IMPLEMENTASI PARALEL DAN WAKTU-NYATA BEBERAPA ALGORITMA PRAPENGOLAHAN CITRA DENGAN MULTI-MIKROKONTROLER RISC

Eril Mozef

Jurusan Teknik Elektro, Politeknik Negeri Bandung
Jl. Geger Kalong Hilir Ds. Ciwaruga, Bandung, Indonesia
e-mail: erilmozef@yahoo.com

ABSTRAK

Pada paper ini disajikan hasil implementasi paralel dan waktu-nyata beberapa algoritma prapengolahan citra filtering dan binerisasi dengan solusi *reprogrammable, reconfigurable* dan *Virtual Peripheral* menggunakan multi-mikrokontroler RISC. Algoritma yang diimple-mentasikan yaitu filter rata-rata, filter intensitas, binerisasi sederhana dan binerisasi lokal. Dengan menggunakan mikrokontroler Scenix SX28 dan SX48, 75 MIPS (dengan frekuensi 75 Mhz dan kecepatan eksekusi 13,3 ns/kataInstruksi) dan dikombinasikan dengan teknik *pipelining*, berhasil didapatkan kecepatan waktu-nyata 30 frame/detik dengan resolusi horizontal 128 pixel untuk keseluruhan algoritma yang diimplementasikan. Untuk suatu sistem prapengolahan citra (filtering dan binerisasi) dibutuhkan paling banyak 6 buah chip (4 buah SX28AC75 dan 2 buah SX48BD75). Hasil ini merupakan penyempurnaan dari disain sebelumnya yang menggunakan 10 buah chip SX28AC, 50 MIPS, yang mana aspek waktu-nyata uutuk keseluruhan algoritma masih belum terpenuhi.

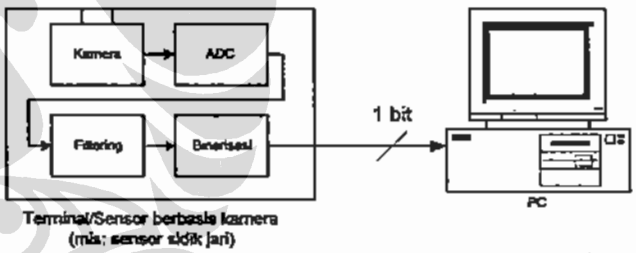
Kata kunci: Pengolahan citra waktu-nyata, Pra-pengolahan citra, *Filtering*, Binerisasi, Mikrokontroler, Scenix, *Virtual Peripheral*.

Makalah diterima [27 Agustus 2002]. Revisi akhir [15 Oktober 2002].

1. PENDAHULUAN

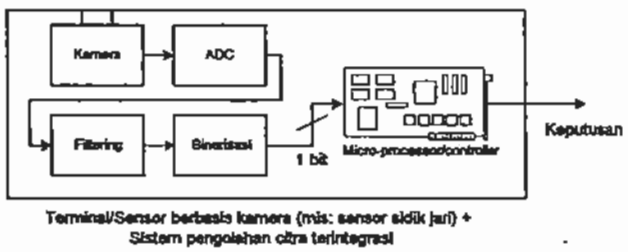
Filtering dan binerisasi merupakan proses prapengolahan citra yang sangat penting dalam suatu sistem pengolahan citra. Filtering adalah tahapan dimana kualitas citra yang diperoleh dari kamera diperbaiki. Sedangkan Binerisasi adalah tahapan dimana jumlah informasi citra direduksi tanpa menghilangkan pengertian citra itu sendiri [1].

Filtering dan binerisasi akan sangat mudah bila realisasinya berbasis PC. Namun proses filtering dan binerisasi ini akan jadi subjek yang menarik bila kita dihadapkan pada masalah transfer citra dari sebuah sensor/terminal dengan jarak cukup jauh. Dalam hal ini kabel data paralel tidak dimungkinkan. Salah satu solusinya adalah dengan melakukan proses filtering dan binerisasi ini langsung pada sensor/terminal dan mengirimkan hanya data binernya secara serial ke PC dengan demikian waktu pengiriman menjadi lebih singkat (Gambar 1). Ini dapat terjadi misalnya pada sistem absensi/akses kontrol sidik-jari multi terminal berbasis jaringan.



Gambar 1. Sistem Pengenalan Sidik Jari berbasis PC dengan bagian Prapengolahan Citra-nya *Stand-alone*

Alternatif lain adalah mengolah langsung citra biner yang dihasilkan dengan prosesor lokal untuk membangun sistem keseluruhan yang *stand-alone* (Gambar 2).



Gambar 2. Sistem Pengenalan Sidik Jari *Stand-alone*

Sistem yang ada dipasaran untuk masalah ini cukup banyak yang ditawarkan misalnya: BAC, Identix, Veridicom, dan lain-lain. Sistem-sistem ini biasanya menggunakan solusi teknologi DSP, FPGA atau ASIC pada sensor/terminalnya. Bila kita ingin mengembangkan sendiri sistem seperti ini tentunya kita akan dihadapkan pada masalah klasik: 1). Komponennya sulit didapatkan dipasaran, 2). Tidak dapat dibeli eceran, 3). Harganya mahal dan 4). Waktu pengembangannya cukup lama. Kita dapat juga menggunakan komponen video jadi, namun masalah 1, 2 dan 3 tetap tak terpecahkan disamping penggunaan komponen atau IC tersebut tidak fleksibel.

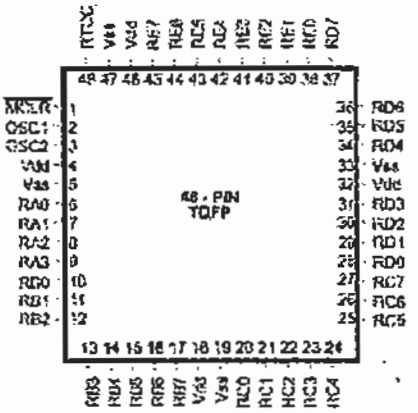
Pada paper ini disajikan hasil implementasi paralel dan waktu-nyata beberapa algoritma prapengolahan citra filtering dan binerisasi dengan solusi *reprogrammable, reconfigurable* dan *Virtual Peripheral* menggunakan multi-mikrokontroler RISC. Algoritma yang diimplementasikan yaitu filter rata-rata, filter intensitas, binerisasi sederhana dan binerisasi lokal.

