, Image, RAM, ROM, Zigzag Scan*

1. PENDAHULUAN

Di era teknologi informasi yang berkembang pengiriman dan penerimaan data sudah menjadi bagian dari kebutuhan sehari-hari seperti data yang dalam berbagai macam format yang disebut sebagai data multimedia. Data multimedia adalah suatu kombinasi data atau media untuk menyampaikan suatu informasi sehingga informasi itu tersaji dengan lebih menarik seperti audio, video, teks, grafik dan animasi. Konsep pengiriman informasi yang meliputi berbagai macam format data dengan proses yang cepat (*real-time*) semakin dibutuhkan untuk berbagai kebutuhan, hal ini harus didukung dengan infrastruktur yang memadai baik saluran komunikasi, perangkat keras maupun aplikasi yang digunakan (Budiarto dan Qudsi, 2018). Hingga saat ini pengiriman informasi secara *real-time* masih mengalami kendala, karena masih terdapat *delay time* yang relatif besar hal ini disebabkan besarnya

kapasitas data yang harus dikirim melampaui kecepatan transmisi yang dimiliki oleh perangkat keras yang ada (Madenda et al, 2010) (Rustamaji et al, 2015). Waktu tunda (*Latency*) pada konsep *real-time* merupakan bagian yang sangat penting karena waktu tanggap yang cepat akan mempengaruhi hasil akhir seperti yang dikemukakan pada penelitian yang dilakukan oleh Hartono (Hartono dan Trismiyati, 2014). Hal ini terkait dengan pengurutan koefisien hasil *Discrete Cosine Transform* (DCT)-terkuantisasi pada algoritma *zigzag scan*. Pengurutan DCT ini sangat berkontribusi pada rasio kompresi citra seperti yang dikemukanan pada penelitian yang dilakukan oleh Wang (Wang dan Yu, 2011).

Metode kompresi citra dapat dilakukan menggunakan *hardware* seperti ASIC maupun FPGA. Secara *hardware* kompresi citra digunakan pada piranti-piranti yang berhubungan dengan citra digital seperti kamera digital, *scanner*, perekam video digital, *set-top-box*, sampai dengan aplikasi video *conferencing* (Kusuma, 2010). Implementasi algoritma kompresi citra pada rangkaian digital menggunakan *Application Specific Integated Circuit* (ASIC) atau *Field Programmable Gate Array* (FPGA) banyak diterapkan pada berbagai piranti karena kinerjanya lebih baik dilihat dari segi waktu kompresi (Agostini et al, 2007) dan dapat diimplementasikan untuk proses komputasi yang memerlukan kecepatan dan presisi tinggi (Irfan dan Setiawan, 2022). Penggunaan FPGA dalam kompresi citra menjadi teknik yang populer dalam hal efisiensi (Awcock dan Thomas, 1995).

Penelitian yang dilakukan oleh Prakash (Prakash dan Gurumurthy, 2012) setiap 64 koefisien DCT yang diperoleh untuk setiap 8 x 8 blok piksel, 64 koefisien DCT tersebut harus tersedia sebelum proses scan untuk menyimpan koefisien DCT tersebut yang masuk secara serial ke dalam memori sementara. Jika pada satu waktu dilakukan pembacaan atau dilakukan penyimpanan koefisien DCT yang masuk, hal ini memperlambat proses *scan*. Pada penelitian yang dilakukan oleh Pratomo (Pratomo et al, 2020), menjelaskan terdapat antrian jika proses proses baru yang datang tapi proses lama belum selesai hal ini waktu pemrosesan lebih lama dan dapat menyebabkan kegagalan pemrosesan citra.

Berdasarkan hasil penelitian yang telah dikembangkan pada penelitian diatas, dapat disampaikan bahwa untuk kebutuhan analisis citra secara *real time* belum memberikan hasil yang optimal baik dari sisi waktu eksekusi maupun dari sisi penggunaan sumber daya pada perangkat FPGA. Aplikasi *real-time* pengolahan tekstur kumpulan citra dari video yang setiap detiknya terdiri dari 25 frame (PAL format) atau 30 frame (NTSC format). Waktu keberadaan (tc) setiap citra dalam video adalah 1/25 atau 1/30 detik, artinya semua proses pengolahan citra tidak boleh melebihi waktu tc (Kardian et al, 2018).

