VHDL IMPLEMENTATION FOR JAWI CHARACTER RECOGNITION

IMAGE ENHANCEMENT

NAMA : NORNITA ADILA BT MUSTAFA

NO. MATRIK : WEK 000344

PENASIHAT : ENCIK ZAIDI RAZAK

MODERATOR : ENCIK MOHD YAMANI IDNA IDRIS

ABSTRAK

Kewujudan tulisan Jawi sejak dahulu lagi telah menghasilkan pelbagai karya penting dalam perkembangan Islam. Sebagai satu bahasa pengantara, tulisan Jawi dianggap penting dalam bidang penulisan. Tidak terhad hanya kepada buku-buku, dokumen-dokumen dan manuskrip-manuskrip yang ditulis dalam tulisan Jawi, malah penggunaanya amat meluas dan meliputi keseluruhan bidang.

Menyedari kepentingannya, timbul minat saya untuk mengkaji dengan lebih mendalam bagaimana komputer boleh mengenali dan memahami tulisan ini tanpa menggunakan tenaga manusia berdasarkan kepada beberapa kajian yang sedia ada yang telah dijalankan oleh mereka yang pakar dalam teknologi komputer digital.

Secara keseluruhan, kajian ini akan membantu memudahkan lagi pengecaman tulisan tangan Jawi yang dilakukan secara digital oleh komputer. Komputer akan membantu dalam pengecaman perkataan Jawi yang sukar untuk dibaca contohnya tulisan tangan Jawi yang mempunyai pelbagai gaya penulisan, manuskrip-manuskrip lama dan lain-lain lagi. Proses peningkatan kualiti imej (*Image Enhancement*) akan meningkatkan lagi kualiti imej yang diinput dengan menghasilkan output yang lebih jelas dan bermutu. Proses peningkatan kualiti akan menghasilkan imej yang lebih terang dan jelas dan langkah pengasingan perkataan Jawi kepada aksara-aksara tunggal (segmentasi) akan dilakukan untuk tujuan proses pengecaman (*recognition*).

PENGHARGAAN

Alhamdulillah, syukur kehadrat Illahi dan segala puji-pujian kepada Allah S.W.T kerana dengan limpah dan kurnia-Nya dapat saya menyiapkan kajian bertajuk "VHDL Implementation for Jawi Character Recognition" bagi memenuhi syarat untuk subjek Projek Ilmiah Tahap Akhir II (WXES 3182) dengan sempurna dan lancar. Pelbagai rintangan dan masalah yang dihadapi sepanjang pelaksanaan projek ini tetapi segalanya dapat diatasi dan diselesaikan dengan adanya usaha yang berterusan tanpa mengenal erti jemu serta bantuan dari semua pihak yang terlibat secara langsung atau tidak langsung.

Kejayaan yang telah dicapai ini telah mendapat bantuan dan tunjuk ajar dari banyak pihak. Saya ingin mengucapkan jutaan terima kasih dan setinggi-tinggi penghargaan kepada semua pihak terutamanya penasihat saya, Encik Zaidi bin Razak dan juga moderator saya, Encik Yamani Idna Idris yang telah banyak memberikan banyak tunjuk ajar, bantuan serta dorongan dalam menyiapkan projek ini. Tidak lupa juga kepada rakan-rakan sepasukan iaitu Nur Hasinah Hassan, Nurainil Mustapa, Suzylarina Samsudin, Hazrin Ahmad dan Razman Abdul Ghani yang juga telah banyak membantu memberikan tunjuk ajar, dorongan, kerjasama serta komitmen yang tidak berbelah bagi terhadap pelaksanaan projek ini.

Akhir sekali, saya amat berharap agar kajian ini akan memberi manfaat kepada semua dan menjadi panduan kepada generasi yang akan datang yang akan meneruskan usaha demi perkembangan tulisan Jawi dimasa hadapan.

Wassalam.

ISI KANDUNGAN **M/Surat**

ISI KANDUNGAN **M/Surat**

ISI KANDUNGAN **M/Surat**

SENARAI RAJAH

BAB I

PENGENALAN

1.1 PENGENALAN

Kajian terhadap penerimaan aksara dan objek-objek atau bentuk-bentuk tulisan tangan oleh komputer atau 'character recognition' telah lama dilakukan. Kepelbagaian bahasa dan tulisan di dunia ini telah menarik perhatian banyak pengkaji untuk mengkaji bagaimana komputer boleh memahami dan mengenali tulisan tangan tanpa menggunakan tenaga manusia.

Kewujudan tulisan Jawi sejak dahulu lagi telah menghasilkan pelbagai karya-karya yang penting berkenaan perkembangan Islam. Buku-buku dan manuskrip-manuskrip lama menjadi pengantara para pengkaji tulisan Jawi untuk mendalami keistimewaan tulisan ini. Menyedari hakikat kepentingan tulisan Jawi, kajian mengenai pengecaman aksara Jawi menggunakan komputer banyak dijalankan. Masalah mengenai kerumitan mengecam tulisan Jawi pada manuskrip lama atau dokumen tulisan tangan akan dapat diatasi.

Pelbagai proses yang terlibat untuk mendapat hasil yang memuaskan. Secara amnya, imej tulisan tangan diimbas menggunakan mesin pengimbas (scanner). Imej yang diinput akan dibaca dan mengalami beberapa proses penting sebelum ianya dipecahkan kepada aksara-aksara tunggal (segmentasi) sebelum proses pengecaman aksara dilakukan. Diantara proses-proses yang penting, salah satunya merupakan proses peningkatan kualiti imej (Image Enhancement Process).

Peningkatan Kualiti Imej (Quality Image Enhancement) diimplementasikan dalam Pengecaman Imej Tulisan Jawi ini bertujuan untuk memperoleh hasil (output) yang lebih terang dan jelas. Peningkatan kualiti imej dapat dihuraikan sebagai satu proses yang menggunakan teknik-teknik tertentu untuk memudahkan pembangunan penyelesaian kepada masalah pengimejan komputer. Ia juga boleh dikatakan sebagai satu cara untuk membuang 'kebisingan' (noise) yang terdapat pada imej yang diinput.

Teknik-teknik yang digunakan untuk peningkatan kualiti imej ini adalah untuk menekankan dan menajamkan lagi ciri-ciri imej untuk tujuan paparan dan analisis. Antara teknik-teknik yang terlibat ialah mengubah tahap kecerahan (Brightness Adjustment), perlicinan tepian (Edge Smoothing), Thresholding, Thinning dan juga segmentasi (Segmentation). Proses-proses yang terlibat akan dijalankan menggunakan perisian VHDL.

1.2 OBJEKTIF PROJEK

Objektif-objektif yang digariskan dalam projek ini ialah :

1. Mengecam imej tulisan tangan Jawi dengan memaparkan output sebagai aksara Jawi tunggal.
2. Mendigitalkan imej tulisan Jawi dan seterusnya mengurangkan gangguan 'kebisingan' serta meningkatkan kualiti imej .
3. Menjalankan proses pengsegmenan dan pemadanan untuk tujuan pengecaman aksara.
4. Membangunkan pangkalan pengetahuan untuk menyimpan *chain code* dan imej digital bagi imej tertentu.
5. Mengenal dan mengecam aksara tunggal Jawi menggunakan perisian VHDL.
6. Menjalankan proses peningkatan kualiti imej dengan menggunakan perkakasan VHDL.
7. Lebih memahami tentang pengimplementasian menggunakan bahasa pengaturcaraan VHDL dan menulis program untuk perkakasan.
8. Pelaksanaan VHDL dilakukan berdasarkan kesinambungan daripada proses menggunakan perisian Matlab 6.1.

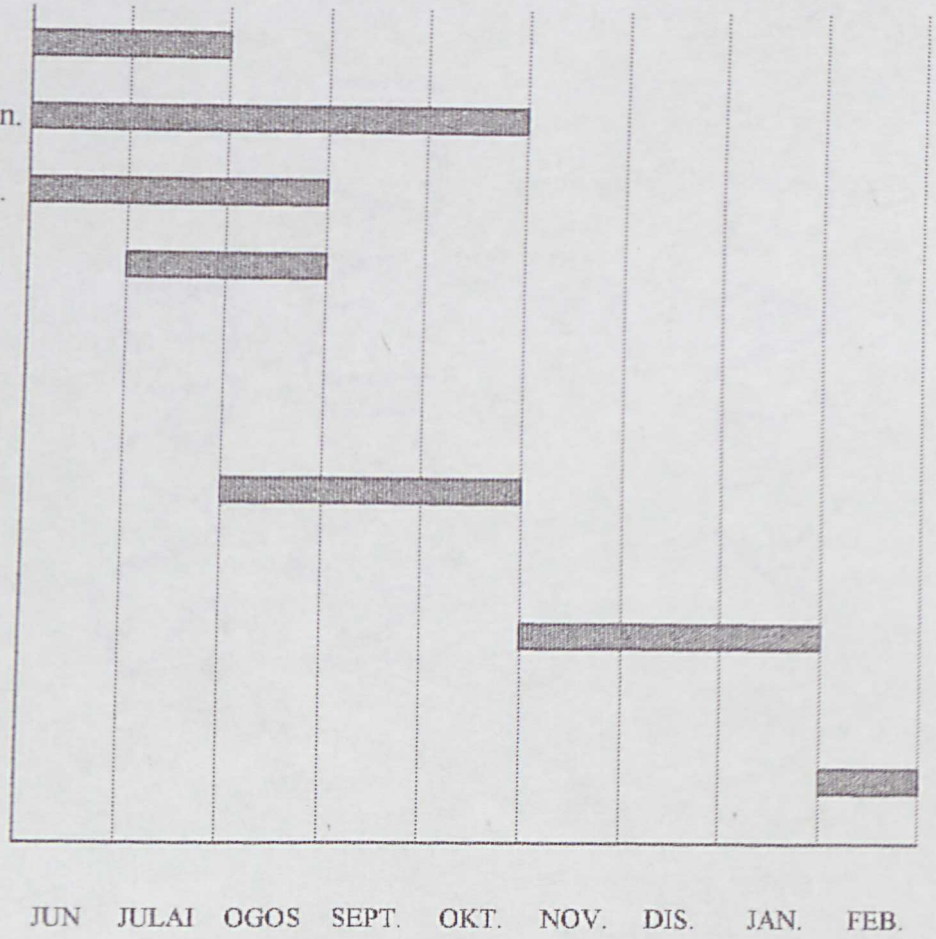
1.3 SKOP PROJEK

1. Imej yang akan diproses adalah bersaiz 8 piksel x 8 piksel .
2. Imej yang terhasil selepas mengalami proses peningkatan kualiti imej ialah imej yang lebih terang dan berkualiti.
3. Imej yang diinput merupakan satu patah perkataan Jawi sahaja. Jika proses peningkatan kualiti imej berjaya dijalankan ke atas patah-patah perkataan tersebut, proses peningkatan kualiti akan diteruskan ke atas baris-baris ayat pula.
4. Proses segmentasi akan memecahkan aksara Jawi yang bersambung dan yang tidak bersambung (aksara-aksara yang membentuk patah-patah perkataan) menjadi aksara-aksara Jawi tunggal.

1.4 GRAF PERANCANGAN PERLAKSANAAN PROJEK

AKTIVITI:

1. Mengkaji dan memahami kehendak projek.
2. Mempelajari perisian VHDL yang akan digunakan.
3. Memahami konsep pemprosesan imej secara teori.
4. Kajian terhadap metodologi- metodologi yang ada dalam pemproses imej digital (spesifik kepada proses peningkatan kualiti imej).
5. Memilih dan memahami metodologi yang akan digunakan dalam implementasi projek.
6. Membina sebuah program dalam bahasa pengaturcaraan VHDL.
7. Membuat ujian terhadap sistem yang telah dibina.



BAB 2

KAJIAN LITERASI

2.1 KAJIAN LITERASI

Kajian literasi merupakan kajian permasalahan yang dijalankan sebelum peringkat implementasi sesuatu projek. Ianya meliputi kajian serta analisa ke atas sistem-sistem terdahulu, kajian ke atas teknik-teknik yang akan digunakan serta kajian ke atas domain bagi projek tersebut. Kajian yang dijalankan meliputi hasil-hasil kajian oleh para pengkaji sistem pengecaman komputer digital yang telah berjaya menghasilkan sistem menggunakan teknik-teknik yang pelbagai.

2.2 KAJIAN YANG SEDIA ADA

Untuk memahami dengan lebih lanjut mengenai projek yang sedang dijalankan, beberapa kajian terhadap kajian terdahulu yang mempunyai persamaan dengan projek Pengecaman Tulisan Jawi dilakukan. Antara kajian yang sedia ada mengenai pengecaman aksara ialah kajian ke atas tulisan bahasa Cina, tulisan bahasa Jepun, tulisan bahasa Arab dan banyak lagi. Penelitian terhadap kaedah pengecaman yang digunakan oleh para pengkaji sebelum ini penting bagi melihat kaedah yang sesuai bagi kajian terhadap tulisan Jawi ini, memandangkan bentuk tulisan Jawi adalah unik. Dibawah merupakan beberapa kajian yang telah saya jalankan untuk memahami kajian yang sedia ada serta memenuhi keperluan kajian literasi ini .