Pada paper ini, penulis berusaha menyempurnakan disain sebelumnya [8] yang menggunakan 10 buah chip SX28AC, 50 MIPS (50 Mhz, 20 ns/kataInstruksi), yang mana aspek waktu-nyata untuk keseluruhan algoritma masih belum terpenuhi. Pada disain kali ini, penulis mencoba untuk menggunakan chip berkemampuan lebih besar yaitu 75 MIPS (bertipe SX28AC75 dan SX48BD75) yang bekerja pada frekuensi 75 Mhz (13,3 ns/kataInstruksi) dengan maksud: 1) untuk mendapatkan kecepatan waktu-nyata 30 frame/detik dengan resolusi horizontal 128 pixel untuk keseluruhan algoritma yang diimplementasikan 2) untuk sekaligus mereduksi jumlah chip yang digunakan menjadi sekitar 6 buah chip.

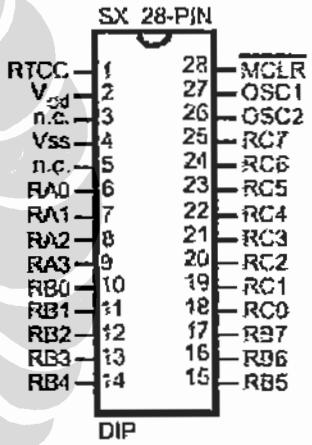
Aplikasi yang ditargetkan untuk sistem ini adalah pengenalan sidik jari *stand-alone* atau multi-terminal berbasis jaringan. Sistem ini dapat pula diterapkan pada aplikasi pengenalan karakter atau aplikasi lain yang memerlukan citra biner dan yang menuntut proses waktu-nyata.

2. MIKROKONTROLER SCENIX DAN KONSEP VIRTUAL PERIPHERAL

Konsep *Virtual Peripheral* relatif baru dan dicetuskan pertama kali tahun 1998 oleh perusahaan Ubicom (sebelumnya Scenix) dengan produknya Scenix. Scenix adalah mikrokontroler 8 bit jenis baru ini berarsitektur *RISC* Hardvard dan masih satu turunan dengan keluarga PIC yang terkenal itu. Scenix memiliki keuntungan cepat namun murah. Scenix memiliki 3 jenis chip berdasarkan kemampuan kerjanya yaitu 50 *MIPS* (50 Mhz, 20 ns/kataInstruksi), 75 MIPS (75 Mhz, 13,3 ns/kataInstruksi) sampai dengan 100 MIPS (100Mhz, 10 ns/kataInstruksi). Bandingkan dengan keluarga MCS'51 dengan harga yang relatif sama hanya mampu mengeksekusi instruksi dalam waktu sekitar 1000ns (1µs).



Gambar 3. Pin Scenix SX48BD



Gambar 4. Pin Scenix SX28AC

Scenix dapat diprogram berulang-ulang dan sudah dilengkapi dengan SPI untuk men-*download* program langsung dari PC ke pin Scenix tanpa perlu lagi *downloader* khusus. Scenix dapat dibeli dengan harga persatuan dibawah 3 US dollar (apalagi dengan kuantitas banyak) relatif sama dengan keluarga MCS'51. Bandingkan dengan DSP seperti TMS320C5402 yang baru bisa dibeli dengan harga 3 US dollar bila kuantitasnya mencapai 1000. Apalagi dengan FPGA seperti Altera EPF10K atau XILINX harga persatuannya bisa mencapai 200 US dollar. Untuk kelas yang sama belum ada mikrokontroler lain yang memiliki perbandingan kualitas-harga sebaik ini.

Oleh karena kecepatannya yang tinggi, kesederhanaan pemrogramannya dan harganya yang murah ini, Scenix cocok dijadikan sebagai "komponen maya" (*Virtual Peripheral*). Pada konsep FPGA, kita membuat komponen secara fisik didalam sebuah chip dengan

mengkonfigurasi *switch*. Pada konsep *Virtual Peripheral*, kita membuat komponen yang diinginkan didalam sebuah chip dengan mengimplementasikan fungsi komponen tersebut secara software. Komponen yang dimaksud tentunya sebatas pada spesifikasi digital (bukan komponen analog). Dengan konsep ini kita dapat membuat sebuah komponen spesifik misalnya: DTMF, FFT, PWM, RTC dan lain sebagainya dengan hanya merubah-rubah program *assembly*-nya.

Komponen apa saja yang telah berhasil diimplementasikan dapat dilihat pada library *Virtual Peripheral* [6][7]. Library ini dapat didownload secara gratis untuk diimplementasikan ke dalam *chip* Scenix yang kita miliki.

Pada paper ini, penulis mengimplementasikan setiap bagian prapengolahan citra dengan komponen maya yang direalisir dengan mikrokontroler Scenix. Scenix memiliki keuntungan dapat difungsikan baik sebagai komponen maya (*Virtual Peripheral*) maupun prosesor. Karena harganya yang relatif murah tersebut, kita tidak perlu ragu untuk menjadikannya sebagai sebuah komponen walaupun dengan begitu kita terpaksa mengorbankan aspek prosesornya. Ini bukan berarti suatu pemborosan namun memang Scenix dirancang untuk tujuan itu (ingat konsep *Virtual Peripheral*).

Pada disain ini, penulis menggunakan 2 jenis mikrokontroler Scenix yaitu SX48BD (48 pin) (Gambar 3) dan SX28AC (28 pin) (
Gambar 4). Bagian prosesornya dapat melakukan operasi dalam 8, 4 atau 1 bit. Scenix memiliki jumlah instruksi yang sedikit dan sederhana yaitu 69 instruksi. Setiap pin Scenix dapat dikonfigurasi baik sebagai input ataupun output.

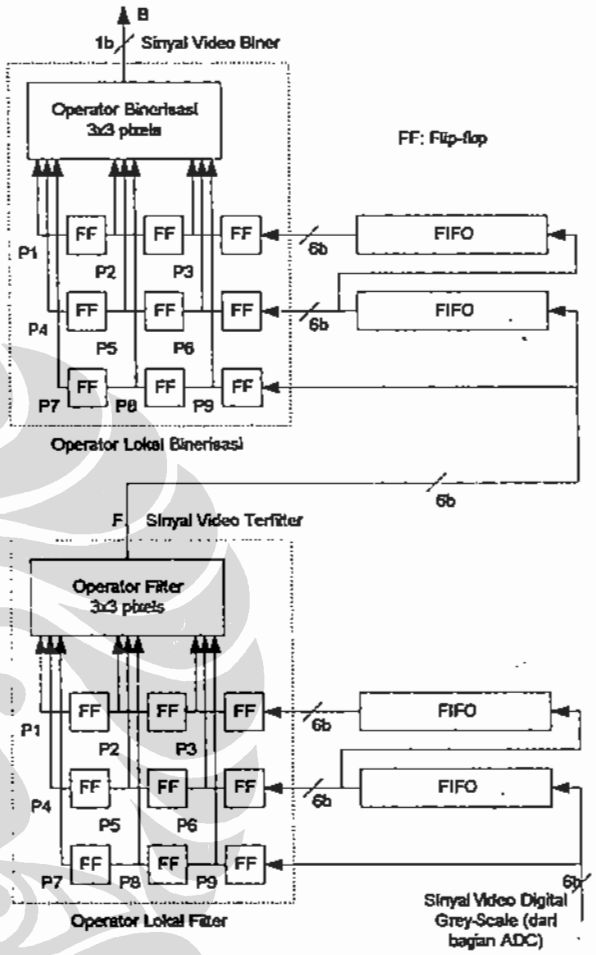
Scenix SX48BD memiliki port I/O berjumlah 36 terdiri dari Ra (4 pin), Rb (8 pin), Rc (8 pin), Rd (8 pin), dan Re (8 pin). SX48BD memiliki 262x8 bit SRAM untuk variabel dan 2048x12 bit Flash memory untuk menyimpan instruksi.

