Malaysian Journal of Computer Science, Vol. 14 No. 1, June 2001, pp. 28-38

REKABENTUK PENDARAB TITIK APUNGAN 32BIT BERTALIAN PAIP MENGGUNAKAN SISTEM PEMBANGUNAN VHDL

Noorzaily Mohamed Noor
Mashkuri Hj. Yaacob
Fakulti Sains Komputer & Teknologi Maklumat
Universiti Malaya
50603 Kuala Lumpur
email: zaily@fsktm.um.edu.my

ABSTRAK

Kertas kerja ini menghuraikan sebuah pendarab titik apungan (PTA) 32bit bertalian paip 18MHz yang direkabentuk menggunakan bahasa perihalan perkakasan VHDL, peralatan sintesis Synopsys FPGA Express dan peralatan pemetaan Xilinx Alliance. PTA ini menggunakan 1007 CLB dan 100 IOB di mana peranti pemetaan yang digunakan ialah xc4036xl-bg432-2 daripada pustaka XC4000 Xilinx FPGA. Pendarab ini mengandungi tiga tahap talian paip yang berlainan fungsi. Tahap pertama melaksanakan penjanaan dedarab dan penambahan dedarab secara simpan-bawa dan penambahan eksponen. Ia menggunakan algoritma Booth tertib kedua dan pepohon Wallace dengan pemampat 4-2. Tahap kedua pula melakukan penambahan bawaan rambatan akhir, penormalan mantisa dan pembetulan eksponen manakala tahap ketiga mengandungi pembundaran, penormalan semula mantisa dan pembetulan semula eksponen. Dengan menggunakan kekayaan algoritma dalam VHDL, peralatan sintesis serta peralatan pemetaan, ianya dapat membantu dalam masalah merekabentuk, membuat penganalisaan samada melalui skematik, gelombang pemasaan atau tinjauan isyarat dan pembolehubah. Selain daripada itu ia juga dapat membina cip yang 'right at first time'.

Katakunci: ***VHDL, Pendarab titik apungan, Aritmetik komputer***

1.0 PENGENALAN

Unit titik apungan (UTA) atau dikenali sebagai ko-pemproses berangka, merupakan komponen utama atau litar khas dalam pemecut grafik, pemproses isyarat digital (PID) dan sistem komputer berkelajuan tinggi. Ia mengandungi beberapa set arahan khas yang difokuskan ke keseluruhan operasi matematik untuk memanipulasi angka dengan cepat dan tepat.

UTA boleh dibina samada dalam cip yang berasingan (seperti Intel 80387 atau Motorola 68881) atau disepadukan dengan Unit Pemprosesan Pusat (UPP) seperti Pentium [1]. Kesepaduan ini disebabkan oleh peningkatan ketumpatan kesepaduan cip kerana kemajuan teknologi semikonduktor. Pemproses Intel 486DX merupakan pemproses Intel yang pertama menggunakan kesepaduan fungsian ko-pemprosesan matematik dalam-cip (*on-chip*); pemproses Intel sebelumnya menggunakan ko-pemproses matematik luar-cip (*off-chip*). Pemproses siri Pentium menggunakan UTA dalam-cip dengan lapan tahap talian paip dan fungsi terdawai kekal (*hardwired function*) untuk meningkatkan prestasi perhitungan matematik. Ini menjadikan pemproses Pentium (1110/133) melaksanakan arahan titik apungan lima hingga sepuluh kali lebih pantas daripada Intel486 ™ DX2-66 [2]. Kebanyakan komputer peribadi pada hari ini, telah sedia-ada UTA, tetapi ia hanya untuk kegunaan tertentu seperti pemproses imej atau paparan. Komputer peribadi yang tiada UTA boleh mengendai perisian yang memerlukannya dengan memuatkan peniru titik apungan (floating point emulator) [3].