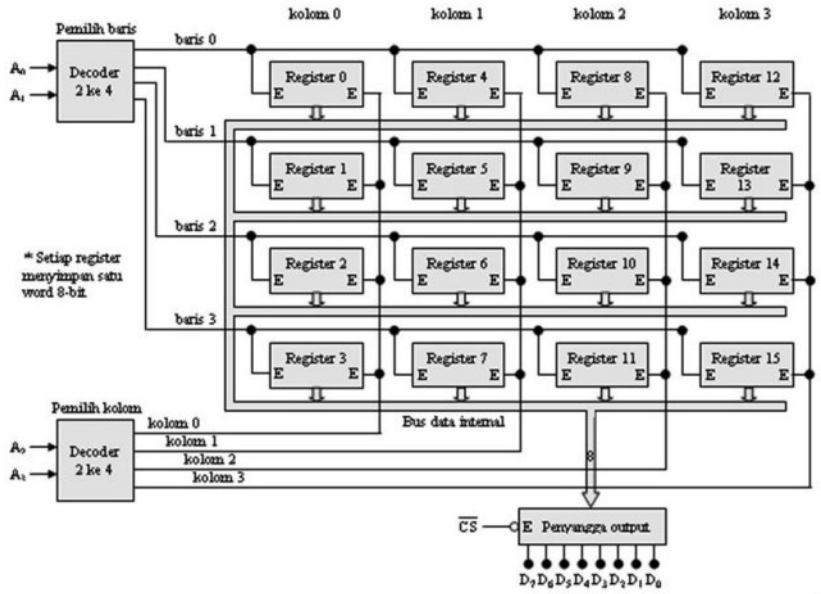
Guna mendukung proses penyimpanan koefisien DCT tersebut diperlukan komponen *Random Access Memory* (RAM) dan *Read Only Memory* (ROM). Untuk mengatasi masalah yang masih terdapat pada penelitian Vijaya dan membuat scanning lebih cepat, diusulkan rancangan arsitektur baru untuk *zigzag scan* yang menggunakan dua buah memori RAM, dengan menggunakan RAM informasi dapat dimanipulasi sebelum ditampilkan keluarannya (Saptono et al, 2013). Tujuannya dari penelitian ini yaitu untuk menghasikan algoritma yang lebih efisien dari sisi waktu proses karena proses dapat dilakukan secara paralel pada RAM 1 dan RAM 2 dan menghasikan sebuah deskripsi komponen elektronik berupa IP *core* untuk proses *scan* data masukan.

2. METODE PENELITIAN

Tahapan penelitian yang dilakukan yaitu pada tahap pertama melakukan studi literatur, pada tahap ini melakukan studi terhadap sejumlah artikel dan buku yang menguraikan tentang penggunaan RAM dan ROM pada algoritma kompresi citra dan video, arsitektur FPGA dan implementasi RAM dan ROM pada IC-FPGA. Tahap kedua implementasi penggunaan RAM dan ROM pada algoritma kompresi citra menggunakan metode pemetaan ke dalam IC-FPGA agar hasil kompresi citra dapat diperoleh secara *real-time*. Tahap ketiga yaitu tahapan pengujian hasil implementasi dan analisis hasil implementasi.

ROM dan RAM merupakan sebuah memori. Pada ROM hanya dapat dilihat informasi yang tersimpan di dalamnya, dengan terlebih dahulu memberikan masukan alamat ke ROM. Menurut alamat ini, informasi yang disimpan pada alamat di ROM dapat dilihat. Pada RAM, dapat menyimpan output ke memori dan membaca informasi yang tersimpan dalam memori (Naik et al, 2015). ROM merupakan jenis memori semikonduktor yang dibuat untuk menyimpan data permanen. Struktur dari ROM sangat komplek, namun ada empat bagian pokok yang membangun ROM diantaranya adalah register *array*, *decoder* baris (*row decoder*), *decoder* kolom (*column decoder*) dan *output buffer* (Zwolinski, 2004). Arsitektur dari ROM ini ditunjukan pada Gambar 1.

Metode desain ROM menggunakan kode VHDL dapat diimplementasikan penyimpanan datanya dalam beberapa cara, yaitu model *signal*, *constant*, dan *package*. Nilai dari ROM didefinisikan sekali, dalam VHDL pendefinisian ROM menggunakan konstanta *array* seperti dalam *syntax*



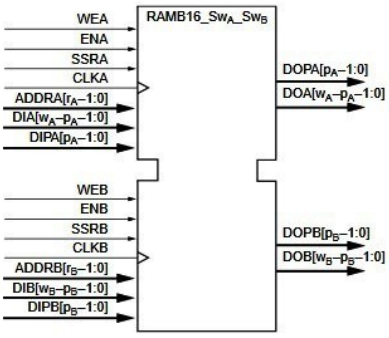
Gambar 1. Arsitektur ROM (Zwolinski, 2004)

```
type <romtype> i s ar ray (0 to
<rom_width>)
o f s td_logi c_ve c tor (<rom_addr>
downto 0 ) ;
cons tant <rom_name> : <romtype>:=(
        1=>"00000001" ,
        2=>"00000010" ,
        3=>"00000110" ,
        4=>"00000111" ,
        5=>"00001111");
```