Scenix SX28AC memiliki port I/O berjumlah 20 terdiri dari Ra (4 pin), Rb (8 pin), dan Rc (8 pin). Setiap pin dapat dikonfigurasi sebagai input ataupun output. SX28AC memiliki 136x8 bit SRAM untuk variabel dan 2048x12 bit Flash memory untuk menyimpan instruksi.

Untuk lebih detilnya mengenai *datasheet* dapat dilihat pada [5] sedangkan untuk aspek pemrogramannya dapat dilihat pada [4].

3. DESAIN UMUM

Secara umum blok diagram sistem prapengolahan citra masih sama dengan disain terdahulu [8]. Hanya yang membedakan nantinya adalah jenis chip yang dipakai, jumlahnya dan implementasi algoritmanya serta proses waktu-nyatanya. Blok diagram ini terdiri dari dua tahap yang saling mendukung yaitu filter dan binerisasi (Gambar 5). Output dari kamera setelah di-*digitize* akan memberikan sinyal video digital *grey-scale*. Sinyal ini kemudian diumpankan kebagian filter. Bagian ini memiliki 2 buah FIFO (First In First Out) 6 bit untuk men-*delay* informasi video sebanyak 1 dan 2 line.



Gambar 5. Blok Diagram Sistem

Selanjutnya Output FIFO diumpankan ke operator filter (3x3) 6 bit. Keuntungan penggunaan Scenix terlihat pada operator ini dimana kita dapat dengan mudah nantinya menghitung, memproses dan merekonfigurasi data berukuran (3x3) 6 bit pada kecepatan video. Output dari operator filter ini kemudian diumpankan ke FIFO bagian binerisasi dengan spesifikasi yang sama dengan bagian filter. Output FIFO ini kemudian diumpankan ke operator binerisasi. Operator binerisasi ini memiliki struktur yang sama dengan operator filter hanya program penghitungan operatornya saja yang berbeda. Selanjutnya output video biner ini dapat dibuffer pada memory citra 1 bit bila diperlukan untuk kemudian dapat diproses secara lokal atau diumpankan ke PC melalui RS232. Untuk komponen RS232-pun dapat direalisasi dengan sebuah

Scenix. Seluruh proses dirancang untuk dapat bekerja secara *on-the-fly* sehingga aspek waktu-nyata dapat terpenuhi.

3.1. Operator Lokal Filter

Filter ada bermacam-macam jenisnya [2][3]. Pada paper ini dibahas hanya Filter intensitas dan Filter rata-rata karena 2 alasan:

(1). Berhubungan erat dengan citra yang dihasilkan oleh sensor yang akan digunakan pada aplikasi yang akan dikembangkan.

(2). Perhitungannya melibatkan pixel yang bertetanggaan 3x3 dan mampu dikerjakan oleh sistem yang dirancang.

Adapun Filter-filter ini dapat dihitung dengan rumus:

Filter rata-rata:

$$Fr=(P1+P2+P3+P4+P5+P6+P7+P8+P9)/9 \quad (1)$$

Filter intensitas:

$$Fi=(P1+P2+P3+P4+P6+P7+P8+P9)/8 \quad (2)$$

P1...P9 adalah pixel-pixel yang terdapat pada matriks pixel 3x3 yang dapat dilihat pada Gambar 6.

P1	P2	P3
P4	P5	P6
P7	P8	P9

Gambar 6. Matriks Pixel 3x3

3.2. Operator Lokal Binerisasi

Binerisasi adalah suatu proses yang bertujuan memperkecil jumlah informasi pada sebuah citra. Dalam hal ini mengubah citra *grey-level* 256 level menjadi citra biner 2 level, hitam dan putih saja. Ada beberapa teknik binerisasi yang pada umumnya adalah merupakan perbandingan *threshold* (batas ambang), misalnya:

Binerisasi Sederhana:

$$\text{If } P5 \geq Th1 \text{ Then } Bs=1 \text{ Else } Bs=0 \quad (3)$$

di mana Th1: Batas ambang

Binerisasi Lokal:

$$\text{If } P \geq Th2 \text{ Then } Bl=1 \text{ Else } Bl=0 \quad (4)$$

di mana:

$P= (P1+P2+P3+P4+P6+P7+P8+P9)/8$

Th2: Batas ambang

4. IMPLEMENTASI

Untuk pengimplementasian algoritma filtering dan binerisasi kita asumsikan bahwa:

(1). Data video yang diproses adalah komposit hitam-putih, sudah di-*digitize* dan berukuran 6 bit. Alasan mengapa 6 bit adalah:

a. Yang kita proses adalah informasi videonya dan bukan aspek kualitas gambar sehingga 6 bit sudah cukup mewakili informasi yang terkandung pada gambar tersebut.

b. Supaya dalam implementasi, kita tidak menggunakan register yang terlalu panjang untuk menampung hasil perhitungan mathematis pada bagian filter dan binerisasi.

c. Dengan 6 bit kita dapat mereduksi jumlah informasi video secara siknifikan dan sekaligus mempercepat proses filtering dan binerisasi.

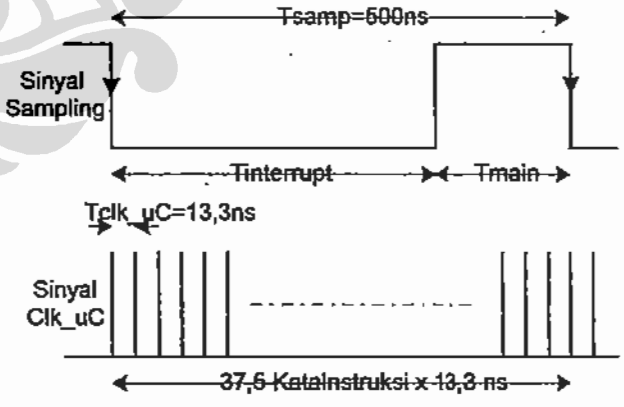
d. IC ADC (*Analog to Digital Converter*) komersial untuk video banyak yang berukuran 6 bit dan dapat dibeli dengan harga yang relatif murah.