2.2.1 KAJIAN 1

Tajuk : Off-Line Arabic Character Recognition

Kajian Oleh : M. S Khorsheed

(Computer Laboratory, University of Cambridge, Cambridge, UK)

Metodologi yang digunakan :

Untuk proses peningkatan kualiti imej, bagi sub-proses *Thinning*, metodologi yang digunakan ialah *Skeletonisation*. Satu algoritma baru diperkenalkan untuk tujuan *skeletonisation* iaitu pengasingan aksara Jawi berdasarkan kepada perangkaian imej aksara. Skeleton kemudian dihasilkan selepas memperoleh matrik bersebelahan bagi kelompok (*clusters*) yang berlainan. Akhirnya, bahagian-bahagian yang tidak penting atau tidak diperlukan pada skeleton akan dibuang. Bagi segmentasi pula, terdapat 2 sistem pengecaman teks bahasa Arab iaitu *Segmentation-based Systems* dan *Segmentation-Free Systems*. Bagi *Segmentation-based Systems*, terdapat 4 kategori iaitu *Isolated/pre-segmented characters, Segmenting a word into characters, Segmenting a word into primitives* dan *Integration of recognition and segmentation.*

1. *Isolated/pre-segmented characters*

- para pengkaji boleh mengecam angka-angka, aksara terasing atau menganggap bahawa aksara tersebut terhasil dari algoritma segmentasi yang boleh dipercayai.

2. *Segmenting a word into characters*

- Pendekatan pertama yang digunakan untuk segmentasi. Sistem tersebut akan cuba untuk segmenkan satu perkataan kepada aksara-aksara tunggal, dan kemudian mengecam setiap aksara secara berasingan.

3. *Segmenting a word into primitives*

- proses ini akan segmenkan sub-perkataan atau menghubungkan komponen kepada simbol-simbol dimana setiap simbol mewakili satu aksara, satu sambungan atau kemungkinan satu pecahan bagi satu aksara.

4. *Integration of recognition and segmentation*

- Segmentasi diimplementasikan selepas proses pengecaman. Pendekatannya ialah dengan mengimbas (scan) perkataan bermula dari sebelah kanan. Sistem tidak selalunya boleh mengecam semua aksara, yang mana ianya berkemungkinan bahawa semua aksara bagi perkataan tersebut yang telah berjaya dikenalpasti mungkin tidak akan diproses.

Bagi *Segmentation-Free Systems* pula, sistem tersebut akan cuba untuk mengecam keseluruhan perkataan tanpa mengsegmenkan dan mengecam aksara-aksara atau primitif-primitif secara individu.

Hasil yang diperolehi :

Kebanyakkan daripada algoritma *thinning* beroperasi dengan menjalurkan secara berulang kontur bagi piksel-piksel. Masalah utama yang berkait dengan pendekatan ini ialah kelajuan pengiraan adalah sangat rendah. Satu alternatif penyelesaian ialah dengan mengimplemenkan algoritma berasaskan vektor yang mana ianya beroperasi secara terus sepanjang pengkodan bagi imej binari.

2.2.2 KAJIAN 2

Tajuk : Segmentation and Recognition of Arabic Characters by Structural Classification.

Kajian Oleh : B.M.F. Bushofa, M. Spann

(School of Electronic and Electrical Engineering, University of Birmingham, Birmingham, UK)

Metodologi yang digunakan :

Bagi perlicinan imej, kaedah perlicinan berasaskan statistik digunakan untuk mengurangkan 'kebisingan' dengan cara menghapuskan bahagian-bahagian yang tidak diperlukan serta mengisi lubang-lubang yang kosong. Satu imej binari bagi teks bahasa Arab, algoritma akan mengubah setiap piksel mengikut kepada nilai pemula dan semua nilai yang berjiranan.

Bagi segmentasi pula, algoritma pengsegmenan akan mencari ketepatan suatu sudut yang dibentuk oleh pertemuan diantara 2 aksara. Sudut tersebut berlaku di *baseline*, yang mana profil mendatar mengandungi bilangan maksimum bagi piksel-piksel hitam. Setelah *baseline* ditemui, algoritma akan diteruskan dengan mengimbas imej bermula dari kanan ke kiri melalui *baseline* tersebut dengan menggunakan tetingkap bersaiz 7X7 dan memeriksa kejiranan bagi piksel tengah. Piksel tengah digunakan sebagai piksel untuk proses segmentasi. Berikut merupakan langkah-langkah proses segmentasi yang dijalankan :

Rajah 2 (a)

X	X	X	Y	0	0	0
1	X	X	Y	0	0	0
X	1	X	0	0	0	0
X	1	1	P	0	0	0
1	1	1	1	1	1	1
X	X	X	X	X	X	X
X	X	X	X	X	X	X

Rajah 2 (b)

1	1	1
1	P	0
1	0	0

Rajah 2(a) menunjukkan tetingkap 7X7 yang digunakan untuk mengesan kedudukan dimana 2 aksara perlu diasingkan. Rajah 2(b) pula merupakan tetingkat 3X3 yang digunakan untuk mengasingkan aksara "Hha" .(X:don't care, Y:1 untuk aksara "Ain" ditengah atau dihujung dan 0)

Langkah 1 :

P ditentukan sebagai titik dimana 2 aksara tersebut perlu diasingkan. Kesemua piksel yang berada diatas P hingga ke tinggi maksima diperiksa. Jika tiada piksel hitam ditemui, maka imej akan dibahagikan pada titik P. Jika terdapat piksel, imej tersebut dipeiksa lebih teliti untuk melihat samada aksara yang berada disebelah kiri merupakan "Hha" dihujung. Ini dilakukan dengan cara menguji piksel hitam pertama yang berada diatas P menggunakan tetingkap 3X3. Jika syarat keadaan tersebut dipenuhi, maka penyambungan antara aksara tersebut akan dipadamkan.

Langkah 2 :

Jika syarat di atas tidak ditemui, piksel bertanda "Y" akan diperiksa (Rajah(a)). Jika kedua-duanya hitam, aksara yang berada disebelah kiri merupakan "Ain" ditengah atau dihujung. Kemudian, segmentasi akan diimplemen pada bahagian tengah pecahan yang menyambungkan aksara tersebut dengan aksara yang berada di sebelah kanan.

Langkah 3 :

Kedua-dua langkah di atas diulangi sehingga kesemua baris teks diimbas.

Langkah 4 :

Algoritma diikuti dengan kontur digunakan untuk mengasingkan aksara-aksara yang tidak bersambung dan mungkin bertindih antara satu sama lain.

Hasil yang diperoleh :

Dalam kajian ini, patah-patah perkataan akan disegmentasikan kepada aksara-aksara tunggal dan bentuk-bentuk kedua (*secondaries*) akan dibuang menggunakan algoritma yang dibina. Maklumat mengenai bentuk-bentuk kedua ini seperti nombor, kedudukan dan jenis direkodkan dan digunakan pada peringkat terakhir proses pengecaman. Terdapat masalah yang timbul jika segmentasi yang dilakukan berdasarkan kepada *baseline*. Masalah tersebut ialah berlaku pertindihan (*overlap*) antara aksara-aksara bersebelahan yang berlaku secara semulajadi dan juga sambungan antara 2 aksara selalunya pendek. Oleh itu penempatan titik adalah sukar serta penentuan panjang yang sesuai untuk sesuatu aksara tunggal juga sukar untuk ditentukan.

2.2.3 KAJIAN 3

Tajuk : Off-Line Arabic Character Recognition: The State of The Art.

Kajian Oleh : Adnan Amin (School of Computer Science and Engineering, University of New South Wales, Australia).

Metodologi yang digunakan :

Segmentasi merupakan satu langkah penting untuk pengecaman aksara teks bahasa Arab dan jika terdapat kesilapan dalam mengsegmenkan bentuk asas aksara-aksara tersebut, ia akan menghasilkan perwakilan komponen aksara yang berbeza. Oleh itu, bagi segmentasi dalam kajian ini, 2 teknik telah digunakan iaitu pengsegmenan secara jelas (*explicit*) dan pengsegmenan tersirat (*implicit*).

1. Pengsegmenan secara jelas (straight segmentation): Dalam teknik ini, patah-patah perkataan disegmenkan terus menjadi huruf-huruf. Segmentasi jenis ini biasanya direka dengan syarat bahawa ia boleh mengenalpasti kesemua titik segmentasi aksara-aksara tersebut.
2. Pengsegmenan tersirat : Perkataan akan disegmenkan secara luaran ke dalam huruf pseudo yang mana ianya kemudian dikenalpasti secara individu.

Sebagai alternatif, kaedah segmentasi juga dilakukan dengan menyurih kontur luaran bagi perkataan dan mengira jarak diantara titik ekstrim bagi persilangan kontur dengan menggunakan garisan menegak (*vertical line*). Segmentasi ini adalah berdasarkan kepada imbasan mendatar dari kanan ke kiri bagi kontur tertutup.

Hasil yang diperolehi :

Pendekatan di atas sangat bergantung kepada nilai *threshold* yang ditetapkan, yang berkait rapat dengan ketebalan aksara. Masalah yang dihadapi ialah pendekatan ini kurang berkesan terhadap imej yang condong atau senget. Tulisan tangan Jawi yang pelbagai rupa dan bentuk atau tulisan yang bercorak *(decorative)* menyukarkan penentuan sempadan sebenar sesuatu aksara. Ini akan menyukarkan proses segmentasi. Selain itu, kepanjangan dan ketebalan tulisan tangan Jawi yang berbeza juga menyukarkan penentuan kepanjangan yang sesuai untuk proses segmentasi kepada aksara tunggal.

BAB 3

METODOLOGI

3.0 METODOLOGI YANG DIGUNAKAN

Metodologi yang digunakan merujuk kepada huraian yang mendalam tentang kaedah penyelidikan serta teknik yang digunakan untuk menyelesaikan masalah projek yang timbul. Metodologi yang digunakan dalam kajian ini hanya berkait rapat dengan proses peningkatan kualiti imej *(image enhancement)*.

3. 1 PENINGKATAN KUALITI IMEJ

Proses peningkatan kualiti imej dilaksanakan untuk menjadikan imej kelihatan lebih berkualiti dan menepati ciri-ciri output yang dikehendaki. Teknik peningkatan kualiti imej digunakan untuk menekankan dan menajamkan lagi ciri-ciri imej untuk tujuan paparan (*display*) dan analisis. Juga merupakan satu proses yang menggunakan teknik-teknik tertentu untuk memudahkan pembangunan penyelesaian kepada masalah pengimejan komputer. Proses pada peringkat ini boleh membuang 'kebisingan' *(noise)* dari imej. Dalam projek Pengimplementasian VHDL bagi Pengecaman Aksara Jawi ini, terdapat 5 sub-proses yang terlibat untuk peningkatan kualiti imej. Sub-proses – sub-proses tersebut ialah Mengubah Tahap Kecerahan (*Bright Adjustment*), Perlicinan Tepian (*Edge Smoothing*), *Thresholding*, *Thinning* dan Segmentasi. Algoritma-algoritma yang akan dibina akan ditulis menggunakan bahasa pengaturcaraan perkakasan VHDL.

3.1.1 Mengubah Tahap Kecerahan (Brightness Adjustment)

Dalam peringkat ini, imej yang melalui proses pengubahan tahap kecerahan akan mengalami perubahan tahap kekaburan yang ada pada imej yang diinput. Imej asal yang gelap akan menjadi lebih terang (mudah dilihat) dan jelas. Teknik yang akan digunakan ialah *Histogram Sliding* . Teknik ini digunakan untuk menjadikan sesuatu imej samada lebih gelap atau lebih terang tetapi tetap mengekalkan hubungan diantara nilai-nilai tahap kelabu (gray-level). Ini dapat dicapai dengan cara menambah atau mengurangkan nombor yang telah ditetapkan daripada kesemua nilai-nilai tahap kelabu tersebut.

Contoh perwakilan oleh imej bergambar :

Imej asal

Imej selepas proses pencerahan

Rajah 3 (a) Proses 'Brightness Adjustment'

Teknik *Histogram Sliding* ini boleh dilakukan dengan menambah atau mengurangkan satu nombor tetap daripada semua nilai tahap kelabu (gray-level). Persamaan adalah seperti berikut :

$$Slide\ (I(r,c)) = I(r,c) + OFFSET$$

dimana OFFSET merupakan nilai anjakan histogram.

Bagi nilai positif bagi OFFSET, ianya akan meningkatkan kecerahan keseluruhan imej, manakala nilai negatif bagi OFFSET, ianya akan membentuk imej yang lebih gelap.

3.1.2 Perlicinan Tepian (Edge Smoothing)

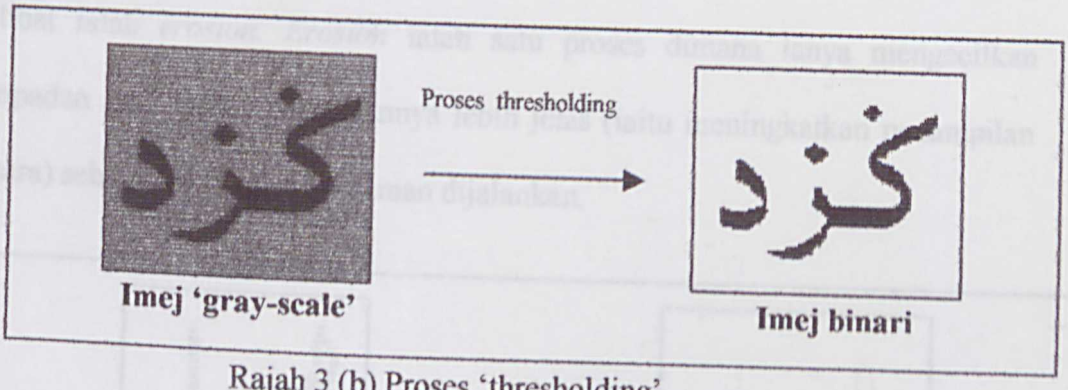
Proses ini juga merupakan salah satu kaedah untuk membuang kebisingan (noise). Imej yang belum mengalami proses perlicinan tepian ini, iaitu imej input, merupakan imej yang mempunyai tepian *(edge)* yang tidak rata, tidak licin, tidak lurus atau kurang kemas. Kemudian, proses tersebut akan menghasilkan imej output imej yang mempunyai tepian *(edge)* yang lebih licin, jelas dan terang supaya keberkesanan proses seterusnya dapat dilihat. Selain itu, proses ini juga dapat menampakkan kesan-kesan khas yang terdapat pada aksara-aksara Jawi seperti ukiran, lengkuk, cabang dan lain-lain.