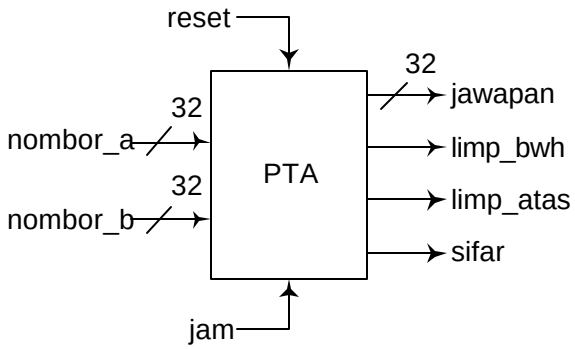
Pada hari ini, perisian aplikasi seperti SPSS, Matlab dan sebagainya memerlukan pemproses titik apungan yang berprestasi tinggi untuk mengendali pengiraan saintifik. Selain daripada itu, ia juga penting untuk komputer grafik, komputer set arahan terkurang, pemproses isyarat digital dan sebagainya. Ini disebabkan oleh peningkatan dalam permintaan terhadap penggunaan peralatan multimedia terutamanya aplikasi-aplikasi grafik tiga-dimensi (3D) masa nyata yang interaktif, di mana ia sedia untuk digunakan walaupun pada komputer peribadi. Dalam proses penukaran geometri untuk aplikasi grafik 3D, terlalu banyak operasi titik apungan diperlukan dan pemprosesan data selari. Bilangan operasi aritmetik dalam aplikasi grafik 3D masa nyata bergantung kepada kualiti objek 3D dan kadar kerangka (*frame rate*). Sebagai contoh, dalam kes skrin resolusi 1600 × 1280 dan 60 kerangka sesaat, jika objek 3D dibentuk dengan poligon yang kecil (25 piksel per poligon) dan 80% daripada keluasan skrin memproses objek 3D secara berterusan, dianggarkan 4 juta poligon sesaat diproses. Untuk prestasi penukaran geometri 3D bagi 1 juta

poligon sesaat memerlukan prestasi aritmetik titik apungan 250 hingga 300 MFLOPS [4]. Oleh itu untuk 4 juta poligon sesaat, ia memerlukan perhitungan titik apungan 1 GFLOPS atau lebih. Untuk memenuhi keperluan diatas, adalah penting sesebuah pemproses titik apungan itu berkelajuan tinggi dan bersaiz kecil.

Secara keseluruhannya pembinaan pemproses titik apungan banyak bergantung kepada bagaimana pendarab, penambah/ penolak dan pembahagi titik apungan direkabentuk terutamanya untuk operasi berfrekuensi tinggi (*high clock operation*) yang merupakan kunci untuk meningkatkan prestasi sistem terutamanya dalam pemprosessan imej dan pemprosesan isyarat digital. Pada khususnya, PTA 32bit atau 64bit merupakan komponen terbesar dalam laluan data dan merupakan salah satu elemen yang menentukan prestasi pemproses dan ia juga sangat rumit untuk dibina secara skematik atau pada tahap transistor. Namun begitu pada hari ini pendarab titik apungan bertalian paip yang berkelajuan tinggi boleh di bina dengan menggunakan bahasa perihalan perkakasan (BPP) iaitu VHDL (*Very high speed integrated circuit Hardware Description Language*) yang boleh disintesiskan dan dilaksanakan dalam tatasusunan get teraturcara medan (TGTM) dengan menggunakan peralatan Synopsys dan Xilinx.

2.0 SENIBINA

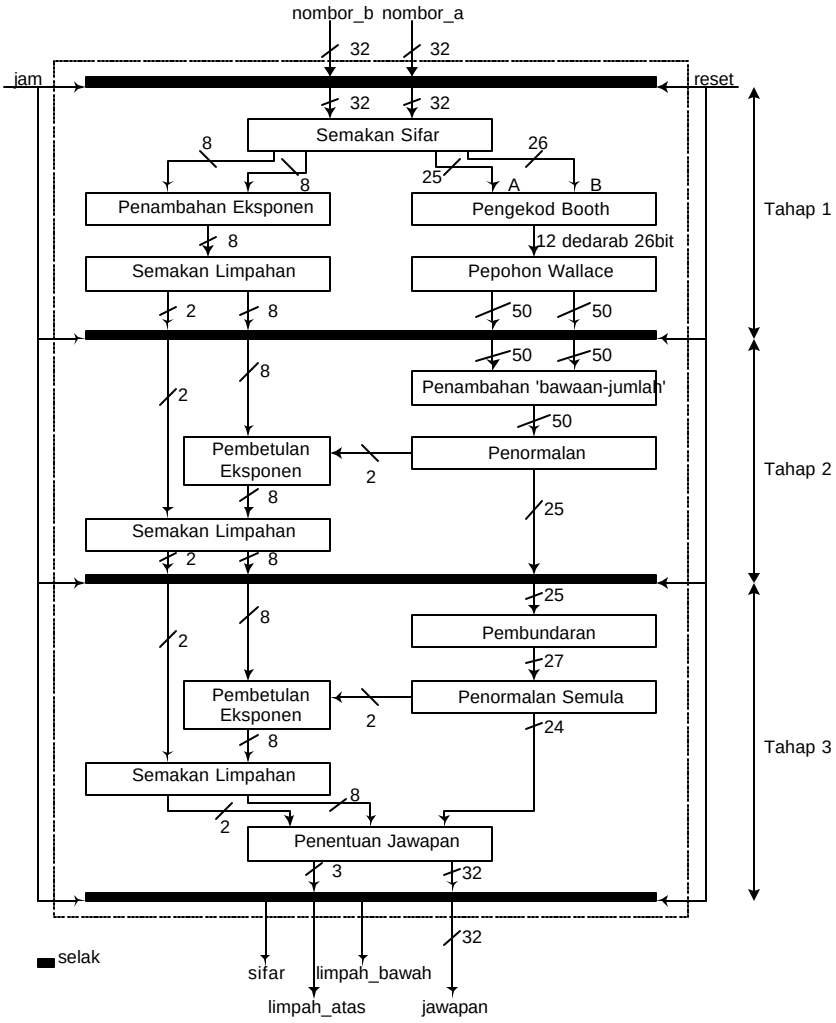
Pembinaan bermula dengan pembangunan rekabentuk teras (*core design*) seperti Rajah 1 dan Program 1. Semua data yang berkaitan adalah dalam jenis piawai IEEE iaitu pakej `std_logic_1164`.