(2). Mikrokontroler yang digunakan adalah Scenix SX28AC.

4.1. Total Instruksi yang Diijinkan Per 1 Siklus Sampling

Waktu 1 siklus sampling (Tsampling) ditentukan dengan cara membagi waktu 1 line video (TlineVideo) dengan total sampling (totSampling) (persamaan 5).

$$Tsampling = TlineVideo/totSampling \quad (5)$$



Gambar 7. Hubungan antara Tsampling, Tclk_uC, Tinterrupt, Tmain dan kataInstruksi

Untuk mencapai eksekusi waktu-nyata maka total waktu eksekusi instruksi-instruksi di Scenix (nbKataInstruksi) harus dapat dikerjakan dalam 1 siklus sampling Tsampling (persamaan 6).

$$nbKataInstruksi \times Tclk\_uC \leq Tsampling \quad (6)$$

Catatan: KataInstruksi (*Instruction Word*) di Scenix merupakan satuan terkecil siklus instruksi karena dalam 1 Instruksi Scenix bisa terdiri dari 1 atau 2 kataInstruksi.

Pada Gambar 7 ditunjukkan 2 buah sinyal: Sinyal Sampling dan Sinyal Clk_uC yang memperjelas hubungan antara Tsampling, Tclk_uC, Tinterrupt, Tmain dan kataInstruksi.

Supaya eksekusi program berjalan dengan baik maka total waktu eksekusi instruksi dibagian main (Tmain) dan interrupt (Tinterrupt) harus lebih kecil sama dengan waktu sampling, sehingga:

$$Tinterrupt+Tmain \leq Tsampling \quad (7)$$

Bila diketahui:

$$Tinterrupt=nbKI\_int \times Tclk\_uC \quad (8)$$

dan

$$Tmain=nbKI\_main \times Tclk\_uC \quad (9)$$

di mana:

nbKI_int : banyaknya kataInstruksi di bagian interrupt;

nbKI_main : banyaknya kataInstruksi di bagian main program.

Bila persamaan 8 dan 9 dimasukkan ke persamaan 7 dan persamaan ini digabung dengan persamaan 6 maka didapat:

$$nbKataInstruksi = nbKI\_int + nbKI\_main \quad (10)$$

Ini berarti bahwa total kataInstruksi ditentukan oleh banyaknya kataInstruksi dibagian interrupt + banyaknya kataInstruksi dibagian main program.

Bila kita asumsikan bahwa totSampling=128 dan TlineVideo=64μs maka dari persamaan 5 didapat Tsampling=500ns atau Fsamp=2MHz. Dengan memasukkan Tsampling yang didapat ke dalam persamaan 6 maka bila kita gunakan chip Scenix dengan kecepatan instruksi 75 MIPS dengan kata lain Tclk_uC=13,3ns akan didapat nbKataInstruksi ≤ 37,5. Ini berarti bahwa dalam mengimplementasi setiap algoritma ke dalam Scenix, total instruksi di bagian interrupt dan main program tidak boleh melebihi 37,5 kataInstruksi. Untuk membuktikan apakah kondisi ini dapat dipenuhi maka sebaiknya kita lihat pada bagian implementasi algoritma berikut ini.

4.2. Implementasi Virtual Peripheral FIFO 128x8

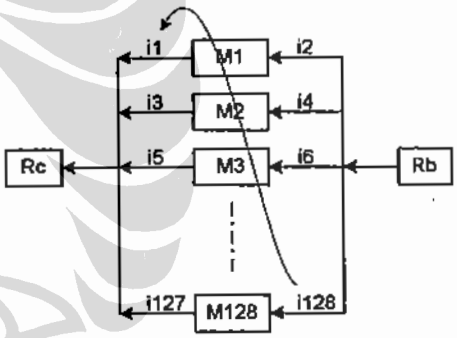
Untuk dapat men-*delay* 1 line video yang disamplingsebanyak 128 pixels maka diperlukan FIFO dengan panjang 128 juga. Untuk dapat mendisain FIFO secara efisien maka hal-hal berikut harus dipertimbangkan

(1). Pemanfaatan RAM internal yang berjumlah 136 byte. Dimana sebanyak 128 byte akan digunakan sebagai buffer FIFO sedangkan sisanya 8 byte untuk variabel.

(2). Penggunaan teknik membaca dan menulis data yang efisien pada memori yang dibentuk secara circular.

Teknik ini dapat dijelaskan sebagai berikut.

Bila M1, M2, M3, ..., M128 merupakan register untuk menampung data, Rb dan Rc merupakan input dan output data (Gambar 8). Maka secara circular kita menjalankan i1, i2, i3, ..., i128 kemudian kembali lagi ke i1 dan begitu seterusnya (i dibaca instruksi). Instruksi i1 memindahkan data dari M1 ke register output Rc. Sedangkan i2 memindahkan data dari register input Rb ke M1 dan begitu seterusnya sampai i128.

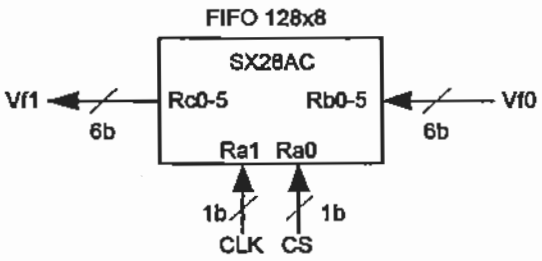


Gambar 8. Diagram proses FIFO 128x8 pada SX28

FIFO yang dimaksud, kita asumsikan memiliki input 6 bit Rb0-5, ouput 6 bit Rc0-5, input kontrol Ra1 untuk Clk dan Ra0 untuk CS masing-masing 1 bit seperti diperlihatkan pada blok diagram

Gambar 9. CS (Composite Synchronisation) adalah sinyal sinkronisasi komposit untuk mensikronkan awal dari sebuah line video.

Catatan: Clk berbeda dengan Clk_uC. Clk adalah clock sistem yang periodanya sama dan tersinkronisasi dengan Tsampling sedangkan Clk_uC adalah clock mikrokontroler.



Gambar 9. Virtual Peripheral FIFO 128x8 dengan SX28AC

Adapun implementasi algoritma FIFO ke dalam bentuk program assembly Scenix dapat dilihat pada listing berikut ini.