Teknik yang akan digunakan ialah Median Filter 3x3. 3x3 merupakan saiz 'mask'. Semakin tinggi saiz 'mask' tersebut, imej yang terhasil akan lebih kabur (*blur*). Median Filter ini telah pun digunakan pada proses pembuangan 'kebisingan' (*noise reduction*).Namun, disebabkan Median Filter ini boleh juga digunakan untuk proses perlicinan tepian ini, saya telah memilih untuk menggunakannya memandangkan saya lebih memahami tentang kaedah ini berbanding kaedah-kaedah lain.

3.1.3 Thresholding

Teknik *threshold* merupakan satu proses untuk menukarkan imej tahap kelabu (*gray-scale*) dan imej berwarna (RGB) kepada imej perduaan (*binary*). Nilai *threhold* yang telah ditetapkan ialah 180. Setiap imej yang melalui proses *thresholding*, mana-mana piksel yang nilainya berada diatas daripada nilai *threshold*, piksel-piksel tersebut akan bertukar menjadi warna putih, mewakili nilai 1 dalam imej binari. Bagi piksel-piksel yang nilainya berada dibawah daripada nilai *threshold*, piksel-piksel tersebut akan bertukar kepada warna hitam dan mewakili nilai 0 bagi imej binari.

Contoh perwakilan oleh gambarajah aksara Jawi :



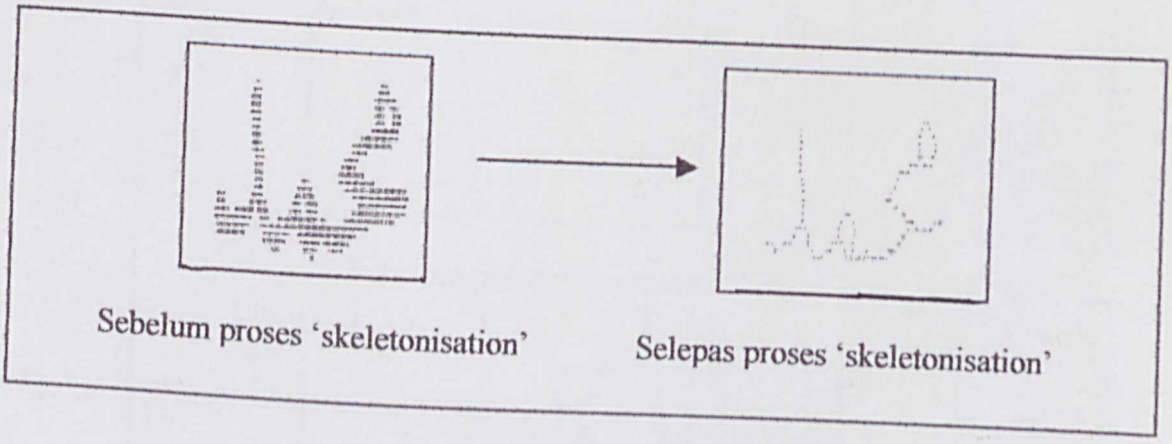
Rajah 3 (b) Proses 'thresholding'

Zero Padding

Saiz sebenar imej yang diperlukan ialah 512 X 512 bilangan piksel. Perwakilan matrik digunakan untuk menunjukkan piksel yang mengandungi imej. Satu teknik yang dinamakan *Zero Padding* diaplikasikan pada sub-proses ini. *Zero Padding* ini akan mengisi piksel-piksel yang tidak mempunyai imej, yang ditunjukkan oleh perwakilan matrik, dengan nilai 0 . Ia bertujuan hanya untuk tidak membiarkan matrik 512 X 512 kosong tanpa sebarang nombor/nilai.

3.1.4 Thinning

Teknik thinning digunakan untuk menghasilkan imej yang tajam dan nipis. Imej yang terhasil ialah imej berasaskan garis lurus. Hasil dari proses ini akan digunakan untuk proses pengsegmenan. 2 teknik yang sering digunakan dalam proses *thinning* ini ialah *skeletonisation* dan *erosion*. Bagi proses *skeletonisation*, imej yang terhasil dipanggil skeleton. Proses *skeletonisation* dapat dihuraikan sebagai satu proses untuk mengurangkan ketebalan imej binari menjadi imej kerangka (skeletel). Semasa proses penipisan dilakukan, bentuk-bentuk asal dan perhubungan antara aksara-aksara masih lagi dipelihara. Satu lagi proses yang terlibat ialah *erosion*. *Erosion* ialah satu proses dimana ianya mengecilkan sempadan imej bagi menjadikannya lebih jelas (iaitu meningkatkan penampilan aksara) sebelum proses pengecaman dijalankan.



Rajah 3 (c) Proses 'thinning'

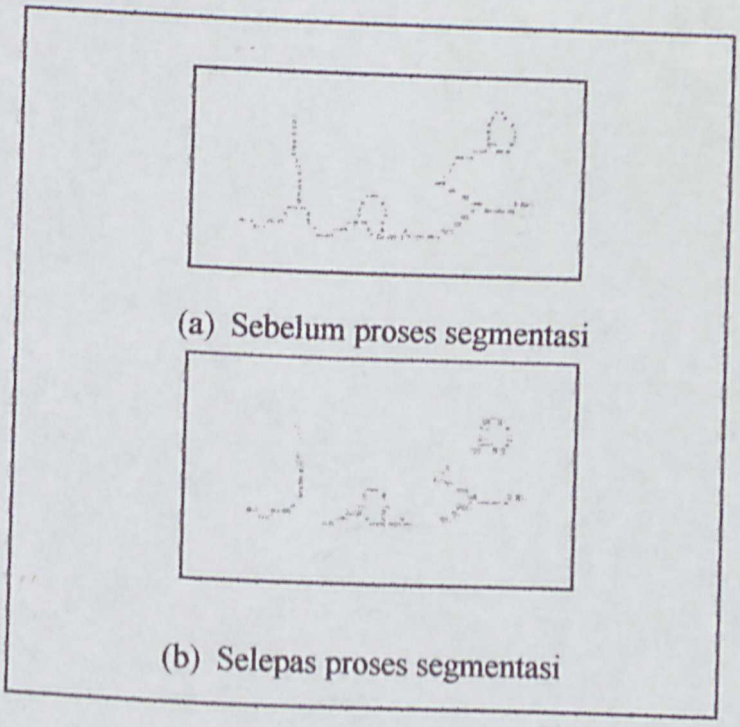
Selepas proses *thinning* imej binari, struktur imej tersebut boleh direkodkan dengan menyurih skeleton. Walaubagaimanapun, proses *thinning* mungkin mengubah bentuk asal sesuatu aksara, yang mana ianya menjadi sukar untuk dikenalpasti. Oleh itu, proses ini harus dilakukan dengan berhati-hati agar tidak berlaku sebarang masalah.

Algoritma *thinning* boleh dikelaskan kepada selari (*parallel*) dan jujukan (*sequential).* Algoritma selari beroperasi ke atas semua piksel secara serentak. Algoritma jujukan pula memeriksa piksel-piksel dan mengubahnya, bergantung kepada keputusan proses yang terdahulu. Pendekatan kedua-dua algoritma ini ialah membuang sempadan piksel-piksel aksara.

3.1.5 Segmentasi (Segmentation)

Proses ini dijalankan dengan cara mengasingkan aksara-aksara Jawi yang bersambung dan tidak bersambung. Pengasingan berlaku apabila terdapat ruang kosong antara aksara-aksara tersebut (yang tidak bersambung). Bagi aksara-aksara Jawi yang bersambung, pemecahan aksara akan dilakukan untuk menjadikannya aksara-aksara tunggal. Sebelum proses pengsegmenan boleh dijalankan, imej mestilah dalam bentuk binari. Teknik yang digunakan untuk proses segmentasi ialah *edge detection*. Fungsi ini dapat mengesan sempadan bagi sesuatu imej. Kaedah Sobel dipilih untuk membolehkan sempadan dikenalpasti dan pengasingan aksara Jawi yang bersambung dan tidak bersambung dapat dilakukan.

Contoh perwakilan oleh satu patah perkataan Jawi :



(a) Sebelum proses segmentasi

(b) Selepas proses segmentasi

Rajah 3 (d)

BAB 4

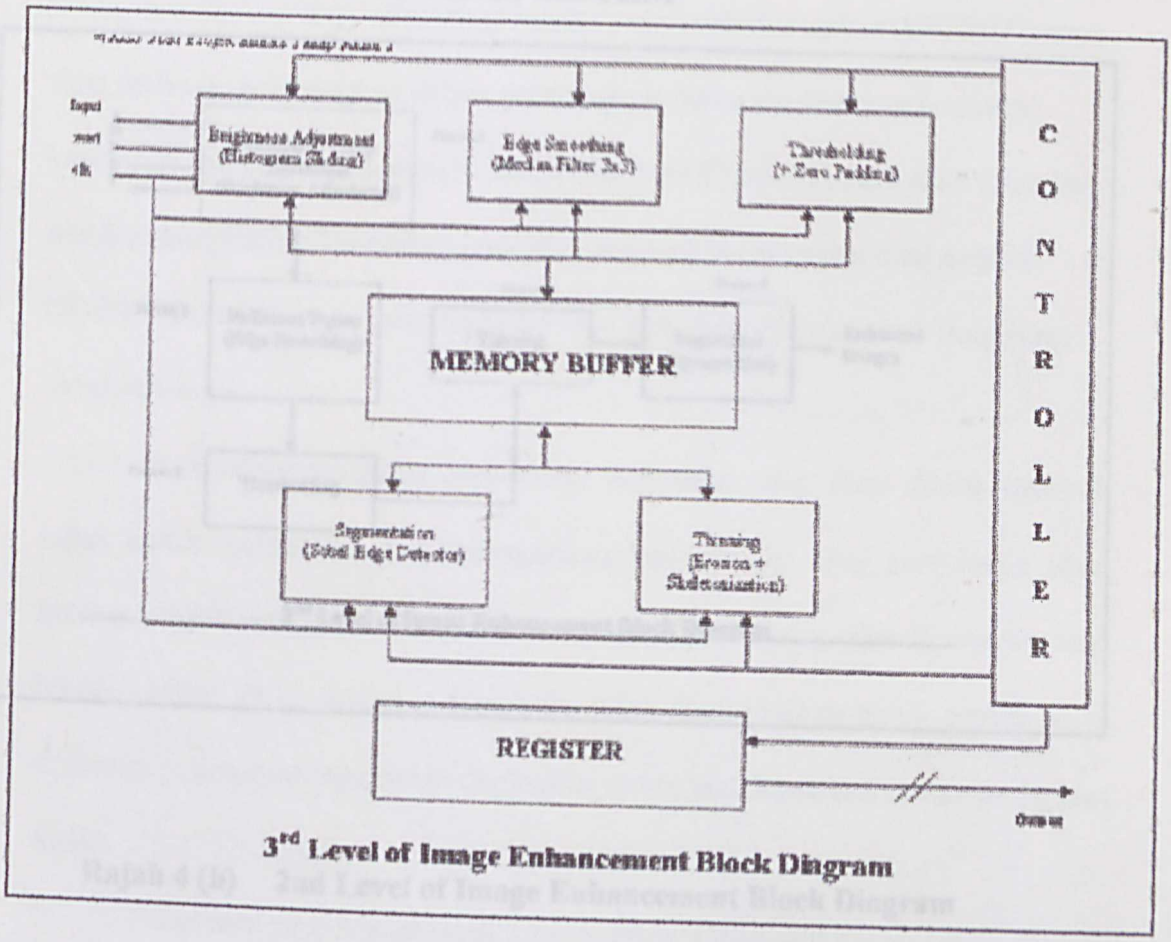
REKABENTUK SISTEM

4.0 REKABENTUK SISTEM

Rekabentuk sistem merupakan huraian yang melibatkan proses percantuman kesemua bahagian-bahagian tertentu kepada sebuah sistem yang mengandungi fungsi-fungsi yang harus dilaksanakan oleh sistem. Dalam kajian ini, rekabentuk modul-modul yang terlibat, yang membentuk sistem akan diterangkan secara mendalam. Terdapat 5 modul atau subproses yang membentuk keseluruhan proses peningkatan kualiti imej. Secara keseluruhan, bagi Gambarajah Peringkat Ketiga (**3rd Level of Image Enhancement Block Diagram),** ia menerangkan secara kasar tentang perjalanan proses bagi sistem. Kita akan dapat melihat keseluruhan proses yang berlaku dalam proses peningkatan kualiti imej.

Untuk penjelasan yang lebh lanjut tentang subproses–subproses tersebut, Gambarajah Peringkat Kedua (**2nd Level of Image Enhancement Block Diagram**) menerangkan dengan mendalam tentang fungsi-fungsi yang berlaku serta teknik-teknik yang digunakan bagi setiap sub-sistem. Bagi Gambarajah Peringkat Pertama **(1st Level of Image Enhancement Block Diagram),** ia menerangkan tentang input dan output yang berlaku bagi setiap sub-proses.