Rajah 1: Rekabentuk teras

```
--Program 1: Isyarat Masukan dan Keluaran
ENTITY pdrbta IS
  PORT(jam             : IN std_logic;
       reset           : IN std_logic;
       nombor_a        : IN std_logic_vector(32 downto 1);
       nombor_b        : IN std_logic_vector(32 downto 1);
       sifar           : OUT std_logic;
       limpah_atas     : OUT std_logic;
       limpah_bawah    : OUT std_logic;
       jawapan         : OUT std_logic_vector(32 DOWNTO 1));
END pdrbta;
```

Senibina ini (rujuk Rajah 2) mengandungi tiga tahap talian paip yang menggunakan flip-flop tak-segerak [5, 6]. Rekabentuk tiga kitaran dipilih untuk cuba meningkatkan kelajuan jam. Tahap pertama mengandungi penambahan eksponen dan semakan limpahan yang dilaksanakan serentak dengan pengekod Booth dan pepohon Wallace. Tetapi sebelum itu semakan sifar dilaksanakan terlebih dahulu. Manakala tahap kedua mengandungi operasi penambahan 'bawaan-jumlah' yang terhasil daripada pepohon Wallace pada penghujung kitaran pertama, pernormalan beserta pembetulan eksponen dan semakan limpahan eksponen. Tahap ketiga pula terdiri daripada operasi pembundaran, penormalan semula, pembetulan eksponen, semakan limpahan eksponen dan penentu keluaran.



Rajah 2: Organisasi pendarab titik apungan 32bit

2.1 Tahap Pertama

2.1.1 Semakan Sifar

Program 2 melaksanakan pengesanan atau penyemakan samada data masukan yakni *p* dan *q* berisyarat datasifar yakni "00000000 00000000 00000000 00000001" di mana bit 1 ke 8 ialah eksponen dan bit 31 ke 9 adalah mantisa dan bit ke32 adalah bit-tanda.

```
--Program 2: Semakan sifar
--semakan untuk nombor_a
a_sifar1 <='1' when p = datasifar else '0';
--semakan untuk nombor_b
b_sifar1 <='1' when q = datasifar else '0';
```

Selain daripada semakan sifar ia juga melaksanakan perlanjutan bit-tanda untuk kedua-dua mantisa, di mana untuk *mantisa_p* mempunyai dua perlanjutan bit tanda dan *mantisa_q* tiga perlanjutan bit-tanda dengan satu bit 0 dikenakan pada bit kurang bererti seperti dalam Program 3.

```
--Program 3 : Perlanjutan bit-tanda
--asem dan bsem ialah songsangan bit-tanda
asem <= '0' when a_sifar1 = '1' else not p(32);
bsem <= '0' when b_sifar1 = '1' else not q(32);
r <=  p(32) & asem & p(31 downto 9);              --mantisa_p
s <=  q(32) & q(32) & bsem & q(31 downto 9) & '0'; --mantisa_q
```

2.1.2 Penambahan Eksponen dan Semakan Limpahan

Program 4 menunjukan operasi penambahan eksponen dengan menggunakan fungsi yang sedia-ada dalam pengkompilasi VHDL Synopsys [7].

```
--Program 4: Program Penambahan eksponen
jeks1 := eks_a  + eks_b - "01111111";
```