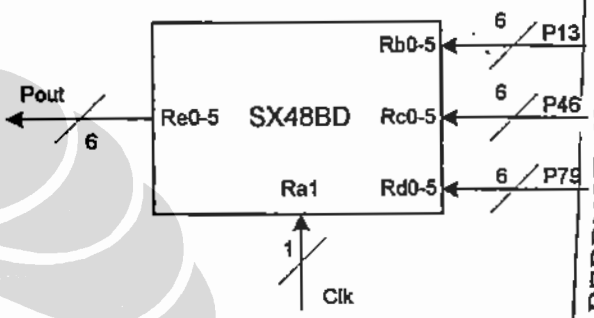
```
1 Main
2         Clr     FSR
3 lpCS    Jnb     Ra.0,lpCS     2,4(jump)
4 lpClk   Jnb     Ra.1,lpClk    2,4(jump)
5         Setb    FSR.4           1
6         Mov     Rc,IND        2
7         Mov     IND,Rb        2
8         Ijnz    FSR,lpClk     2,4(jump)
9         Jmp     lpCS          3

Total siklus      14
```

Untuk menghemat siklus, program ini diimplementasikan keseluruhannya pada main program (main) dan bukan pada bagian Interrupt. Begitu juga data yang masuk dan keluar dari FIFO adalah 6 bit sedangkan pemrosesan dalam 8 bit. Hal ini tidak menjadi masalah dan tidak perlu di-*masking* karena data-data tersebut tidak untuk dihitung namun sekedar ditransfer jadi tidak memiliki *carry*. Dua bit input yang tak terpakai dapat dihubungkan ke lojik 0 atau *ground*. Hal ini berlaku pula untuk
Gambar 10,
Gambar 13, dan
Gambar 15.

Setiap baris program dapat dijelaskan sebagai berikut (B dibaca Baris).

B1: Label untuk menunjukkan awal main program.
B2: Mengosongkan register FSR (register pointer alamat RAM). Bagian ini tidak masuk dalam loop jadi tidak masuk dalam perhitungan total siklus.
B3: Mendeteksi adanya sinyal CS (sisi naik). Jika ada maka keinstruksi berikutnya. Jika tidak maka tunggu sampai ada.
B4: Mendeteksi adanya sinyal Clk (sisi naik). Jika ada maka keinstruksi berikutnya. Jika tidak maka tunggu sampai ada.
B5: Awal alamat Bank RAM yang ditunjuk oleh pointer FSR ada 7 yaitu 10h, 30h, 50h, 70h, 90h, B0h, D0h dan F0h. Agar pointer menunjuk kealamat yang benar maka bit ke-4 dari FSR dipaksa menjadi 1.
B6: Data pada alamat yang ditunjuk oleh FSR dioutputkan ke register Rc.
B7: Data pada register Rb diinputkan pada register yang alamatnya ditunjuk oleh FSR.
B8: Isi dari FSR dinaikkan satu. Bila pada saat dinaikkan FSR belum kembali ke-nol maka ulangi Baris 4 s/d 7. Bila tidak maka ke instruksi berikutnya.
B9: Kembali ke Baris 3.

4.3. Implementasi Virtual Peripheral Operator Lokal Filter



Gambar 10. Virtual Peripheral Operator Lokal Filter dengan SX48BD

Pada disain terdahulu [8] proses perhitungan dipecah menjadi 3 bagian yang masing-masing bagian diimplementasikan ke dalam sebuah chip Scenix SX28AC-50MHz dan ini ternyata memiliki permasalahan dalam hal minimisasi kataInstruksi. Pada implementasi Operator Lokal Filter dan Binerisasi kali ini, penulis menggunakan Scenix dengan jumlah port dan frekuensi kerja lebih tinggi yaitu tipe SX48BD dengan frekuensi 75MHz dan 36 port I/O (
Gambar 10). Dengan demikian jumlah kataInstruksi yang diperbolehkan lebih banyak (37,5) dan seluruh input dan output dapat ditangani sekaligus.

Perlu dicatat disini bahwa Operator ini hanya memerlukan sinyal clock CLK dan tidak lagi memerlukan sinyal CS sebab *stream* data yang keluar dari FIFO sudah tersinkronisasi dengan baik terhadap line.

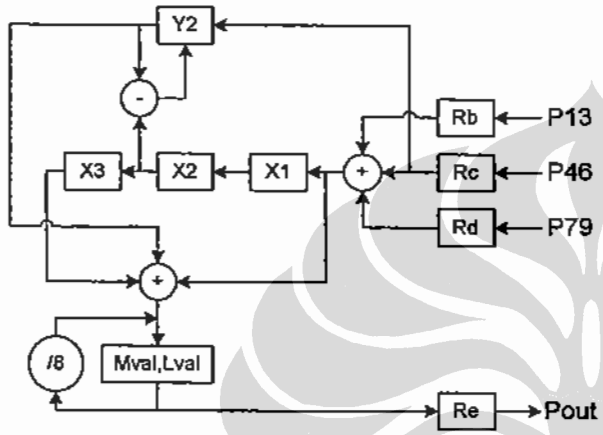
4.3.1. Implementasi Operator Filter Intensitas

Agar Operator Lokal ini dapat dieksekusi dalam waktu kurang dari 500ns (1 clock) dan kurang dari 37,5 kataInstruksi maka caranya adalah dengan menghitung harga pixel-pixel yang masuk ke Operator Lokal dari 3 line FIFO per 3 pixel dalam satu kolom (misalnya P1,

P4, dan P7) dan menggesernya hasilnya ke register-register yang berada disebelah kirinya (Gambar 11).

Bila kita asumsikan bahwa P1, P2, P3, P4, P5, P6, P7, P8 dan P9 telah tersedia dalam bentuk matriks pada FIFO (dengan urutan seperti pada Gambar 6) dan siap untuk diumpankan pada operator lokal maka setelah 3 clock berturut-turut berikutnya, nilai register Rb, Rc, Rd, x1, x2, x3 dan Re adalah sebagai berikut:

Re=(P1+P2+P3+P4+P6+P7+P8+P9)/8
Rb=P3, Rc=P6, Rd=P9,
x3=P2+P5+P8, x2= P3+P6+P9, x1=P3+P6+P9.



Gambar 11. Diagram proses filter intensitas pada SX48

Adapun implementasi algoritma filter intensitas dapat dilihat pada listing berikut ini.

```
1  Main
2      Jb     Ra.1,main    ;2,4(jump)
3      Clr    bufc         ;1
4      Clr    mval         ;1
5      Sub    y2,x2        ;2
6      Mov    w,Rb         ;1
7      Add    w,Rc         ;1
8      Add    w,Rd         ;1
9      Mov    x1,w         ;1
10     Add    w,y2         ;1
11     Rl     bufc         ;1
12     Add    w,x3         ;1
13     Rl     bufc         ;1
14     Mov    lval,w       ;1
15     Snz    bufc         ;1
16     Inc    mval         ;1
17     Rr     mval         ;1
18     Rr     lval         ;1
19     Rr     mval         ;1
20     Rr     lval         ;1
21     Rr     mval         ;1
22     Rr     lval         ;1
23     Mov    Re,lval      ;2
24     Mov    x3,x2        ;2
25     Mov    x2,x1        ;2
26     Mov    y2,Rc        ;2
27     Jmp    Main         ;3

Total siklus       34
```

Untuk menghemat siklus, program ini diimplementasikan keseluruhannya pada main program (main) dan bukan pada bagian Interrupt. Setiap baris program dapat dijelaskan sebagai berikut (B dibaca Baris).