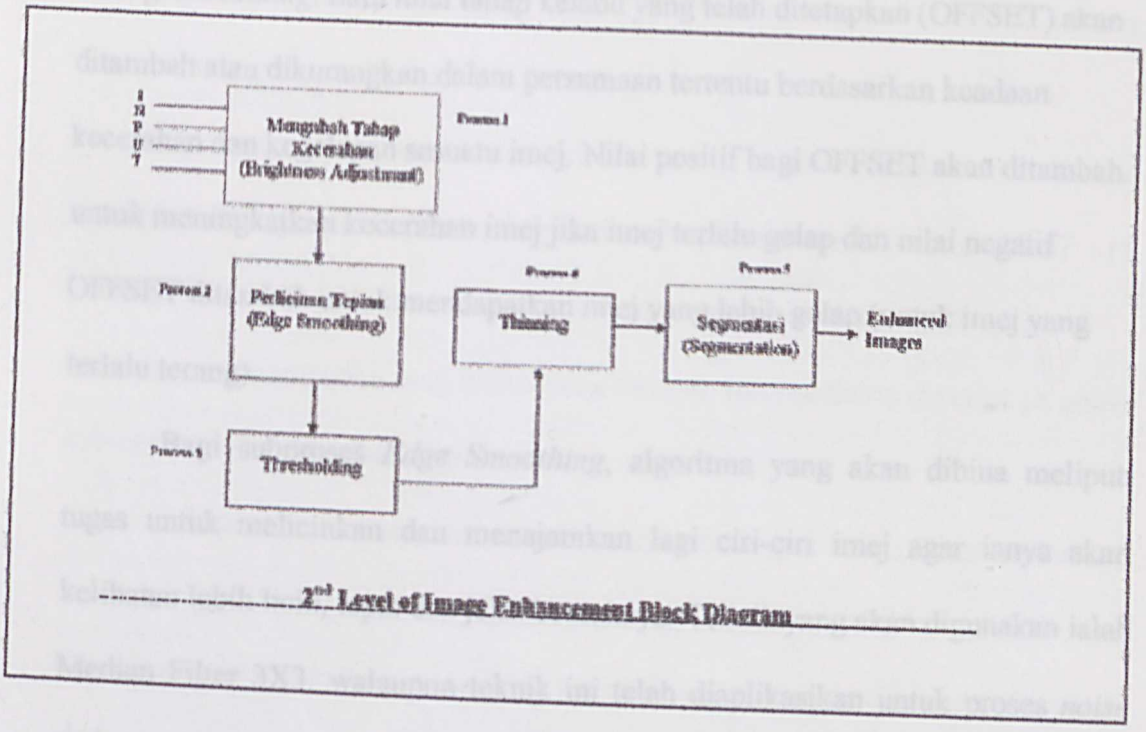
4.1 REKABENTUK SPESIFIK SISTEM



Rajah 4 (a) 3rd Level of Image Enhancement Block Diagram

Gambarajah blok menunjukkan secara lebih terperinci mengenai perjalanan keseluruhan proses, bermula dari imej yang diinput diawal proses sehingga output yang dijana diakhir proses. Output dari proses sebelumnya iaitu *noise reduction* akan digunakan sebagai input kepada proses peningkatan kualiti imej ini. Output yang dijana pada akhir proses dijangka akan menepati output yang dikehendaki jika semua subproses dilaksanakan dengan jayanya.

4.2 REKABENTUK ASAS SISTEM



Rajah 4 (b) 2nd Level of Image Enhancement Block Diagram

Rajah menunjukkan peringkat kedua gambarajah blok bagi proses peningkatan kualiti imej. Terdapat 5 sub-proses yang terlibat, iaitu Mengubah Tahap Kecerahan (*Brightness Adjustment*), Perlicinan Tepian (*Edge Smoothing*), *Thresholding*, *Thinning* dan juga Segmentasi (*Segmentation*). Imej input bagi *Brightness Adjustment* diperolehi dari output proses *Noise Reduction*. Input bagi 4 subproses lain diperolehi dari output daripada subproses-subproses yang dilaksanakan sebelumnya.

Bagi sub-proses *Brightness Adjustment*, teknik yang akan digunakan ialah *Histogram Sliding*. Satu nilai tahap kelabu yang telah ditetapkan (OFFSET) akan ditambah atau dikurangkan dalam persamaan tertentu berdasarkan keadaan kecerahan dan kegelapan sesuatu imej. Nilai positif bagi OFFSET akan ditambah untuk meningkatkan kecerahan imej jika imej terlalu gelap dan nilai negatif OFFSET ditambah untuk mendapatkan imej yang lebih gelap (untuk imej yang terlalu terang).

Bagi subproses *Edge Smoothing*, algoritma yang akan dibina meliputi tugas untuk melicinkan dan menajamkan lagi ciri-ciri imej agar ianya akan kelihatan lebih licin, tepat dan jelas bentuknya. Teknik yang akan digunakan ialah Median Filter 3X3. walaupun teknik ini telah diaplikasikan untuk proses *noise reduction*, namun ia juga boleh digunakan untuk melaksanakan proses perlicinan tepian.

Subproses yang ketiga ialah *Thresholding*. Dalam subproses ini, imej yang diinput dalam bentuk *gray-scale*, akan dioutput sebagai imej berbentuk binari. Proses ini menukarkan imej berbentuk *gray-scale* dan RGB kepada imej binari (bit 1 dan 0). Proses ini sangat penting kerana imej binari diperlukan untuk subproses–subproses yang seterusnya. Saiz sebenar imej yang diperlukan ialah 512 X 512 bilangan piksel. Perwakilan matrik digunakan untuk menunjukkan piksel yang mengandungi imej. Satu teknik yang dinamakan *Zero Padding* diaplikasikan pada subproses ini. *Zero Padding* ini akan mengisi piksel-piksel yang tidak mempunyai imej, yang ditunjukkan oleh perwakilan matrik, dengan

nilai 0 . Ia bertujuan hanya untuk tidak membiarkan matrik 512 X 512 kosong tanpa sebarang nombor/nilai.

Sub-proses yang seterusnya ialah *thinning*. Proses ini akan menukarkan input yang mempunyai ketebalan imej beberapa piksel kepada imej 'skeleton' (1 piksel). Teknik *skeletonisation* dan *erosion* digunakan untuk mengurangkan ketebalan dan sempadan imej binari yang diinput. Imej skeleton diperlukan untuk sub-proses seterusnya iaitu segmentasi.

Sub-proses yang terakhir ialah segmentasi. Output imej (dalam kajian ini, input ialah perkataan Jawi yang bersambung) selepas proses ini ialah aksara-aksara tunggal Jawi. Perkataan yang bersambung tadi akan diasingkan menjadi aksara-aksara tunggal yang akan digunakan dalam proses pengecaman. Kaedah Sobel akan digunakan. Kaedah ini dipilih untuk membolehkan sempadan dikenalpasti dan pengasingan aksara Jawi yang bersambung dan tidak bersambung dapat dilakukan.

Gambarajah yang berikutnya akan menerangkan dengan lebih terperinci perjalanan subproses–subproses yang berlaku dalam proses peningkatan kualiti imej.

4.3 REKABENTUK MODUL INPUT/OUTPUT SISTEM

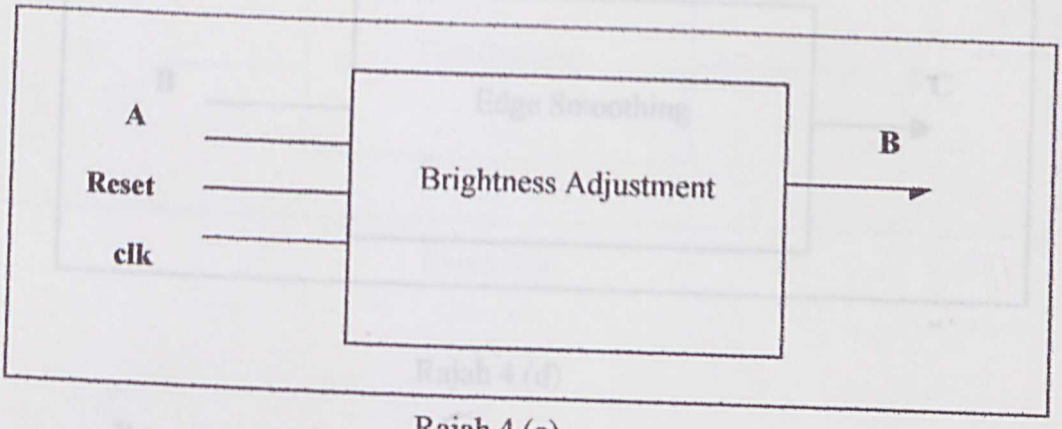
1st Level of Image Enhancement Block Diagram

Merujuk kepada Rajah **1st Level of Image Enhanment Block Diagram**, ia menunjukkan secara ringkas input dan output bagi setiap subproses. Imej input bagi semua subproses merupakan output bagi subproses yang sebelumnya kecuali subproses Mengubah Tahap Kecerahan (*Brightness Adjustment*). Input bagi subproses itu diperolehi dari proses yang sebelumnya iaitu pengurangan kebisingan (*Noise Reduction*).

Bagi output bagi setiap subproses, ia akan disimpan sementara di dalam Memory Buffer sebelum ianya diperlukan oleh sub-proses yang berikutnya. Output bagi setiap subproses ada yang berbeza. Terdapat 2 subproses yang menghasilkan output dalam bentuk *gray-scale* iaitu Mengubah Tahap Kecerahan (*Brightness Adjustment*) dan Perlicinan Tepian (*Edge Smoothing*). Bagi subproses *Thresholding*, *Thinning* dan Segmentasi, outputnya adalah dalam bentuk binari (bit 1 dan bit 0).

1st Level of Image Enhancement Block Diagram

Brightness Adjustment



Rajah 4 (c)

Input :

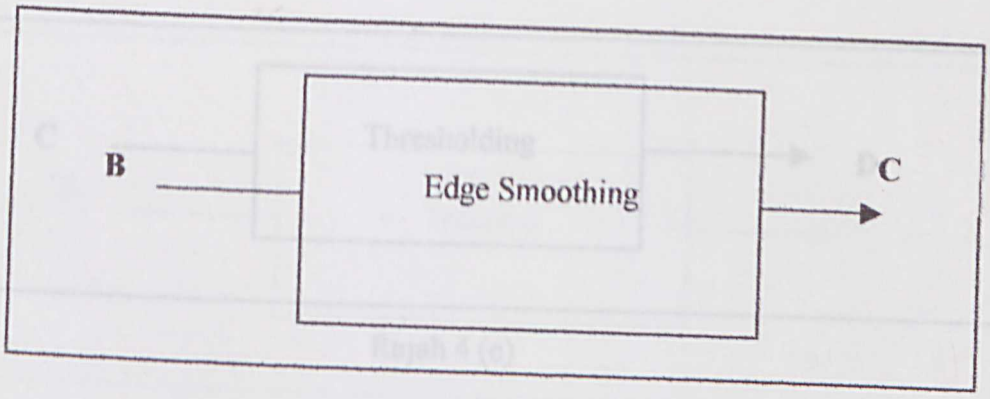
1. **A** – data yang diinput merupakan input 8-bit. Input diperoleh selepas imej mengalami proses pengurangan kebisingan (*noise reduction*).
2. Reset – Menyediakan sistem untuk kembali ke keadaan asal.
3. Clk – Sediakan model dengan jam sistem (1 bit)

Output :

B ialah imej tahap kelabu *(gray scale)* bersaiz 8-bit. Imej output tersebut akan lebih terang dan jelas berbanding imej asal yang diinput.

Imej yang dioutput itu akan disimpan secara sementara didalam *Memory Buffer.*

Edge Smoothing



Rajah 4 (d)

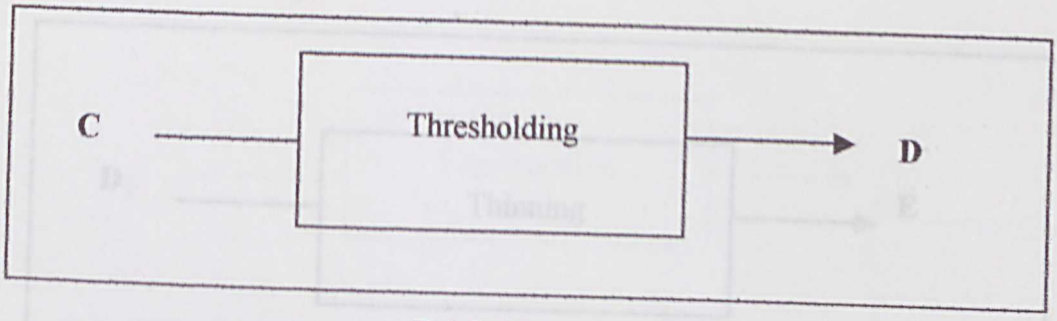
B :

Imej yang diinput merupakan output dari proses sebelumnya, *Brightness Adjustment*. Imej tersebut diperolehi dari *Memory Buffer*, tempat simpanan sementara imej-imej yang telah mengalami perubahan semasa proses-proses tertentu dilaksanakan. Imej merupakan imej tahap kelabu (*gray scale*).

C :

Output merupakan imej yang mempunyai tepian yang lebih tajam dan licin. Rupabentuk sebenar imej boleh dilihat dengan lebih jelas dan tepat. Imej output akan disimpan secara sementara di dalam *Memory Buffer*.

Thresholding



Rajah 4 (e)

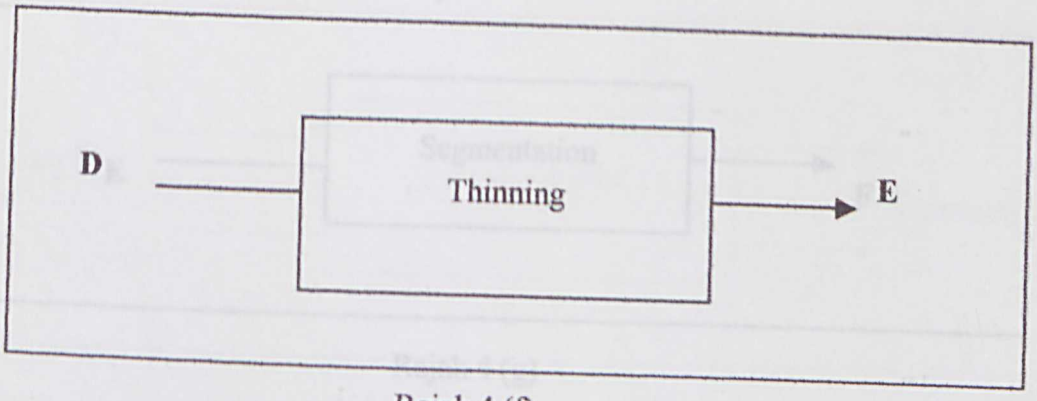
C :

Imej yang diinput merupakan imej tahap kelabu (*gray scale*) dan ia merupakan output proses sebelumnya, *Edge Smoothing*. Imej tersebut diperoleh dari *Memory Buffer*.