Sebuah RAM dapat dimodelkan dalam banyak cara. Karena data dapat disimpan dan dibaca dalam RAM, maka sinyal data dinyatakan sebagai mode *InOut*. Selain itu, tiga sinyal kontrol disediakan. Pertama, CS (*Chip Select*), adalah sinyal kontrol umum untuk mengaktifkan chip RAM tertentu. Dua sinyal lainnya adalah OE (*Output Enable*) dan WE (*Write Enable*). Hanya satu dari dua sinyal ini harus diaktifkan pada satu waktu. Ketika Data dari RAM maka sinyal OE yang aktif, sebaliknya jika data ditulis ke RAM maka sinyal WE yang aktif (Zwolinski, 2004). Dalam VHDL pendefinisian ROM dapat menggunakan konstanta array seperti dalam syntax berikut:

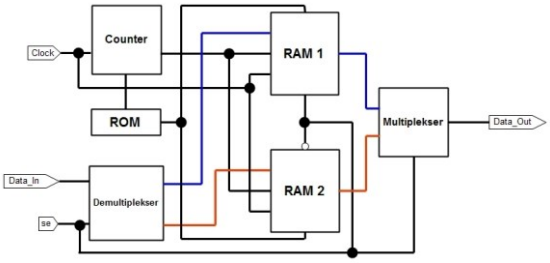
```
type <ramtype> i s ar ray (0 to
<ram_width>)
o f s td_logi c_ve c tor (<ram_addr>
downto 0 ) ;
s i g n a l <ram_name> : <ramtype> := ( o
the r s=> ( o the r s =>'0 '));
```

Secara fisik, blok RAM memiliki dua akses port yang berdiri sendiri, yaitu Port A dan Port B. Struktur dari 2 port ini sama, untuk pertukaran data dan mendukung proses menulis dan membaca. Diagram dari struktur RAM ini seperti yang ditunjukan pada Gambar 2.



Gambar 2. Struktur RAM (Zwolinski, 2004)

Pada penelitian ini rancangan terdiri *counter*, demultiplekser, ROM, 2 buah RAM dan multiplekser seperti yang ditunjukan pada Gambar 3. Memori RAM pertama menyimpan data koefisien DCT secara serial dan memori RAM kedua melakukan proses pembacaan, sehingga kedua RAM tersebut dapat bekerja secara bersaman/paralel. Dengan menggunakan konsep paralelisme dapat mempersingkat waktu proses komputasi (Pertiwi et al, 2014). Pada periode *clock* pertama menyiapkan 64 koefisien DCT dari 8 x 8 blok yang merupakan data masukan pertama. Koefisien DCT yang masuk disimpan secara serial dilakukan bersamaan dengan proses pembacaan *zigzag*.



Gambar 3. Rancangan pengurutan *zigzag scan* yang menggunakan dua buah memori RAM

Proses penulisan dan pembacaan data pada RAM 1 dan RAM 2 urutannya dibangkitkan terlebih dahulu oleh *Counter*. *Counter* juga berfungsi untuk membangkitkan urutan pembacaan posisi pemetaan yang tersimpan dalam ROM. Kerja dari Counter ini secara paralel untuk RAM 1, RAM 2 dan ROM. Hasil pembacaan data *input* pada RAM terhadap posisi pemetaan pada ROM merupakan hasil *zigzag scan* dengan pemetaan.

RAM 1 dan RAM 2 melakukan proses penulisan dan pembacaan secara bergantian. Bagian yang berperan dalam proses kerja dalam melakukan proses penulisan pada RAM 1 dan RAM 2 secara bergantian ini yaitu demultiplekser dengan sinyal kontrol yang tepat sinyal *input* dapat diarahkan ke salah satu jalur *output* sehingga demultiplekser ini berfungsi sebagai data *distributors* (Tocci dan Widmer, 2001). Sedangkan bagian yang berperan dalam proses pembacaan secara bergantian antara RAM 1 dan RAM 2 yaitu multiplekser yang dapat disebut juga sebagai *selector* (Sugiartowo dan Ambo, 2018).