B1: Label untuk menunjukkan awal program.

B2: Pendeteksian munculnya clock CLK (pada sisi naik).

B3-4: Variabel bufc dan mval diinisialisasi dengan nol.

B5: Nilai x2 dikurangi y2 hasilnya ditampung y2. x2=Rb''+Rc''+Rd'' pada clock sebelumnya. Sedangkan y2=Rc''. Jadi setelah proses pengurangan y2=Rb''+Rd''. Harga Rc'' yang identik dengan pixel P5 sudah dihilangkan sesuai dengan rumus filter intensitas.

B6-9: Rb'+Rc'+Rd' hasilnya ditampung di register w dan x1. Sampai disini dipastikan tidak ada carry karena nilai Rb', Rc' dan Rd' maksimum adalah 63x3=189 sedangkan w dan x1 dapat menampung maksimum 255.

B10: w=w+y2 dimana w=Rb'+Rc'+Rd' dan y2=Rb''+Rd''. Jadi sekarang w= Rb'+Rc'+Rd'+Rb''+Rd'' dan carry C=1 (dalam hal terburuk) karena nilai w sekarang bisa melebihi 255.

B11: Pada saat di-Rotate Left (Rl), nilai C akan masuk pada LSB dari variabel bufc. Jadi bufc=00000001 (biner).

B12: w=w+x3 dimana w= Rb'+Rc'+Rd'+Rb''+Rd'' dan x3=Rb'''+Rc'''+Rd'''. Jadi w= Rb'+Rc'+Rd'+Rb''+Rd''+Rb'''+Rc'''+Rd''' dan carry C=1.

B13: sama dengan B11, sekarang bufc=00000011.

B14: Harga w disimpan dulu pada lval.

B15-16: Jika bufc tidak nol maka mval=00000001

B17-22: mval dan lval membentuk bilangan 16 bit. Pada saat Rotate Right (Rr) terhadap mval maka seluruh bit tergeser kekanan sekali. Pada saat masuk, bit paling kiri MSB akan terisi nilai carry C berikutnya pada saat keluar, nilai bit paling kanan akan ditampung oleh carry C. Pada saat Rotate Right (Rr) terhadap lval, nilai C yang berisi LSB dari mval akan masuk ke MSB dari lval, kemudian LSB dari lval akan mengisi C. Proses menggeser mval dan lval ini diulang sebanyak 3 kali untuk mendapatkan pembagian 8.

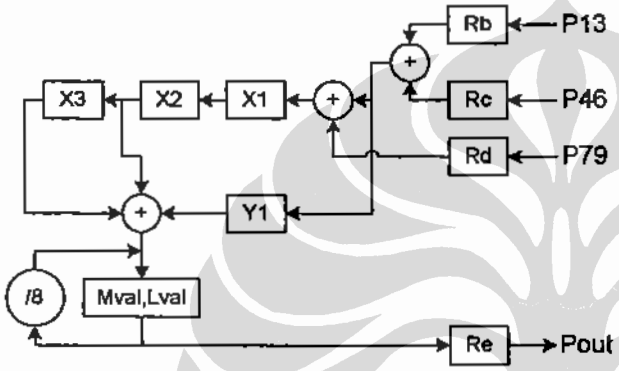
B23: Hasil pembagian 8 pada proses sebelumnya menyebabkan bilangan tidak lebih dari 255 dan ini di-outputkan ke port Re.

B24-26: Proses menggeser nilai x2 ke x3 dan nilai x1 ke x2 untuk menyiapkan data pada clock berikutnya.

B27: kembali ke proses awal.

4.3.2. Implementasi Operator Filter Rata-rata

Rumus yang telah diberikan pada persamaan 1 untuk filter rata-rata mengandung pembagian dengan 9. Pembagian ini sulit untuk diimplementasikan karena Scenix tidak memiliki perhitungan pecahan (floating point). Untuk mengatasi hal ini, maka salah satu pixel dikorbankan dan tidak diikutkan dalam perhitungan [3]. Cara ini tidak akan mengurangi kualitas filter secara keseluruhan dan mata kita dalam hal ini tidak dapat melihat perbedaannya. Dalam hal ini yang dipilih misalnya pixel yang berada dipojok P9. Proses yang terjadi pada register internal mikrokontroler ditunjukkan pada Gambar 12.



Gambar 12. Diagram proses filter rata-rata pada SX48

Adapun implementasi algoritma filter rata-rata dapat dilihat pada listing berikut ini.

```
1  Main
2      Jb     Ra.1,main     ;2,4(jump)
3      Clr    bufc          ;1
4      Clr    mval          ;1
5      Mov    w,Rb          ;1
6      Add    w,Rc          ;1
7      Mov    y1,w          ;1
8      Add    w,Rd          ;1
9      Mov    x1,w          ;1
10     Mov    w,y1          ;1
11     Add    w,x2          ;1
12     Rl     bufc          ;1
13     Add    w,x3          ;1
14     Rl     bufc          ;1
15     Mov    lval,w        ;1
16     Snz    bufc          ;1
17     Inc    mval          ;1
18     Rr     mval          ;1
19     Rr     lval          ;1
20     Rr     mval          ;1
21     Rr     lval          ;1
22     Rr     mval          ;1
23     Rr     lval          ;1
24     Mov    Re,lval       ;2
25     Mov    x3,x2         ;2
26     Mov    x2,x1         ;2
27     Jmp    Main          ;3

Total siklus       32
```

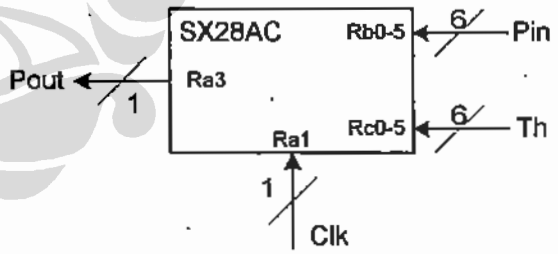
B2-4: Sama dengan keterangan B2-4 filter intensitas.
B5-B6: w=Rb'+Rc'
B7: y1=w= Rb'+Rc'
B8: w= Rb'+Rc'+Rd'
B9: x1= w= Rb'+Rc'+Rd'
B10-11: w=y1+x2= Rb'+Rc'+Rb''+Rc''+Rd''
B12: Bila ada Carry simpan di bufc
B13: w= w+x3= Rb'+Rc'+ Rb''+Rc''+Rd''+ Rb'''+Rc'''+Rd'''. Perhatikan bahwa Rd'=P9 telah dihilangkan dari perhitungan.
B14: Bila ada Carry simpan di bufc
B15-B26: Sama dengan keterangan B14-25 filter intensitas.
B27: kembali ke proses awal.