D :

Output merupakan imej binari (bit 0 dan bit 1). Imej binari ini akan digunakan untuk proses segmentasi. Output akan disimpan secara sementara di dalam *Memory Buffer*.

Thinning



Rajah 4 (f)

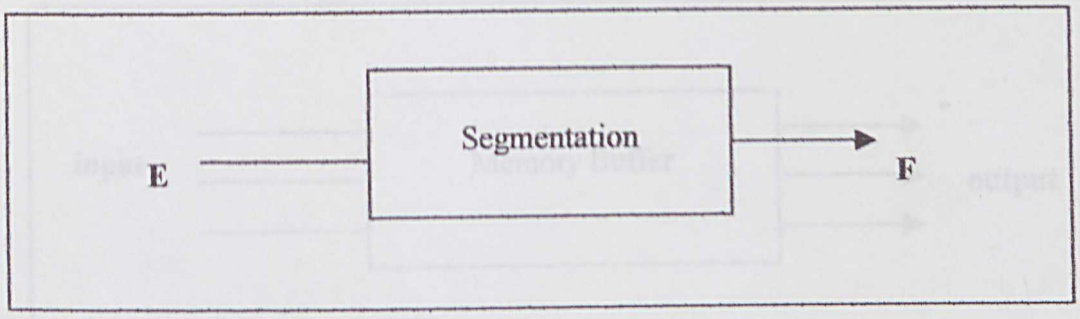
D :

Input merupakan imej binari yang didapati daripada *Memory Buffer*. Ia merupakan output daripada proses sebelumnya, *Thresholding*.

E:

Output bagi proses ini ialah imej 'skeleton' iaitu imej asal (binari) yang dikurangkan ketebalan bentuk garisnya, daripada beberapa piksel ke satu piksel sahaja. Output akan disimpan secara sementara di dalam *Memory Buffer*.

Segmentation



Rajah 4 (g)

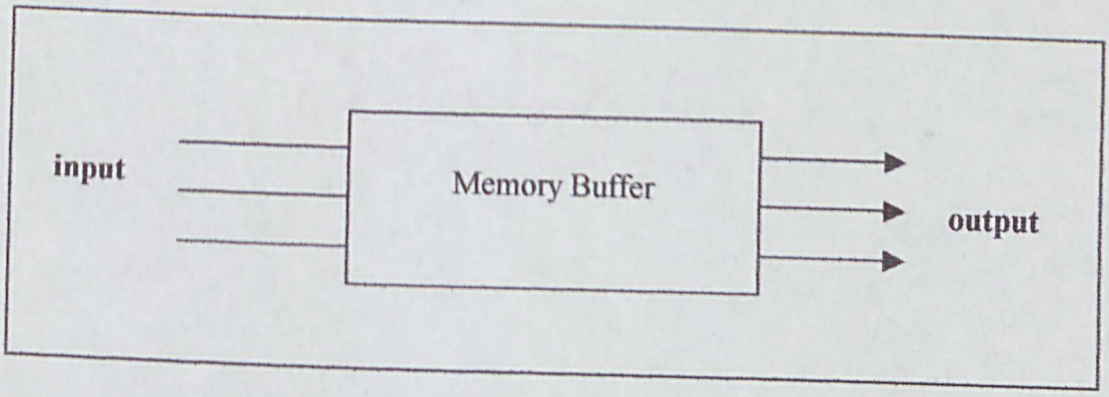
E :

Input bagi proses ini ialah imej binari yang berbentuk 'skeleton'. Imej tersebut diperolehi dari *Memory Buffer*. Ia merupakan output daripada proses sebelumnya, *Thinning*.

F:

Output proses ini ialah imej binari yang telah disegmenkan. Imej (perkataan Jawi) yang panjang telah diasingkan kepada aksara-aksara tunggal. Imej yang telah disegmenkan ini akan digunakan untuk proses pengecaman. Imej output tersebut akan disimpan secara sementara di dalam *Memory Buffer*.

Memory Buffer



Input :

Input bagi *Memory Buffer* diperolehi daripada output bagi proses-proses yang telah selesai diimplementasikan. Ia hanya menyimpan secara sementara sebelum imej-imej tersebut diperlukan oleh sesuatu proses.

Output :

Output bagi *Memory Buffer* ialah imej yang diperlukan oleh sesuatu proses untuk dijadikan sebagai input proses tersebut. Setiap output bagi subproses yang telah dilaksanakan akan disimpan secara sementara di dalam *Memory Buffer* sebelum ia diambil sebagai input untuk subproses yang berikutnya. Saiz ruang penyimpanan ialah 512 X 512 bilangan piksel.

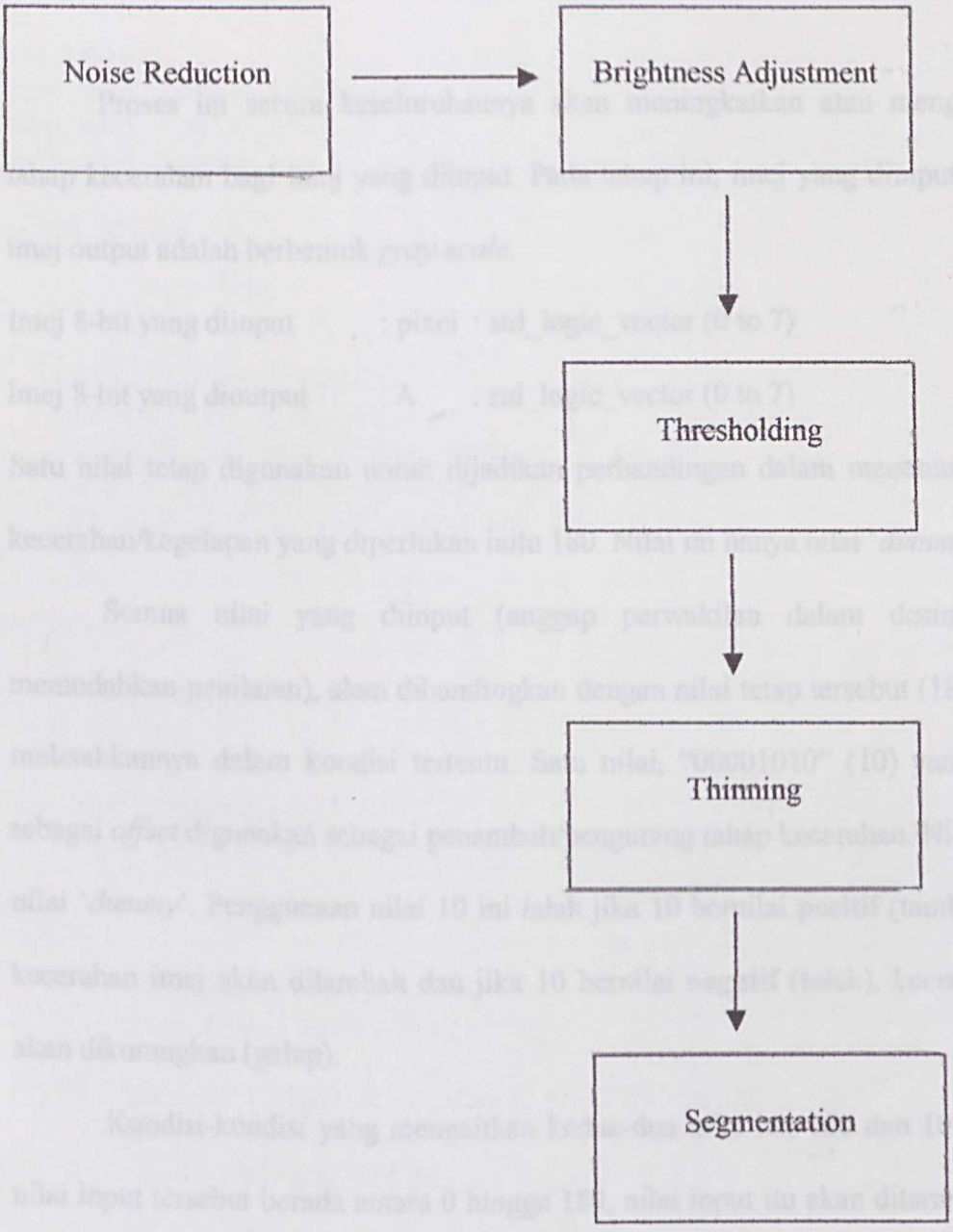
BAB 5

PEMBANGUNAN SISTEM

5.0 PEMBANGUNAN SISTEM

Dalam bahagian ini akan diterangkan mengenai algoritma-algoritma yang telah dibangunkan untuk menerangkan kelakuan subproses-subproses yang terlibat dalam Proses Peningkatan Kualiti Imej (*Image Enhancement*). Terdapat 4 proses yang benar-benar diperlukan untuk melaksanakan Proses Peningkatan Kualiti Imej tersebut berbanding 5 proses seperti yang telah dinyatakan dalam bahagian-bahagian terdahulu. Subproses-subproses itu adalah Mengubah Tahap Kecerahan (*Brightness Adjustment*), *Thresholding*, *Thinning* dan Segmentasi *(Segmentation*). Proses Perlicinan Tepian (*Edge Smoothing*) tidak dilakukan kerana proses ini telah dijalankan pada peringkat *Noise Reduction.* Algoritma-algoritma yang dibangunkan menggunakan bahasa pengaturcaraan VHDL.

Modul yang terdapat dalam Pembangunan Sistem :



Rajah 5 (a) Modul Pembangunan Sistem

5. 1 Mengubah Tahap Kecerahan (Brightness Adjustment)

Proses ini secara keseluruhannya akan meningkatkan atau mengurangkan tahap kecerahan bagi imej yang diinput. Pada tahap ini, imej yang diinput dan juga imej output adalah berbentuk *gray-scale*.

Imej 8-bit yang diinput : pixel : std_logic_vector (0 to 7)

Imej 8-bit yang dioutput : A : std_logic_vector (0 to 7)

Satu nilai tetap digunakan untuk dijadikan perbandingan dalam menentukan tahap kecerahan/kegelapan yang diperlukan iaitu 180. Nilai ini hanya nilai '*dummy*'(lalai).

Semua nilai yang diinput (anggap perwakilan dalam desimal untuk memudahkan penilaian), akan dibandingkan dengan nilai tetap tersebut (180) dengan meletakkannya dalam kondisi tertentu. Satu nilai, "00001010" (10) yang dikenali sebagai *offset* digunakan sebagai penambah/pengurang tahap kecerahan. Nilai ini juga nilai '*dummy*'. Penggunaan nilai 10 ini ialah jika 10 bernilai **positif** (tambah), maka kecerahan imej akan ditambah dan jika 10 bernilai **negatif** (tolak), kecerahan imej akan dikurangkan (gelap).

Kondisi-kondisi yang mengaitkan kedua-dua nilai itu, 180 dan 10 ialah, jika nilai input tersebut berada antara 0 hingga 180, nilai input itu akan ditambah dengan 10 untuk meningkatkan tahap kecerahan imej agar ianya tidak terlalu gelap. Jika nilai input tersebut berada antara 181 hingga 255, nilai input itu akan ditolak dengan 10 untuk mengurangkan tahap kecerahan imej tersebut agar ianya tidak terlalu cerah.

Algoritma modul *Brightness Adjustment*

```
----------------------------------------------------------
-- Auto-generated module template: BR_AD

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;
use ieee.std_logic_arith.all;

entity BR_AD is

  port (

        clk: in std_logic;
        reset: in std_logic;
        pixel: in std_logic_vector(0 to 7);
       add : std_logic_vector(0 to 7):="00001010";
       A: out std_logic_vector(0 to 7)
   );

end BR_AD;

architecture BEHAVIOR of BR_AD is

begin
    process (reset,clk)
    variable pixel_value:std_logic_vector(0 to 7);

  begin

     if rising_edge(clk) then
        pixel_value:=pixel;

       if pixel > 180 and pixel <= 255 then
          pixel_value:= pixel_value - add;  --gelapkan
            A<=pixel_value;

          else pixel_value:= pixel_value + add; --cerahkan
            A<=pixel_value;
          end if;
       end if;

 end process;
end BEHAVIOR;
```

5.2 Thresholding

Untuk proses ini, imej yang diinput yang mana ianya imej *grey-scale* akan ditukarkan kepada imej berbentuk *binary* (bit 0 dan bit 1). Imej *binary* penting untuk digunakan dalam proses-proses yang berikutnya.

Imej 8-bit yang diinput : A : std_logic_vector (0 to 7)

Imej yang dioutput : B : std_logic

Secara keseluruhannya, proses ini akan menukarkan imej *gray-scale* yang mempunyai nilai antara 0 dan 255 kepada imej hitam putih (*binary*). Bagi proses ini, satu nilai yang dipanggil nilai *threshold*, telah ditetapkan untuk membandingkannya dengan nilai bagi piksel-piksel untuk memenuhi kondisi-kondisi tertentu. Nilai *threshold* itu adalah **180.** Nilai ini hanya nilai *dummy* (lalai).