3. HASIL DAN PEMBAHASAN

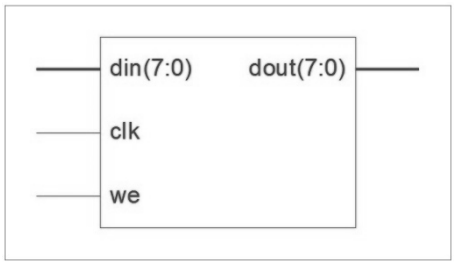
Data input yang menjadi masukan dalam pemrograman menggunakan VHDL seperti yang ditunjukan pada tabel program berikut :

Program:

```
l i b r a r y IEEE;
use IEEE.STD_LOGIC_1164 . a l l ;
use i e e e . s td_logi c_ar i th . a l l ;
use i e e e . s td_logic_uns igned . a l l ;
e n t i t y RAM i s
por t (
clk , we : in s td_l o g i c ;
din : in s td_logi c_ve c tor (7 downto 0 ) ;
dout : out s td_logi c_ve c tor (7 downto 0 )
) ;
end RAM;
a r c h i t e c t u r e Behavioral o f RAM i s
type memory_t i s ar ray (1 to 64) o f s
td_logi c_ve c tor (7 downto 0 ) ;
s i g n a l ram_tmp : memory_t:= ( o the r
s=> ( o the r s =>'0 '));
s i g n a l i : i n t e g e r range 1 to 64:=1;
begin
pr o c e s s ( clk , we )
begin
i f ( r i s ing_edge ( c l k ) ) then
        i f we='1 ' then
                i f ( i <65) then
                        ram_tmp ( i ) <=
                        din ;
                        dout <=
                        ram_tmp( i ) ;
                        i<=i +1;
                end i f ;
                i <=1;
        end i f ;
        end i f ;
        end pr o c e s s ;
end Behavioral ;
```

Berdasarkan program di atas, data input terdiri dari 64 urutan data input yang masing-masing data terdiri dari 8 bit *vector*. Data input ini disimpan dalam rangkaian RAM. Untuk membaca data secara berututan mulai dari data ke 1 sampai dengan data ke 64 digunakan *counter* sebanyak 64 kali terdiri dari 6 bit *vector* data alamat. Data alamat ini digunakan untuk menempatkan data *input* sesuai dengan urutan yang dibangkitkan oleh *counter*. Output dari RAM yang terdiri dari 8 bit *vector* menjadi masukan untuk D *Flip-flop* yang berfungsi sebagai *register*. Output dari D flip-flop ini merupakan hasil proses pembacaan data input.

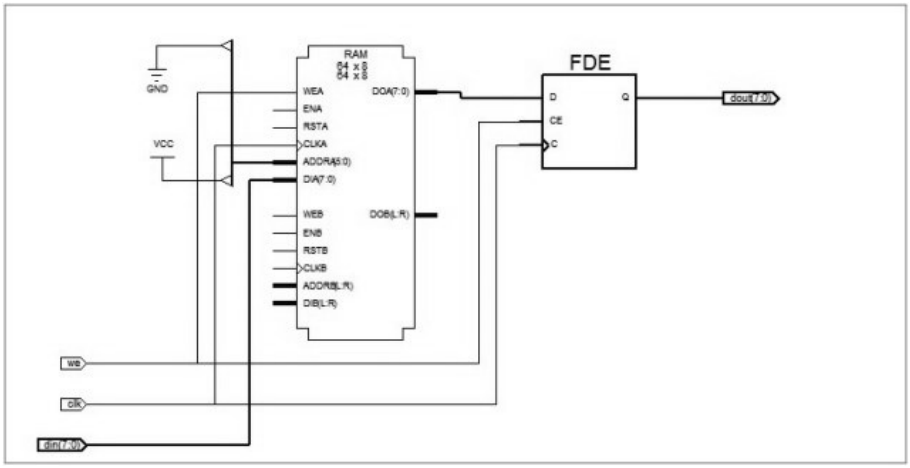
Hasil yang diperoleh yaitu berupa sintesis *register transfer logic* (RTL) yang diterjemahkan menjadi *netlist* yaitu daftar komponen yang digunakan dalam perancangan. Entitas level teratas hasil keluaran urutan posisi dari blok data input ini seperti yang ditunjukan pada gambar 4. Pada Gambar 4 sinyal clk dan we digunakan untuk mengendalikan kapan proses dimulai. Data input masuk melalui pin din(7:0) dengan lebar data 8 bit. Hasil proses dikeluarkan melalui pin dout(7:0) dengan lebar data 8 bit.



Gambar 4. Entitas top level blok data *input*

Rangkaian yang didapat dari hasil simulasi ini seperti yang ditunjukan pada Gambar 5. Rangkaian yang dihasilkan yaitu 1 buah RAM dan 1 buah Register 8-bit yang merupakan D *flip-flop* sebanyak 8 buah. Urutan untuk menulis dan membaca data masukan didefinisikan sebagai parameter i yang berfungsi sebagai *counter* untuk membangkitkan urutan mulai dari 1 sampai dengan 64. Data yang ditulis disimpan dalam RAM dibaca sesuai urutan yang dibangkitkan oleh *counter* i.