4.4. Implementasi Virtual Peripheral Operator Lokal Binerisasi

Pada bagian ini akan dibahas implementasi operator lokal binerisasi sederhana dan lokal.

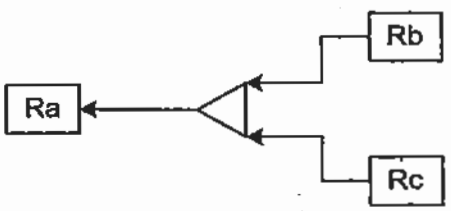
4.4.1. Implementasi Operator Binerisasi Sederhana

Binerisasi Sederhana dapat diimplementasikan dengan hanya sebuah Scenix SX28AC (Gambar 13).



Gambar 13. Virtual Peripheral Operator Binerisasi Sederhana dengan SX28AC

Proses yang dilakukan untuk Binerisasi Sederhana ini adalah membandingkan nilai register Rb (nilai Data) dan Rc (nilai Threshold) (Gambar 14).



Gambar 14. Diagram proses Binerisasi Sederhana pada SX28

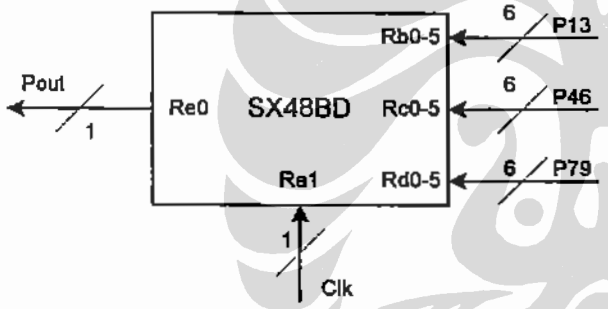
Adapun implementasi algoritma Binerisasi Sederhana dapat dilihat pada listing berikut ini.

```
1 Main
2       Clrb   Ra.3          ;1
3       Csbe   Rb,Rc         ;3,4(Jump)
4       Setb   Ra.3          ;1
5       Jump   Main          ;3

Total siklus        8
```

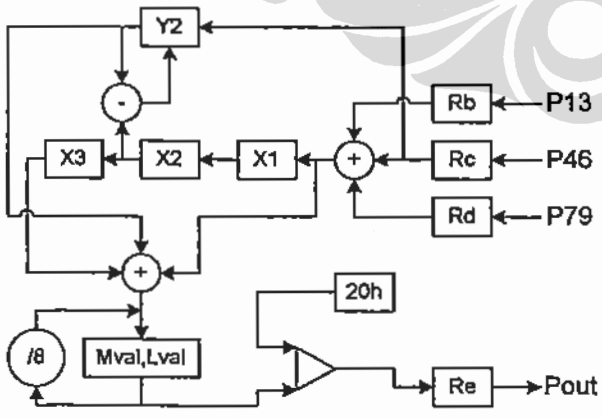
B2-4: Untuk membandingkan nilai Rb dan Rc. Bila Rb lebih kecil atau sama dengan Rc maka Ra.3=0 bila tidak maka Ra.3=1.

4.4.2. Implementasi Operator Binerisasi Lokal

Binerisasi Lokal diimplementasikan dengan sebuah Scenix SX48AC (
Gambar 15).



Gambar 15. Virtual Peripheral Operator Binerisasi Lokal dengan SX48BD



Gambar 16. Diagram proses Binerisasi Lokal pada SX48

Proses yang dilakukan untuk Binerisasi Lokal ini adalah hampir sama dengan filter intensitas. Hanya bedanya disini bahwa hasil lval dibandingkan dengan suatu harga Threshold misalnya dalam contoh ini Th=32 (20h) (Gambar 16).

Adapun implementasi algoritma Binerisasi Lokal dapat dilihat pada listing berikut ini.

```
1  Main
2       Jb     Ra.1,main     ;2,4(jump)
3       Clr    bufc          ;1
4       Clr    mval          ;1
5       Sub    y2,x2         ;2
6       Mov    w,Rb          ;1
7       Add    w,Rc          ;1
8       Add    w,Rd          ;1
9       Mov    x1,w          ;1
10      Add    w,y2          ;1
11      Rl     bufc          ;1
12      Add    w,x3          ;1
13      Rl     bufc          ;1
14      Mov    lval,w        ;1
15      Snz    bufc          ;1
16      Inc    mval          ;1
17      Rr     mval          ;1
18      Rr     lval          ;1
19      Rr     mval          ;1
20      Rr     lval          ;1
21      Rr     mval          ;1
22      Rr     lval          ;1
23      Clrb   Re.0          ;1
24      Csbe   lval,#$20     ;3,4(jump)
25      Setb   Re.0          ;1
26      Mov    x3,x2         ;2
27      Mov    x2,x1         ;2
28      Mov    y2,Rc         ;2
29      Jmp    Main          ;3

Total siklus        37
```

B1-B22: Keterangannya sama dengan filter intensitas.
B23-B25: Jika lval lebih kecil atau sama dengan 20h atau 32 desimal maka Re0=0 jika tidak maka Re0=1.
B26-B28: Sama dengan B24-B26 pada bagian filter intensitas.

4.5. Hasil dan Diskusi

Pada Tabel 1 disajikan data hasil implementasi 5 jenis operasi untuk merealisasi proses filtering dan binerisasi. Pada tabel ini diasumsikan bahwa chip yang digunakan bekerja pada frekuensi 75 Mhz dan Tsampling=500ns.

Total kataInstruksi yang dihasilkan berbeda untuk setiap operasi. Hal ini menyebabkan waktu eksekusi programnyapun berbeda. Waktu eksekusi ini akan

mempengaruhi waktu sampling, clock sistem dan resolusi. Waktu sampling, yang mempengaruhi clock sistem dan resolusi, tidak boleh berbeda untuk setiap operasi karena terkait masalah sinkronisasi. Waktu sampling ini ditentukan oleh waktu eksekusi terbesar dari kombinasi beberapa operasi pembentuk sistem.

Untuk membangun suatu sistem prapengolahan citra waktu-nyata lengkap beberapa kombinasi berikut dapat dilakukan:

A: FIFO+Op.Filter Rata-rata+Op. Binerisasi Sederhana.
B: FIFO+Op.Filter Intensitas+Op. Binerisasi Sederhana.
C: FIFO+Op.Filter Rata-rata+Op. Binerisasi Lokal.
D: FIFO+Op.Filter Intensitas+Op. Binerisasi Lokal.