Kondisi-kondisi yang terlibat ialah, jika input yang nilainya berada antara 0 dan 180, nilai-nilai ini akan ditukar kepada **'0'** iaitu warna hitam. Penetapan ini lebih mudah kerana piksel yang berada pada nilai antara 0 hingga 180 (*gray-scale*) lebih gelap dan hampir kepada nilai '0' (*binary*). Manakala bagi input yang nilainya berada antara 181 hingga 255, piksel-piksel itu akan ditukar kepada **'1'** iaitu warna putih. Penetapan ini lebih mudah kerana piksel yang berada pada nilai-nilai tersebut lebih cerah dan lebih hampir kepada nilai '1'.

Algoritma modul *Thresholding*

```
----------------------------------------------------------------
-- Auto-generated module template: THRESHOLD

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;
use ieee.std_logic_arith.all;

entity THRESHOLD is
    port (
        clk: in std_logic;
        reset : in std_logic;
        A: in std_logic_vector (0 to 7); --imej grey-scale
        B: out std_logic                 --imej binary

    );

end THRESHOLD;

architecture BEHAVIOR of THRESHOLD is

begin
   process (reset,clk)
      variable Qpixel :std_logic;

   begin

      if rising_edge(clk) then
         if reset ='1' then
            Qpixel:='1';

            if A > 180 and A <= 255 then
                Qpixel:='1' ;    --putih
                B<=Qpixel;

            else Qpixel:='0' ; --hitam
                B<=Qpixel;
            end if;
        end if;
      end if;
   end process;

end BEHAVIOR;
```

5.3 *Thinning*

Bagi proses ini, imej yang diinput (*binary*) akan menghasilkan output imej yang lebih nipis (ketebalan adalah 1 piksel). Ini memudahkan proses pengecaman.

Imej *binary* yang diinput : B : std_logic

Imej yang dioutput : C : std_logic

Bagi proses ini, imej 'Ba' digunakan sebagai imej *default*. Untuk menghasilkan imej 'Ba' yang nipis, beberapa formula matematik digunakan berdasarkan *array* yang tertentu. Kemudian, hasil dari formula tersebut akan dipadankan dengan imej 8X8 iaitu 64 piksel. Nombor piksel yang sama dengan hasil formula tersebut akan ditetapkan sebagai '0'. Bagi nombor piksel yang berlainan dari hasil formula tersebut akan ditetapkan sebagai '1'.

Satu *indicator* telah digunakan untuk menentukan piksel mana yang akan mengalami proses *thinning*. *Indicator* yang saya gunakan ialah **load**. Piksel akan mengalami proses 'thin' jika *load<='1'*.

Secara keseluruhannya, imej akan kelihatan seperti berikut :

1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1
1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1

Piksel yang diwakili dengan nilai '0' adalah berwarna hitam. Piksel-piksel lain yang bukan '0' diletakkan nilai '1' iaitu berwarna putih. Oleh itu, piksel-piksel bernilai '0' ini akan menyerlahkan aksara imej yang telah diproses. Susunan piksel diatas akan dibaca dari kiri ke kanan bermula dari piksel 1 hingga piksel ke-64. Walaubagaimanapun, berdasarkan algoritma yang telah dibina, kesemua 64 piksel disusun dalam satu baris memanjang yang mendatar.

Algoritma modul *Thinning*

```
----------------------------------------------------
-- Auto-generated module template: THIN

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;

entity THIN is
    port (

        clk,load: in std_logic;
        reset: in std_logic;
        B: in std_logic;
        C: out std_logic
    );

end THIN;

architecture BEHAVIOR of THIN is

signal num: natural:= -1;

begin
    process(reset, clk)

     variable result, result1, result2, result3, result4,
              result5:integer;
     variable X:std_logic;

    begin
          num <= num + 1;

          if load='1' then

        N1: for i in 1 to 3 loop
            result1:=8*i + 1;
            result2:=8*i + 8;
           end loop N1;

        N2: for i in 1 to 2 loop
            result3:=5*i + 29;
            end loop N2;
```

```
    N3: for i in 1 to 4 loop
          result4:= i + 42;
          end loop N3;

    N4: for i in 1 to 1 loop
        result5:= 60*i + 1;
        end loop N4;

          result:=result1;
          result:=result2;
          result:=result3;
          result:=result4;
          result:=result5;

          X:='0';

       elsif load='0' then
       X:='1';

     end if;

    C<=X;
  end process;
end BEHAVIOR;
```

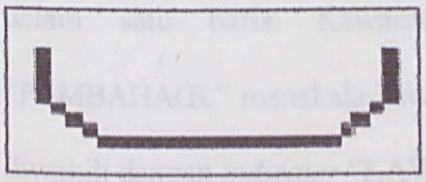
5.4 Segmentasi (*Segmentation*)

Segmentasi merupakan proses terakhir bagi proses peningkatan kualiti imej (*Image Enhancement*). Secara keseluruhannya, proses ini akan bertindak mengasingkan/ memecahkan kawasan-kawasan dalam keseluruhan piksel antara aksara asas dan aksara tambahan bagi aksara tunggal Jawi. Syarat asas bagi proses segmentasi ialah aksara asas dan aksara tambahan bagi huruf Jawi yang sama atau berlainan tidak boleh berada dalam baris piksel yang sama. Jika syarat tersebut tidak dipatuhi dimana aksara asas dan aksara tambahan berada pada baris yang sama, maka proses segmentasi tidak dapat dilakukan. Imej *default* yang digunakan disini ialah imej aksara Jawi 'Ba' dan titiknya.

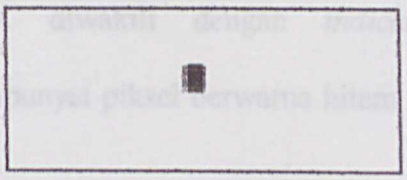
Input dan output adalah berbentuk *binary* (bit 0 dan bit 1). Berdasarkan algoritma yang telah dibina, penggunaan *indicator* akan menunjukkan bahawa kedua-dua aksara iaitu aksara asas dan aksara tambahan 'Ba' telah diasingkan.

Algoritma yang dibangunkan hanya memfokuskan kepada sempadan pengasingan bagi aksara asas dan aksara tambahan. Aksara-aksara yang telah diasingkan ini kemudiannya akan mengalami proses *recognition* yang akan mengenalpasti bentuknya.

Rajah dibawah menunjukkan aksara Jawi 'Ba' selepas proses Segmentasi :



Rajah 5 (b) Aksara Asas



Rajah 5 (c) Aksara Tambahan

Secara keseluruhannya, imej akan kelihatan seperti berikut :

Piksel ke- 1

X	X	X	X	X	X	X	X
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1
X	X	X	X	X	X	X	X
1	1	1	1	0	1	1	1

Petunjuk :

0 – mewakili piksel berwarna hitam

1 – mewakili piksel yang berwarna putih

X – mewakili sempadan pengasingan aksara asas dan aksara tambahan.

Pernomboran piksel-piksel ini dibaca dari kiri ke kanan.

Merujuk kepada algoritma yang dibina, semua piksel disusun secara mendatar dalam satu baris. Kawasan bertanda X diwakili dengan *indicator* "PEMBAHAGI." manakala kawasan yang mempunyai piksel berwarna hitam (0) diwakili dangan *indicator* "KAW_AKSARA".

Algoritma modul Segmentasi

```
------------------------------------------------------------
-- Auto-generated module template: SEGMENT
--
-- Note: comments beginning with WIZ should be left intact.
--       Other comments are informational only.
--
-- PeakVHDL software version: 5.20c
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_signed.all;
use ieee.std_logic_arith.all;

entity SEGMENT is
    port (

        clk: in std_logic;
        reset: in std_logic;
        C: in std_logic;
        D: out std_logic

    );

end SEGMENT;

architecture BEHAVIOR of SEGMENT is

signal num:integer range 1 to 64:= -1;
signal show_divide:string(0 to 9);

begin

    process(reset, clk )

       variable divider:string(0 to 9);
       variable X:std_logic;

    begin
            X:=C;
            num<= num + 1;
```

```
                if reset= '1' then
                X:='1';
                D<=X;
                show_divide<="PEMBAHAGI.";

            else if reset='0' then
                X:='0';
                D<=X;
                show_divide<="KAW_AKSARA";

          end if;
        end if;

      D<=X;
    end process ;

end BEHAVIOR;
```

BAB 6

PENGUJIAN SISTEM

6.0 PENGUJIAN SISTEM

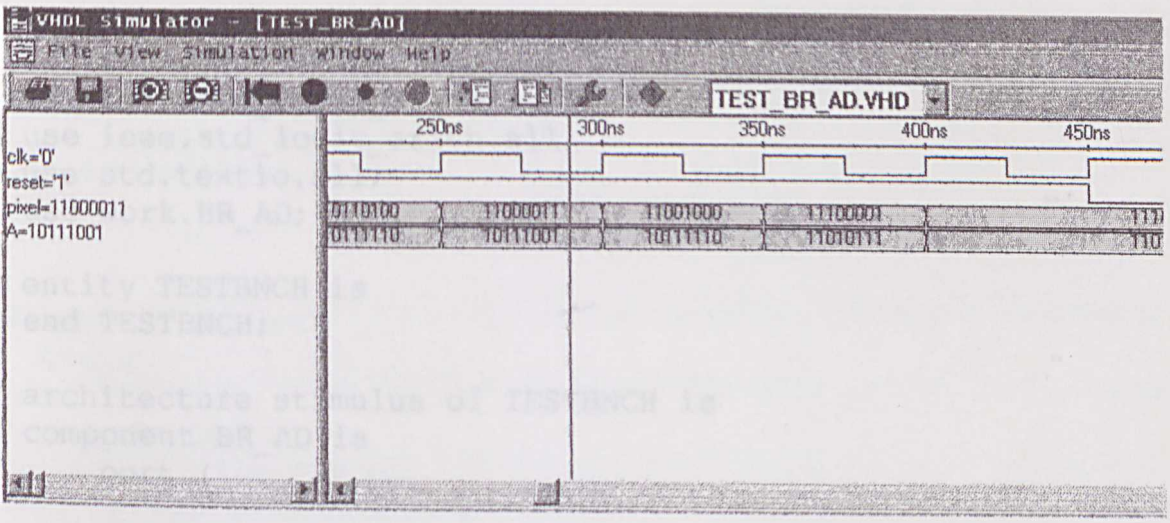
Pengujian sistem akan melibatkan algoritma *test-bench* yang akan menunjukkan bagaimana keadaan output bagi imej selepas melalui proses-proses yang tertentu. Dalam test bench ini akan dimasukkan nilai piksel-piksel bagi imej tersebut. Selepas algoritma *test bench* ini berjaya dilarikan, ia akan menghasilkan *waveform* yang akan menunjukkan perubahan keadaan piksel (output) bagi setiap piksel yang diinput. 2 proses yang pertama iaitu *Brightness Adjustment* dan *Thresholding*, input yang dimasukkan hanya nilai-nilai percubaan untuk memastikan bahawa ianya memenuhi kondisi-kondisi tertentu yang telah dibangunkan didalam algoritma modul-modul tersebut. Output yang terhasil tidak akan digunakan dalam proses-proses yang seterusnya. Bagi 2 poses yang lain iaitu *Thinning* dan Segmentasi, imej *default* yang diuji ialah aksara Jawi 'Ba'. Input yang dimasukkan di dalam algoritma *test bench* adalah berdasarkan kedudukan piksel-piksel yang membentuk huruf Jawi 'Ba' tersebut didalam susunan piksel bersaiz 8X8.

6. 1 Brightness Adjustment

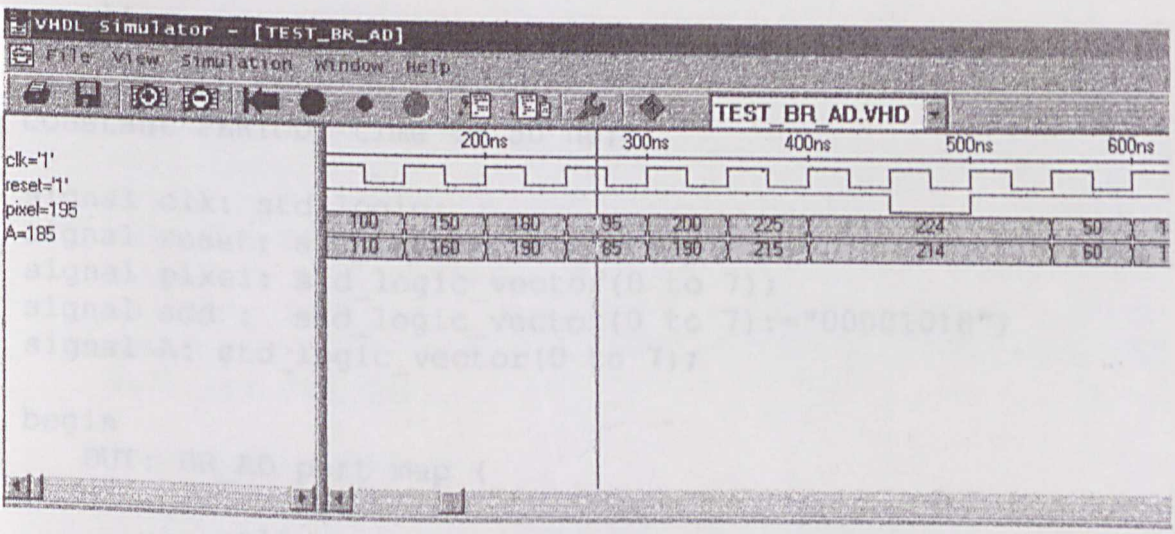
Bagi proses ini, output yang terhasil ialah lebih jelas dilihat dengan mengubah tahap kecerahannya. Jika imej yang diinput terlalu gelap, proses ini akan menambah nilai *offset* agar ianya lebih terang dan jelas, manakala jika imej input terlalu terang, proses ini akan mengurangkan nilai *offset* agar imej itu lebih jelas dan mudah dilihat. Nilai lalai bagi *offset* ini adalah 10. Penambahan atau pengurangan nilai *offset* ini pula adalah berdasarkan kondisi-kondisi tertentu. Algoritma *test bench* akan menunjukkan bahawa kondisi-kondisi ini telah dipenuhi dan outputnya dapat dilihat melalui *waveform* yang terjana, hasil selepas algoritma *test bench* ini berjaya dilarikan. Input-input yang dimasukkan dalam algoritma test bench untuk proses ini hanya nilai-nilai percubaan untuk memastikan bahawa algoritma yang dibina menepati kehendak ataupun sebaliknya. Hasil output dari proses ini tidak akan digunakan dalam proses-proses yang seterusnya.