Pembacaan data input ini hasilnya menjadi output dout(7:0). Rangkaian D *flip-flop* berfungsi sebagai rangkaian logika sekuensial dimana di dalamnya terdapat peralatan memori dan pewaktu, sehingga output dari pembacaan data input ini disimpan dalam D *flip-flop*. Proses untuk membangkitkan *counter* i sampai dengan mendapatkan *output* diawali dengan sinyal we (*write enable*) yang berarti semua proses dimulai. Fungsi dari blok data *input* ini yaitu untuk memastikan urutan data *input* yang sudah didefinisikan terbaca dengan benar.

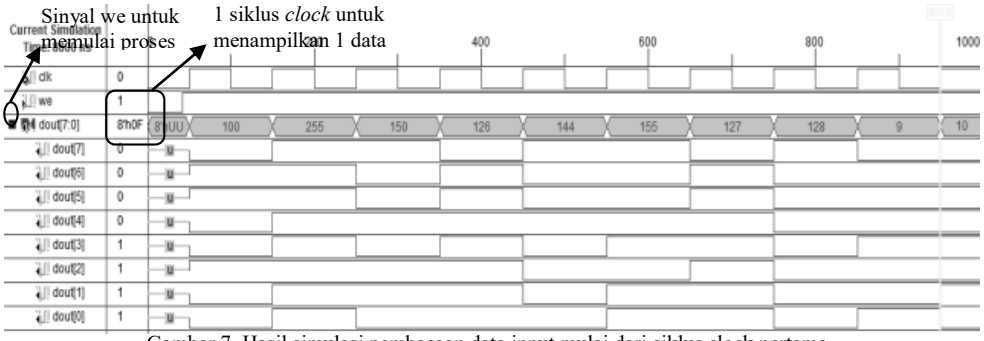


Gambar 5. Rangkaian data *input*

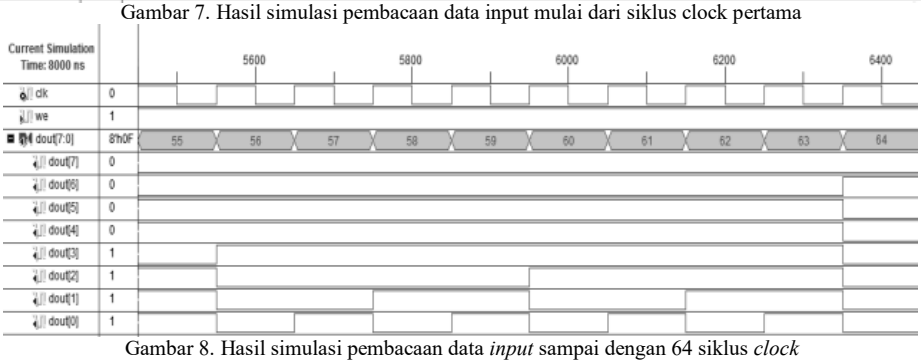
INPUT Project Status			
Project File:	Input.ise	**Current State:**	Placed and Routed
Module Name:	Input	**• Errors:**	No Errors
Target Device:	xc3s500e-5ft256	**• Warnings:**	17 Warnings
Product Version:	ISE, 8.1i	**• Updated:**	Mon Nov 21 12:14:41 2016

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Logic Distribution				
Number of occupied Slices	8	4,656	1%	
Number of Slices containing only related logic	8	8	100%	
Number of Slices containing unrelated logic	0	8	0%	
Total Number 4 input LUTs	16	9,312	1%	
Number used for 32x1 RAMs	16			
Number of bonded IOBs	18	190	9%	
IOB Flip Flops	8			
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	2,139			
Additional JTAG gate count for IOBs	864			

Gambar 6. Ringkasan sumber daya yang digunakan pada data *input*



Gambar 7. Hasil simulasi pembacaan data input mulai dari siklus clock pertama



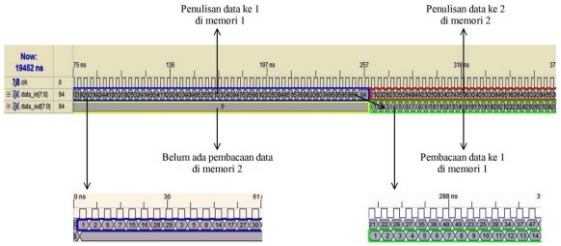
Gambar 8. Hasil simulasi pembacaan data *input* sampai dengan 64 siklus *clock*

Gambar 6 merupakan ringkasan sumber daya yang digunakan untuk RAM pada blok *input* untuk mendukung algoritma *zigzag scan* menggunakan metode pemetaan. Blok data *input* memerlukan LUTs sebanyak 16 slice atau sekitar 1% dari sumber daya yang tersedia sebanyak 9.312 slice bonded IOBs sebanyak 18 atau sekitar 9% dari sumber daya yang tersedia sebanyak 190 dan 1 GCLKs.