Dengan resolusi 128 pixel (Tsampling=500ns), performansi waktu-nyata berhasil dicapai oleh ke-4 sistem: A, B, C dan D. Ini dimungkinkan karena jumlah kataInstruksi yang dihasilkan berada dibawah 37,5 kataInstruksi yang diijinkan.

Tabel 1: Data hasil implementasi 5 jenis operasi untuk merealisasi proses filtering dan binerisasi.

Operasi	Tot. Kata Instruksi	Chip	F chip (Mhz)	T instruksi (ns)	T eksekusi (ns)	T sampling (ns)	Waktu-nyata
FIFO	14	SX 28	75	13,3	173,3	500	Ya
Op. Filter Rata2	32	SX 48	75	13,3	426,7	500	Ya
Op. Filter Intensitas	34	SX 48	75	13,3	453,3	500	Ya
Op. Biner. Sederhana	8	SX 28	75	13,3	106,4	500	Ya
Op. Biner. Lokal	37	SX 48	75	13,3	492,1	500	Ya

Tabel 2 memberikan informasi akan kebutuhan chip Scenix untuk setiap jenis sistem dan setiap jenis operasi. Untuk sistem A dan B diperlukan 4 buah chip Scenix (3 buah SX28AC75 dan 1 buah SX48BD75). Untuk sistem C dan D diperlukan 6 chip Scenix (4 buah SX28AC75 dan 2 buah SX48BD75).

Hasil ini jauh lebih baik daripada disain terdahulu [8] yang menggunakan 10 buah chip SX28AC50 dan masih memiliki permasalahan waktu-nyata pada bagian operasi filter dan binerisasi.

Jumlah siklus yang didapatkan baik untuk FIFO maupun Operator Lokal berhasil mencapai target yang ditetapkan yaitu tidak melebihi 37,5 siklus. Ini berarti bahwa proses filtering maupun binerisasi dapat dilakukan secara waktu nyata dengan resolusi horizontal 128 pixel.

Jumlah sampling (resolusi) video yang dihasilkan saat ini masih relatif kecil yaitu 128. Kelihatannya akan sangat sulit bagi kita untuk menaikkan resolusi menjadi 256 dikarenakan rata-rata operasi filter dan binerisasi memerlukan kataInstruksi diatas 30 siklus. Walaupun kita gunakan chip Scenix dengan kecepatan instruksi 100 MIPS (100 Mhz, 10 ns/kataInstruksi) yang berarti Tsampling adalah 250ns.

Tabel 2: Kebutuhan chip Scenix untuk proses waktu-nyata dari setiap sistem dengan resolusi 128 pixel.

Sistem	Jenis operasi	SX28AC 75	SX48BD 75
A	FIFO	2	-
	Op. Filter. Rata-rata	-	1
	Op. Binerisasi Sederhana	1	-
B	FIFO	2	-
	Op. Filter Intensitas	-	1
	Op. Binerisasi Sederhana	1	-
C	FIFO	4	-
	Op. Filter Rata-rata	-	1
	Op. Binerisasi Lokal	-	1
D	FIFO	4	-
	Op. Filter Intensitas	-	1
	Op. Binerisasi Lokal	-	1

Agar bisa mencapai resolusi 256 pixel solusi yang mungkin diterapkan pertama-tama adalah mengupayakan algoritma lebih efisien lagi kemudian solusi kedua adalah mencari kemungkinan memecah perhitungan menjadi beberapa bagian dengan konsekuensi penambahan jumlah chip.

5. KESIMPULAN

Pada paper ini, telah dibahas perancangan dan implementasi hardware secara sofware dengan konsep *Virtual Peripheral* untuk prapengolahan citra waktu-nyata filtering dan binerisasi.

Jumlah kataInstruksi yang didapatkan baik untuk FIFO maupun Operator Lokal berhasil mencapai target yang ditetapkan yaitu tidak melebihi 37,5 siklus. Ini berarti bahwa proses filtering maupun binerisasi dapat dilakukan secara waktu nyata 30 frame/detik dengan resolusi horizontal 128 pixel.

Sistem lengkap prapengolahan citra filtering dan binerisasi dengan operasi: FIFO, Filter Rata-rata (atau Filter Intensitas) dan Binerisasi Sederhana telah berhasil diimplementasikan ke dalam 4 buah chip Scenix (3 buah SX28AC75 dan 1 buah SX48BD75). Sedangkan bila operasi Binerisasi Sederhananya digantikan dengan Binerisasi Lokal, memerlukan 6 chip Scenix (4 buah SX28AC75 dan 2 buah SX48BD75).

Jumlah resolusi (sampling) video yang dihasilkan saat ini adalah 128 dan ini masih relatif kecil. Kelihatannya akan sangat sulit bagi kita untuk menaikkan resolusi menjadi 256 dikarenakan rata-rata operasi filter dan binerisasi memerlukan kataInstruksi diatas 30 siklus, walaupun digunakan chip Scenix dengan kemampuan 100 MIPS (100 Mhz, 10 ns/kataInstruksi). Solusi yang terbaik untuk itu adalah dengan mengupayakan algoritma lebih efisien dan/atau dengan mencari kemungkinan memecah perhitungan menjadi beberapa bagian dengan konsekuensi penambahan jumlah chip.

Walaupun aspek waktu-nyata hanya dapat terpenuhi pada resolusi yang rendah namun teknik-teknik perancangan dengan *Virtual Peripheral* yang diperkenalkan ini telah membuka jalan bagi perancangan hardware secara software terutama bagi mereka yang banyak bereksperimen dengan hardware pengolahan citra waktu-nyata yang relatif mahal biaya pengembangannya.

REFERENSI

[1] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall International Editions, 1989.

[2] H.R. Myler, dan A.R. Weeks, *Computer Imaging Recipes in C*, PTR Prentice-Hall, New Jersey, 1993.

[3] A. Nalwan, *Pengolahan Gambar secara Digital*, Elex Media Computindo, Jakarta, 1997.

[4] SX-Key/Blitz Development System Manual Version 1.1, Parallax Inc., sx-key_manual_v1_1.pdf, http://www.parallaxinc.com.

[5] High-Performance 8-Bit Micro-controllers with EE/Flash Program Memory and In-System Programming Capability, 11 Feb 1999, sx_datasheet.pdf, http://www.ubicom.com.

[6] Scenix-SX Virtual Peripheral Library Guide, library.pdf, http://www.ubicom.com.

[7] Scenix-User's Manual Virtual Peripheral ™ Methodology & Modules, vp_usermanual.pdf, http://www.ubicom.com.

[8] E. Mozef, *Perancangan Prapengolahan Citra Filtering dan Binerisasi Secara Waktu-Nyata Dengan Virtual Peripheral*, Prosiding SNKK3, vol.3, no.1, Jakarta, Agustus 2002, pp. 33-38.