Waveform bagi *Brightness Adjustment* dijalankan dalam 2 bentuk format vektor iaitu *binary* dan *decimal*. Format vektor *decimal* adalah bertujuan untuk memudahkan penilaian terhadap output yang diperolehi samada ianya menepati kehendak atau sebaliknya. Selepas algoritma di atas berjaya dilarikan, *waveform* yang terhasil adalah seperti berikut :

Option → Vector Format Display : Binary



Option → Vector Format Display : Decimal



Gambarajah 6(a) : *Brightness Adjustment Waveform*

Algoritma *test bench Brightness Adjustment*

```
--------------------------------------------------------------------------------
Auto-generated test bench template for: BR_AD

 library ieee;
 use ieee.std_logic_1164.all;
 use ieee.std_logic_signed.all;
 use ieee.std_logic_arith.all;
 use std.textio.all;
 use work.BR_AD;

 entity TESTBNCH is
 end TESTBNCH;

 architecture stimulus of TESTBNCH is
 component BR_AD is
     port (

         clk: in std_logic;
         reset: in std_logic;
         pixel: in std_logic_vector(0 to 7);
         add : in std_logic_vector(0 to 7);
         A: out std_logic_vector(0 to 7)

     );

 end component;
 constant PERIOD: time := 50 ns;

 signal clk: std_logic;
 signal reset: std_logic;
 signal pixel: std_logic_vector(0 to 7);
 signal add :  std_logic_vector(0 to 7):="00001010";
 signal A: std_logic_vector(0 to 7);

 begin
    DUT: BR_AD port map (

            clk,
            reset,
            pixel,
            add,
            A
          );
```

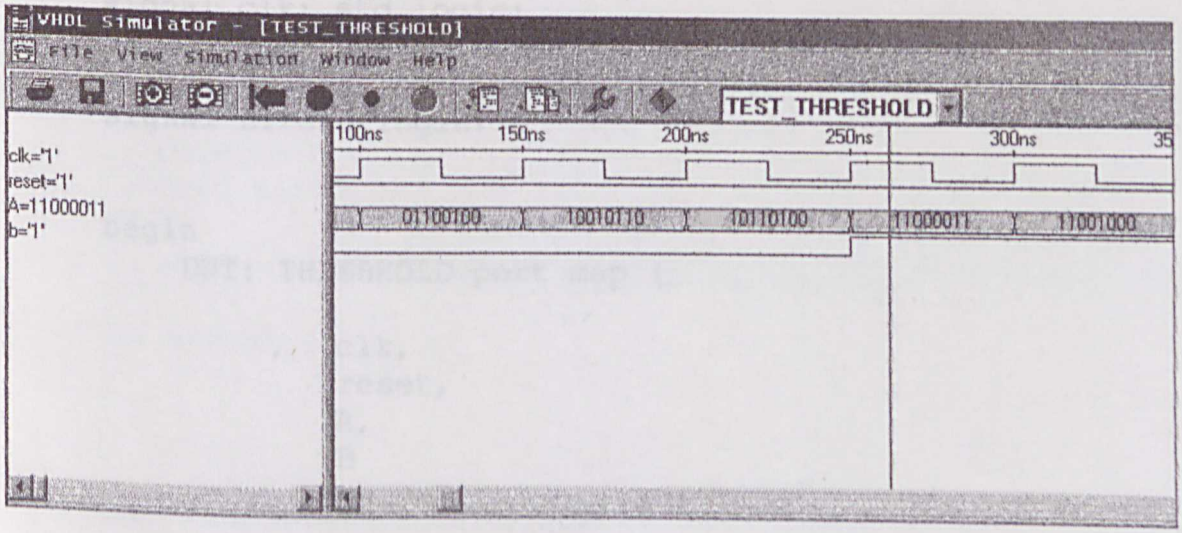
```
  CLOCK: process
   begin

        clk <= '1';
        wait for 25 ns;
        clk <= '0';
        wait for 25 ns;
    end process;


STIMULUS: process
  begin

      reset<='1';
      wait for PERIOD;
      pixel <="00110010";
      wait for PERIOD;
      pixel <= "01100100";
      wait for PERIOD;
      pixel <= "10010110";
      wait for PERIOD;
      pixel <= "10110100";
      wait for PERIOD;
      pixel <= "11000011";
      wait for PERIOD;
      pixel <= "11001000";
      wait for PERIOD;
      pixel <= "11100001";
      wait for PERIOD;
      pixel <= "11100000";
      wait for PERIOD;
      reset<='0';
      wait for PERIOD;

    end process STIMULUS;

end stimulus;
```

6.2 Thresholding

Bagi proses ini, algoritma *test bench* turut dijana untuk membolehkan output terhasil adalah betul dan dapat dilihat. Seperti juga *Brightness Adjustment, waveform* yang terhasil selepas algoritma *test bench* berjaya dilarikan akan menunjukkan samada output yang diperoleh menepati kehendak proses *Thresholding* atau sebaliknya.

Nilai input bagi proses ini juga merupakan nilai percubaan untuk membuktikan bahawa proses ini berjaya dijalankan. Walaupun ianya kelihatan mudah dan ringkas, namun proses *Thresholding* adalah penting. Proses ini akan menukarkan imej yang berbentuk *grey-scale* kepada imej *binary*. Penentuan samada input-input yang dimasukkan akan menghasilkan output bit '0' atau bit '1' adalah berdasarkan kondisi-kondisi yang terdapat di dalam algoritma modul bagi proses ini. *Waveform* yang terhasil selepas algoritma *test bench* berjaya dilarikan akan memaparkan input yang dimasukkan serta output yang terhasil.

Option → Vector Format Display : Binary



Gambarajah 6 (b) : *Thresholding Waveform*

Algoritma *test bench Thresholding*

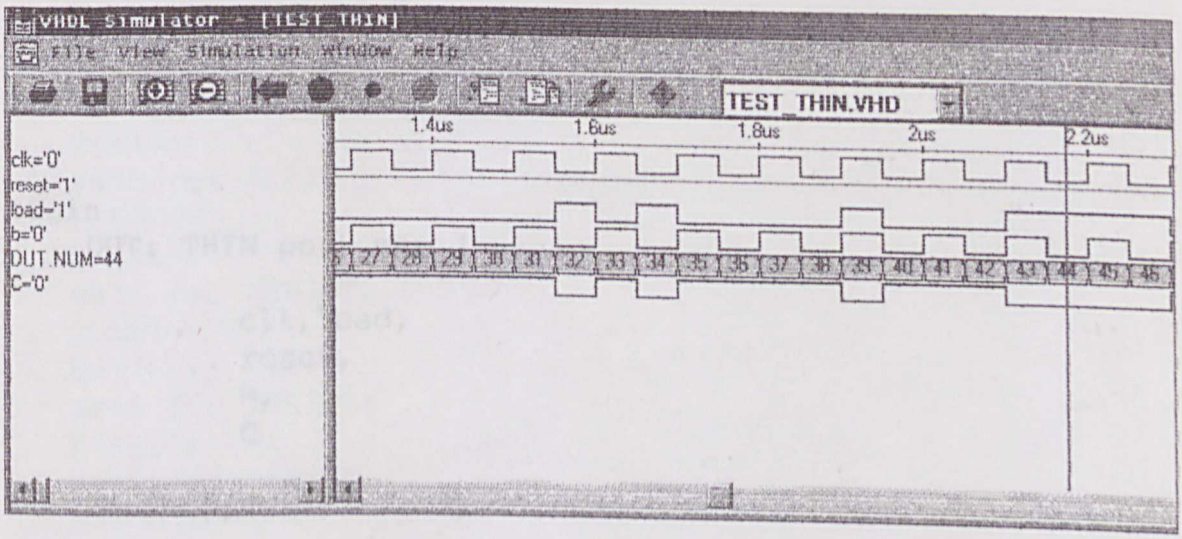
```
--------------------------------------------------------------------------------
-- Auto-generated test bench template for: THRESHOLD

library ieee;
use ieee.std_logic_1164.all;
use std.textio.all;
use work.THRESHOLD;

entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component THRESHOLD is
    port (

        clk: in std_logic;
        reset: in std_logic;
        A: in std_logic_vector(0 to 7);
        B: out std_logic

    );

end component;

constant PERIOD : time := 50 ns;

signal clk: std_logic;
signal reset: std_logic;
signal A: std_logic_vector(0 to 7);
signal B: std_logic;


begin
    DUT: THRESHOLD port map (

            clk,
            reset,
            A,
            B
          );
```

```
    CLOCK2: process
      begin

            clk <= '1';
             wait for 25 ns;
         clk <='0';
        wait for 25 ns;
       end process CLOCK2;


 STIMULUS2: process
   begin

    reset<='1';
     wait for PERIOD;
    A <="00110010";
     wait for PERIOD;
    A <= "01100100";
     wait for PERIOD;
    A <= "10010110";
   wait for PERIOD;
    A <= "10110100";
   wait for PERIOD;
    A <= "11000011";
    wait for PERIOD;
    A <= "11001000";
    wait for PERIOD;
    A <= "11100001";
    wait for PERIOD;
   A <= "11100000";
   wait for PERIOD;
    reset<='0';
    wait for PERIOD;

    end process STIMULUS2;

end stimulus;
```

6.3 *Thinning*

Proses Thinning ini akan menguji imej aksara Jawi 'Ba'. Seperti yang telah diterangkan dalam bahagian Thinning bagi bab 5, Pembangunan Sistem, imej aksara Jawi 'Ba' merupakan piksel-piksel yang berwarna hitam yang terbentuk dalam susunan piksel bersaiz 8X8. Algoritma test bench yang telah dibina akan menginput nilai-nilai bagi setiap piksel. Setelah algoritma ini berjaya dilarikan, waveform akan terbentuk dan akan menunjukkan piksel-piksel mana yang akan mengalami proses Thinning. Merujuk kepada waveform, kesemua piksel disusun secara mendatar, dalam satu barisan yang panjang. Satu indicator telah digunakan iaitu load. Jika load<=' 1', maka piksel akan bernilai '0' iaitu berwarna hitam. Piksel-piksel ini kemudian akan membentuk aksara Jawi 'Ba' yang setebal satu piksel (nipis). Berikut merupakan *waveform* yang terhasil selepas algoritma test bench berjaya dilarikan :

Option → Vector Format Display: Binary



Gambarajah 6 (c) : *Thinning Waveform*

Algoritma *test bench Thinning*

```
--------------------------------------------------
-- Auto-generated test bench template for: THIN

library ieee;
use ieee.std_logic_1164.all;
use std.textio.all;
use work.THIN;

entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component THIN is
    port (

        clk,load: in std_logic;
        reset: in std_logic;
        B: in std_logic;
        C: out std_logic

    );

end component;

constant PERIOD: time :- 50 ns;

signal clk,load: std_logic;
signal reset: std_logic;
signal B: std_logic;
signal C: std_logic;


begin
    DUT: THIN port map (

            clk,load,
            reset,
            B,
            C

        );
```

```
CLOCK1: process
begin
     clk<='1';
     wait for PERIOD;
     clk<='0';
     wait for PERIOD;
end process CLOCK1;

STIMULUS1: process
begin

reset<='1';
load<='0';
B<='1';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
load<='1';          --9
B<='1';
wait for PERIOD;
 load<='0';
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
```

```
 B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
load<='1';
B<='1';                    --16
wait for PERIOD;
load<='1';
B<='1';                    --17
wait for PERIOD;
load<='0';
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
 load<='1';
B<='0';                    --24
wait for PERIOD;
B<='1';                    --25
wait for PERIOD;
load<='0';
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
load<='1';
B<='0';                  --32
```