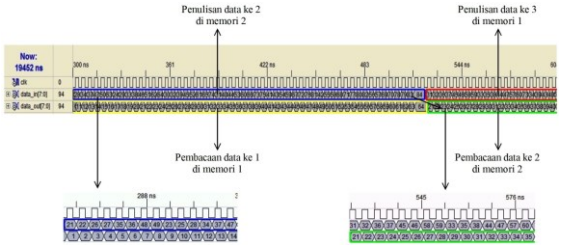
Data input yang disimpan dalam RAM dibaca secara berurutan berdasarkan *counter* i. Data input yang dibaca secara berurutan ditampilkan dalam variabel dout. Hasil simulasi dari proses pembacaan data *input* seperti yang ditunjukan pada Gambar 7, dout(7:0) merupakan data keluaran yang terdiri dari 8 bit *vector* dari proses pembacaan data *input*. Hasil simulasi menunjukkan pada siklus *clock* pertama pembacaan data didapat data yang ditampilkan dimulai dari urutan data pertama yaitu data yang bernilai 100. Proses pembacaan data ini terus berlanjut sampai dengan data terakhir yaitu data ke 64 yang bernilai 64. Untuk membaca satu data input diperlukan sebanyak 1 periode *clock*. Sesuai dengan hasil simulasi pada Gambar 8 untuk mendapatkan keluaran 64 data diperlukan 64 * 1 *clock* = 64 siklus *clock*.

Setelah proses pembacaan pada RAM tersebut selesai, selanjutnya dilakukan proses penulisan hasil pembacaan data *input*. Proses penulisan yang dilakukan berdasarkan urutan *zigzag scan* yang sudah didefinisikan. Urutan *zigzag scan* ini disimpan dalam ROM. Hasil penulisan data yang disimpan dalam RAM 1 dan RAM 2 kemudian dipetakan dengan posisi pemetaan yang tersimpan dalam ROM merupakan hasil *zigzag scan* dengan pemetaan.

Fungsi dari *zigzag scan* dengan pemetaan ini diperiksa menggunakan dua jenis simulasi, yaitu simulasi *behavioral* dan simulasi *post-route*. Hasil simulasi *behavioral* dari proses *zigzag scan* dengan pemetaan ini pada saat proses penulisan data ke dalam RAM 1 ditunjukkan pada Gambar 9. Gambar 10 menjelaskan pada 64 siklus *clock* pertama terjadi proses penulisan data ke 1 di RAM 1 dan pada 64 siklus *clock* pertama ini belum ada pembacaan data di RAM 2 karena belum terjadi proses penulisan data di RAM 2, selanjutnya pada 64 siklus *clock* yang kedua terjadi proses penulisan data ke 2 di RAM 2 dan terjadi proses pembacaan data ke 1 di RAM 1. Proses pada RAM 1 dan RAM 2 ini terus berulang untuk melakukan penulisan dan pembacaan data. Simulasi *behavioral* ini dilakukan untuk mengetahui kondisi ideal dari keluaran yang dihasilkan.



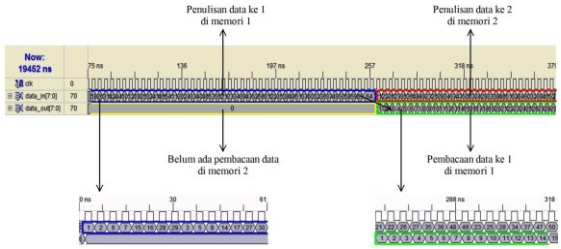
Gambar 9. Hasil simulasi *behavioral* proses penulisan data ke 1 di RAM 1 pada siklus *clock* pertama dan proses penulisan data ke 2 di RAM 2 dan pembacaan data ke 1 di RAM 1 pada siklus *clock* kedua



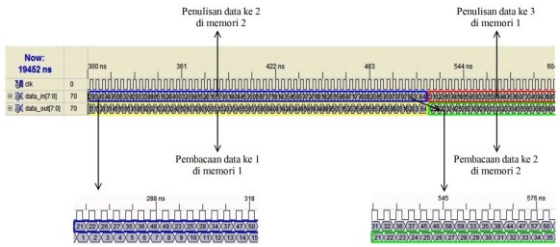
Gambar 10. Hasil simulasi *behavioral* proses penulisan data ke 3 di RAM 1 dan pembacaan data ke 2 di RAM 2 pada siklus *clock* ketiga

Hasil simulasi *post-route* dari proses *zigzag scan* dengan pemetaan ini pada saat proses penulisan data ke dalam RAM 1 ditunjukkan pada Gambar 11. Gambar 12 menjelaskan pada 64 siklus *clock* pertama terjadi proses penulisan data ke 1 di RAM 1 dan pada 64 siklus *clock* pertama ini belum ada pembacaan data di RAM 2 karena belum terjadi proses penulisan data di RAM 2, selanjutnya pada 64 siklus *clock* yang kedua terjadi proses penulisan data ke 2 di RAM 2 dan terjadi proses pembacaan data ke 1 di RAM 1. Proses pada RAM 1 dan RAM 2 ini terus berulang untuk melakukan penulisan dan pembacaan data. Simulasi *post-route* dilakukan untuk menyesuaikan hasil rancangan dengan karakteristik *hardware* yang digunakan.

Hasil simulasi *post-route* dari proses *zigzag scan* dengan pemetaan ini pada saat proses penulisan data ke dalam memori 1 ditunjukkan pada Gambar 11. Gambar 11 menjelaskan pada 64 siklus *clock* pertama terjadi proses penulisan data ke 1 di memori 1 dan pada 64 siklus *clock* pertama ini belum ada pembacaan data di memori 2 karena belum terjadi proses penulisan data di memori 2, selanjutnya pada 64 siklus *clock* yang kedua terjadi proses penulisan data ke 2 di momori 2 dan terjadi proses pembacaan data ke 1 di memori 1. Pada 64 siklus *clock* ketiga terjadi proses penulisan data ke 3 di memori 1 dan terjadi proses pembacaan data ke 2 di memori 2 seperti yang diperlihatkan pada Gambar 12. Proses penulisan data ke memori dan pembacaan data dari memori ini terus berulang dan bergantian antara memori 1 dan memori 2. Hasil pembacaan data baik dari memori 1 maupun memori 2 merupakan hasil *zigzag scan* dengan pemetaan.



Gambar 11. Hasil simulasi *post-route* proses penulisan data ke 1 di RAM 1 pada siklus *clock* pertama dan proses penulisan data ke 2 di RAM 2 dan pembacaan data ke 1 di RAM 1 pada siklus *clock* kedua



Gambar 12. Hasil simulasi *post-route* proses penulisan data ke 3 di RAM 1 dan pembacaan data ke 2 di RAM 2 pada siklus *clock* ketiga

Dari hasil simulasi *behavioral* didapat hasil yang sesuai dengan rancangan yang dibuktikan dengan sinyal yang dihasilkan, rancangan *zigzag scan* dengan metode pemetaan dapat mengurutkan koefisien-koefisien DCT-terkuantisasi. Hasil simulasi *behavioral* ini masih dalam tingkat *software*. Sedangkan simulasi *post-route* merupakan simulasi yang paling akurat sesuai dengan karakteristik *hardware* yang sebenarnya.

4. KESIMPULAN

Berdasarkan hasil uji coba yang sudah dilakukan menggunakan simulasi *behavioral* dan *post-route* didapat hasil RAM 1 dan RAM 2 dapat bekeja secara paralel, yaitu pada siklus clock yang sama RAM 1 melakukan proses penulisan dan RAM 2 melakukan proses pembacaan, pada siklus clock berikutnya RAM 1 melakukan proses pembacaan dan RAM 2 melakukan proses penulisan. Proses ini terus berulang dan bergantian antara RAM 1 dan RAM 2. Proses pembacaan dan penulisan antara RAM 1 dan RAM 2 dilakukan secara paralel maka proses scan koefisien DCT dapat berlangsung dengan cepat, sehingga algoritma *zigzag scan* dengan pemetaan ini dapat mendukung kecepatan kompresi citra. Proses *scan* data masukan dapat beroperasi secara *real-time* pada frekuensi 250 MHz atau untuk setiap byte data membutuhkan waktu proses selama 4 ns dan untuk ukuran matriks 8 x 8 data dibutuhkan waktu proses sebesar 256 ns. Hasil perancangan prototipe IC RAM dan ROM menggunakan FPGA dapat diimplementasikan untuk proses kompresi citra secara *real-time* dengan maksimum 3 Mega piksel per frame/citra dengan 25 frame per detiknya. Deskripsi komponen (IP *Core*) yang dihasilkan memerlukan LUTs sebanyak 16 *slice* atau sekitar 1% dari sumber daya yang tersedia sebanyak 9.312 *slice bonded* IOBs sebanyak 18 atau sekitar 9% dari sumber daya yang tersedia sebanyak 190 dan 1 GCLKs. Agar prototype IC *zigzag scan* yang dihasilkan lebih optimal pengujian harus dilakukan dengan menggabungkan seluruh tahapan kompresi citra, karena proses *zigzag scan* ini merupakan salah satu bagian dari tahap kompresi citra.