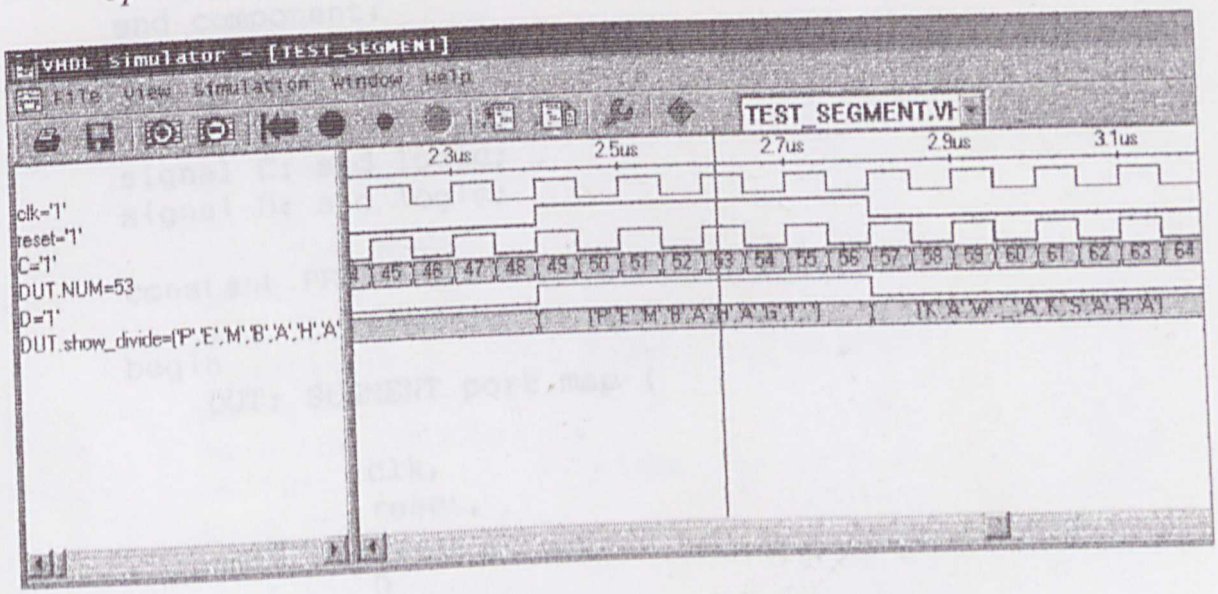
```
wait for PERIOD;
load<='0';
B<='1';
wait for PERIOD;
load<='1';
B<='0';                --34
wait for PERIOD;
load<='0';
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
load<='1';
B<='1';                --39
wait for PERIOD;
 load<='0';
B<='0';
wait for PERIOD;
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
load<='1';
B<='1';                --43
wait for PERIOD;
B<='0';                --44
wait for PERIOD;
B<='1';                --45
wait for PERIOD;
B<='0';                --46
wait for PERIOD;
 load<='0';
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
 reset<='1';
B<='1';
wait for PERIOD;
B<='0';
wait for PERIOD;
```

```
        B<='1';
        wait for PERIOD;
        B<='0';
        wait for PERIOD;
        B<='1';
        wait for PERIOD;
        B<='0';
        wait for PERIOD;
        B<='1';
        wait for PERIOD;
        B<='0';
        wait for PERIOD;
        B<='1';
        wait for PERIOD;
         reset<='0';
        B<='0';
        wait for PERIOD;
        B<='1';
        wait for PERIOD;
        B<='0';
        wait for PERIOD;
        load<='1';
        B<='1';              --61
        wait for PERIOD;
        load<='0';
        B<='0';
        wait for PERIOD;
        B<='1';
        wait for PERIOD;
        B<='0';
        wait for PERIOD;
         reset<='1';
        wait for PERIOD;
      end process STIMULUS1;

end stimulus;
```

6.4 Segmentasi (Segmentation)

Proses ini merupakan subproses terakhir dalam proses Peningkatan Kualiti Imej (*Image Enhancement*). Seperti subproses-subproses yang lepas, algoritma test bench juga dibina untuk menjalankan pengujian terhadap modul yang dibina. Dalam algoritma ini, kesemua nilai bagi piksel akan diinput (64 piksel). Input yang dimasukkan adalah berdasarkan imej aksara Jawi 'Ba' yang berada dalam susunan piksel bersaiz 8X8. Namun, input yang dimasukkan adalah secara mendatar dalam satu barisan yang panjang. Secara keseluruhannya, proses Segmentasi ini akan mengasingkan antara aksara asas dan aksara tambahan bagi aksara Jawi 'Ba' tersebut. Berikut adalah *waveform* yang terhasil selepas algoritma *test bench* berjaya dilarikan :

Option → Vector Format Display : Binary



Gambarajah 6 (d) : *Segmentation Waveform*

Algoritma *test bench Segmentation*

```
--------------------------------------------------
-- Auto-generated test bench template for: SEGMENT


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use std.textio.all;
use work.SEGMENT;

entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component SEGMENT is
    port (

        clk: in std_logic;
        reset: in std_logic;
        C: in std_logic;
        D: out std_logic

    );

end component;

signal clk: std_logic;
signal reset: std_logic;
signal C: std_logic;
signal D: std_logic;

constant PERIOD:time:=50 ns;

begin
    DUT: SEGMENT port map (

            clk,
            reset,
            C,
            D

        );
```

```
CLOCK: process

Begin

  clk<='1';
wait for PERIOD;
clk<='0';
wait for PERIOD;

end process CLOCK;


STIMULUS1: process
begin
   reset<='1';        -- Segmentasi berlaku
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<= '1';
wait for PERIOD;
C<='0';
wait for PERIOD;
 reset<='0';        -- Segmentasi berhenti
C<='1';
wait for PERIOD;
C<='0';             -- piksel ke-10
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
```

```
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';                    --piksel ke-20
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';                    -- piksel ke-30
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
 reset<='0';
C<='0';
wait for PERIOD;
C<='1';
```

```
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';              -- piksel ke-40
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
 reset<='1';          --Segmentasi berlaku
C<='1';
wait for PERIOD;
C<='0';              -- piksel ke-50
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
C<='1';
wait for PERIOD;
C<='0';
wait for PERIOD;
 reset<='0';
```

```
    C<='1';
    wait for PERIOD;
    C<='0';
    wait for PERIOD;
    C<='1';
    wait for PERIOD;
    C<='0';          -- piksel ke-60
    wait for PERIOD;
    reset<='0';      -- Segmentasi berhenti
    C<='1';
    wait for PERIOD;
    C<='0';
    wait for PERIOD;
    C<='1';
    wait for PERIOD;
    C<='0';
    wait for PERIOD;
     reset<='1';
     wait for PERIOD;
    end process STIMULUS1;

end stimulus;
```

BAB 7

PERBINCANGAN

7.0 PERBINCANGAN

Keputusan yang diperolehi adalah berbeza bagi keempat-empat subproses kerana setiap subproses akan melaksanakan fungsi yang berlainan. Bagi proses peningkatan kualiti imej (Image Enhancement) yang diaplikasikan dalam *VHDL Implementation for Character Jawi Recognition*, terdapat 4 subproses yang diperlukan iaitu Mengubah Tahap Kecerahan (*Brightness Adjustment*), *Thresholding*, *Thinning* dan juga Segmentasi (*Segmentation*). Proses Perlicinan Tepian *(Edge Smoothing*) tidak dilaksanakan kerana fungsi di dalam proses ini menyamai proses yang dijalankan dalam proses *Noise Reduction.*

Terdapat kelemahan dan kelebihan yang telah dikenalpasti dalam mengimplementasikannya dalam bahasa pengaturcaraan VHDL. Ianya akan diterangkan mengikut subproses-subproses tersebut.

Brightness Adjustment

Bagi proses ini, kelebihannya dalam pengimplementasian ke atas Jawi Character Recognition ialah kesesimbangan bagi tahap kecerahan bagi imej yang diinput. Selepas mengalami proses ini, imej yang terhasil akan lebih jelas kelihatan serta terang. Namun begitu, terdapat kelemahan yang dikenalpasti dan perlu diperjelaskan. Disebabkan oleh nilai input hanya menggunakan nilai percubaan, maka ianya tidak mengambilkira situasi sebenar iaitu menginput nilai bagi piksel-piksel yang sebenar bagi sesebuah imej. Maka, oleh itu, nilai *offset* juga menggunakan satu nilai dummy. Nilai offset ini perlu dikenalpasti secara tepat untuk membolehkan agar penyeimbangan yang sempurna dapat dijalankan. Penggunaan nilai offset yang tepat akan memberikan hasil yang diingini, iaitu imej yang terang, jelas dan 'cantik'.

Thresholding

Bagi subproses yang kedua ini, ianya hanya melibatkan penukaran format dari grey-scale kepada binary. Thresholding berjaya dilakukan keatas nilai-nilai piksel yang diinput walaupun ianya hanya menggunakan nilai-nilai percubaan. Nilai percubaan ini digunakan kerana ianya hanya untuk memastikan bahawa proses ini dapat dilaksanakan mengikut keperluan dan juga kerana output yang diperoleh tidak akan digunakan untuk proses-proses yang lain.

Thinning

Merujuk kepada algoritma modul bagi subproses ini, ianya akan menipiskan piksel-piksel yang tertentu berdasarkan satu penunjuk, *indicator*. Kelemahan-kelemahan yang telah dikenalpasti ialah:

- tiada penghantaran parameter yang akan membaca nilai dari satu piksel ke piksel yang lain.
- Pemprosesan melibatkan piksel satu per satu. Untuk pemprosesan yang lebih cepat, ianya boleh dilakukan melalui pemprosesan piksel bersaiz 4X4. Ini bermaksud 8 piksel akan mengalami proses thinning secara serentak.
- Input tidak diambil dari buffer. Keadaan ini berlainan dari apa yang telah dirancangkan. Ini adalah kerana input bagi proses ini tidak sama dengan output dari proses sebelumnya, Thresholding. Buffer tidak diwujudkan dalam algoritma yang dibina.

Untuk meningkatkan lagi kualiti dan kefungsian algoritma yang dibina, perlu ditambah fungsi bagi proses thinning ini. Ianya perlu ditunjukkan bagaimana piksel-piksel yang membentuk sesuatu imej (imej asal yang tebal) akan dinipiskan menjadi imej yang setebal satu piksel (imej baru yang nipis dan lebih tajam).

Segmentasi

Bagi subproses terakhir dalam proses peningkatan kualiti imej ini, kelebihannya ialah ia akan mengasingkan aksara asas dan aksara tambahan walaupun aksara tambahan tersebut berada samada di atas ataupun dibawah. Walaubagaimanapun,

ini hanya boleh diimplemetasikan keatas aksara Jawi tunggal sahaja. Tetapi, kelemahan masih terdapat dalam algoritma yang dibina.

- Tiada penghantaran parameter yang akan membaca nilai piksel dari satu piksel ke piksel yang lain.
- Algoritma yang dibina hanya menunjukkan sempadan yang mengalami pengasingan. Bagi kawasan yang mempunyai imej, ianya tidak menunjukkannya secara jelas.
- Input sepatutnya diperoleh dari buffer iaitu output dari proses yang sebelumnya, Thinning. Tiada penggunaan buffer dalam algoritma modul. Keadaan ini berlainan dari apa yang telah dirancangkan.

Untuk meningkatkan lagi kualiti algoritma yang dibina, beberapa perkara harus diperbaiki atau ditambah. Penggunaan buffer adalah lebih efisien kerana input bagi proses ini merupakan output bagi proses yang sebelumnya, Thinning. Selain itu, perlu menunjukkan dengan jelas kawasan yang diasingkan serta kawasan-kawasan yang tidak mengalami pemecahan ini, iaitu piksel-piksel bagi imej aksara asas dan juga aksara tambahan bagi aksara Jawi tersebut.

Cadangan saya ialah agar projek ini dapat diteruskan dimasa yang akan datang dengan memperbaiki kelemahan yang ada, meningkatkan lagi fungsi-fungsi yang terdapat dalam algoritma modul yang dibina seterusnya dapat memenuhi objektif yang hendak dicapai. Kajian yang lebih mendalam perlu dilakukan agar pemahaman yang lebih tepat dapat diaplikasikan kedalam

algoritma yang akan dibina. Projek ini amat penting dalam merealisasikan pengecaman tulisan Jawi .

Kesimpulannya, projek ini telah membantu saya dalam memahami prosedur-prosedur yang perlu dilakukan dalam proses peningkatan kualiti imej (Image Enhancement). Walaupun masih terdapat banyak kelemahan, ianya masih boleh diperbaiki. Saya juga serba sedikit telah memahami bagaimana bahasa pengaturcaraan VHDL dapat membantu dalam proses-proses pengecaman tulisan Jawi ini. Pembangunan algoritma menggunakan bahasa pengaturcaraan VHDL ini perlu dilakukan dengan lebih teliti dan lengkap agar matlamat sebenar dapat dicapai seterusnya merealisasikan pengecaman tulisan Jawi .

RUJUKAN

RUJUKAN

- Umbaugh, E. Scott, Computer Vision and Image Processing: A Practical Approach Using CVIP Tools, Prentice-Hall International, 2001.
- Kenneth R. Castleman, Digital Image Processing, Prentice Hall International Editions, 1996.
- Zainalabedin Navabi, VHDL 2nd Edition : Analysis and Modeling Of Digital Systems, McGraw-Hill International Editions, 1998.
- Milan Sonka, Vaclav Hlavac, Roger Boyle, Image Processing, Analysis, and Machine Vision, 2nd Edtion, PWS Publishing, 1999.
- Adnan Amin. (1998). Off-Line Arabic Character Recognition: The State of The Art. Vol. 31. Pages 517-530.
- Adnan Amin, Seung-Gwon Kim and Claude Sammut. (1997). Hand-printed Chinese Character Recognition via Machine Learning. 190-194.
- Adnan Amin. (2001). Segmentation of Printed Arabic Text. 115-126
- M. S. Khorsheed. (2002). Off-Line Arabic Character Recognition – A Review. 31-45.
- Johnson, M.L. Umbaugh, S.E, Store Separation Flight Test Image Enhancement Tools, telah dipersembahkan dalam di Joint Aircarft-Stores Compatibility subgroup of Joint Ordinance Comanders Group Aircraft-Stores Compatibility Seminar, September 29, 1993, Fort Walton Beach, FL.

- Omar bin Abdul Rahim. Pengecaman Aksara Tulisan Jawi, Latihan Ilmiah Universiti Malaya.
- Bhasker, J. VHDL Primer, Third Edition Bell Laboratories, Lucent Technologies Allentown, PA. Prentice Hall PTR.
- Rushton, Andrew VHDL for Logic Synthesis, Second Edition, TransEDA Limited, Southamton, UK. John Wiley & Sons. 2001.
- Naylor, David. and Jones, Simon. VHDL : A Logic Synthesis Approach. Chapman & Hall. 1997
- Image Processing Fundamental, http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Contents.html
- http://citeseer.nj.nec.com/context/21946/0
- http://cis.jhu.edu/~ulisses/lab/lab5.html
- http://www.eee.bham.ac.uk/et_gr/publications/etrp2.pdf
- http://www.mathworks.com
- http://www.google.com