DAFTAR PUSTAKA

AGOSTINI, L.V., SILVA, I.S., BAMPI, S., 2007. Multiplierless and fully pipelined JPEG compression soft IP targeting FPGAs. Microprocessors and Microsystems. vol. 31. No. 8. pp. 487–497.

AWCOCK, G.J, THOMAS, R., 1995. Applied Image Processing. MACMILLAN Press LTD.

BUDIARTO, J., QUDSI, J., 2018. Deteksi Citra Kendaraan Berbasis Web Menggunakan Javascript Framework Library. Jurnal MATRIK Jurnal Manajemen. Teknik Informatika dan Rekayasa Komputer. Vol. 18. No. 1. pp. 125–133.

HARTONO, P., TRISMIYATI., 2014. Kamera pada pengolahan citra digital. Metal Indonesia. Vol. 36. No. 2. pp. 70–75.

IRFAN, M., SETIAWAN, H., 2022. Sistem pendukung Keputusan untuk UMKM. [ebook]. Implementasi Komputasi Akar kuadrat Resolusi Tinggi Pada Field Programmable Gate Array (FPGA). Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK). Vol. 9. No. 7. pp. 1617-1622.

KARDIAN, A.R., SUDIRO, S.A., MADENDA, S., 2018. Efficient Implementation of Mean, Variance and Skewness Statistic Formula for Image Processing using FPGA Device. Bulletin of Electrical Engineering and Informatics. Vol. 7. No. 3. pp. 386–392.

KUSUMA, E.D., 2010. Kompresi Citra JPEG Berbasis FPGA Xilinx Spartan-3E. Tesis Teknik Elektro Universitas Gajah Mada.

MADENDA, S., HAYET, L., BAYU, I., 2010. Kompresi Citra Berwarna Menggunakan Metode Pohon Biner Huffman. Laboratorium Pengolahan Citra dan Multimedia Universitas Gunadarma.

NAIK, C.N., VELVANI, V.M., PATEL, P.J., PAREKH, K.G., 2015. VLSI Based 16 Bit ALU with Interfacing Circuit. International Journal of Innovative and Emerging Research in Engineering. Vol. 2. No. 3. pp. 65–69.

PERTIWI, A., MADENDA, S., SUDIRO, S.A., 2014. Desain Skematik Algoritma

Histogram Untuk Kebutuhan Analisis Tekstur Citra Berbasis FPGA (Field Programmable Gate Array). Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT). Vol. 8. pp. 110–117.

PRAKASH, V.A.M., GURUMURTHY, K.S., 2012. VLSI Architecture for Low Power Variable Length Encoding and Decoding for Image Processing Applications. International Journal of Advance in Engineering & Technology. Vol. 4. No. 5. pp. 105–120.

PRATOMO, A.H., KASWIDJANTI., W., MU'ARIFAH, S. 2020., Implementasi Algoritma Region of Interest (ROI) Untuk Meningkatkan Performa Algoritma Deteksi Dan Klasifikasi Kendaraan. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK). Vol. 7. No. 1. pp. 155–162.

RUSTAMAJI, H.C., MARIANI, M., YUWONO, B. 2015., Aplikasi Kompresi Data Menggunakan Metode Huffman Statik Pada Perangkat Mobile Berbasis Android. Telematika. Vol. 11. No. 1. pp. 9-18.

SAPTONO, D., FIRDAUS, R.A., PERTIWI, A. 2013., Implementasi Penampil Citra Dengan Menggunakan Picoblaze FPGA. Seminar Nasional Teknologi Informasi dan Komunikasi Terapan (SEMANTIK). pp. 78–82.

SUGIARTOWO, AMBO, S.N., 2018., Implementasi Simulasi Media Pembelajaran Rangkaian Kombinasional Berbasis Kolaburasi Multimedia Simulator Dan Pemrogram Delphi. Jurnal Informatika UPGRIS. Vol. 4. No. 2. pp. 170-180.

TOCCI, R.J., WIDMER, N.S. 2001., Digital Systems Principles and Applications, 8th edition. Prentice Hall.

WANG, X., YU, M. 2011., Power research of JPEG circuits in FPGA. Proc. - 7th International Conference on Intelligent Information Hiding and Multimedia Signal Processing. pp. 206–208.

ZWOLINSKI, M. 2004., Digital System Design with VHDL. Pearson Education.