Persamaan dalam Program 4 juga melakukan penolakan nilai pincang 127 atau "01111111". Hasiltambah eksponen tersebut seterusnya disemak untuk memeriksa samada limpahan wujud atau tidak. Semakan limpahan ditentukan dengan menganalisa kesamaan bit kelapan *eks_a, eks_b* dan *jeks1* atau *jeks1* = "00000001" seperti dalam Program 5.

```
--Program 5: Semakan Limpahan pada Tahap Pertama
if ((eks_a(7) = eks_b(7)) and (eks_a(7) /= jeks1(7))) or
     (jeks1 = "00000001") then
    lats1 := not eks_a(7); --Limpahan Atas
    lbwh1 := eks_a(7);     --Limpahan Bawah
end if;
```

2.1.3 Pengekod Booth

Program 6 merupakan aturcara untuk pengekod Booth di mana d_m ialah *s*, x_m ialah *r* (*r* dan *s* dari Program 3), *x_k* ialah dedarab 26bit dan *n_1* ialah satu isyarat untuk menghasilkan dedarab *Dd* dalam pelengkap duaan. Program 7 pula merupakan aturcara pembinaan 12 pengekod Booth, di mana keluaran *Dd* merupakan data dalam bentuk matrik 1x26.

```
--Program 6: Pengekod Booth
case d_m is –-d_m ialah data masukan tiga bit

   when "001" | "010" =>                              --fungsi +r
    x_k <= x_m(24)&x_m;
    n_1 <= '0';
   when "011" =>                                      --fungsi +2r
    x_k <= x_m&'0';
    n_1 <= '0';
   when "100" =>                                      --fungsi –2r
    for indek in 1 to 25 loop
    x_k(indek) <= not x_m(indek - 1);
    end loop;
    x_k(0) <= '1';
    n_1 <= '1';
   when "101" | "110" =>                              --fungsi -r
    for indek in 0 to 24 loop
    x_k(indek) <= not x_m(indek);
    end loop;
    x_k(25) <= not x_m(24);
    n_1 <= '1';
   when others =>                                     --sifar
    x_k <= (x_k'range => '0');
    n_1 <= '0';
 end case;
```

```
--Program 7: Aturcara untuk penjanaan dedarab
booth (r, s(2 downto 0), t(1), Dd(1)(25 downto 0));
booth (r, s(4 downto 2), t(2), Dd(2)(27 downto 2));
booth (r, s(6 downto 4), t(3), Dd(3)(29 downto 4));
booth (r, s(8 downto 6), t(4), Dd(4)(31 downto 6));
booth (r, s(10 downto 8), t(5), Dd(5)(33 downto 8));
booth (r, s(12 downto 10), t(6), Dd(6)(35 downto 10));
booth (r, s(14 downto 12), t(7), Dd(7)(37 downto 12));
booth (r, s(16 downto 14), t(8), Dd(8)(39 downto 14));
booth (r, s(18 downto 16), t(9), Dd(9)(41 downto 16));
booth (r, s(20 downto 18), t(10), Dd(10)(43 downto 18));
booth (r, s(22 downto 20), t(11), Dd(11)(45 downto 20));
booth (r, s(24 downto 22), t(12), Dd(12)(47 downto 22));
booth (r, s(26 downto 24), t(13), Dd(13)(49 downto 24));
```

2.1.4 Pepohon Wallace

Program 8 menunjukkan aturcara perlanjutan bit-tanda, perlanjutan bit-sifar untuk setiap dedarab dan Program 9 melaksanakan susun-atur keseluruhan bit supaya membentuk data bermatrik 13 x 1 untuk digunakan dalam Program 10 yang merupakan sebahagian daripada aturcara sambungan antara penambah 16-2 [8]. Isyarat *semz* dan *t* adalah masukan dan *bw* dan *jum* adalah keluaran manakala *bkisy* ialah bawaan-keluar dan *cryisy* ialah bawaan.

```
--Program 8: Perlanjutan bit-tanda dan perlanjutan bit-sifar
--Bit-tanda
--m = 25 dan n = 26
lanjutant : for k in 1 to n/2 generate
                lanjtanda :for i in m+n-1 downto m+k*2-1 generate
                           Dd(k)(i) <= Dd(k)(m+k*2-2);
                           end generate;
            end generate;
--Bit-sifar
lanjutans : for k in 1 to n/2 - 1 generate
                lanjsifar :for i in 0 to k*2-1 generate
                           Dd(k+1)(i) <= '0';
                           end generate;
            end generate;


--Program 9: Susun-atur dedarab dalam bentuk matrik 13×1
--l = m+n
kolum: for kol in 1 to l-1 generate
          gen : for z in 1 to n/2 generate
                semz(kol)(z) <= Dd(z)(kol-1);
                end generate;
       end generate;


--Program 10 : Pepohon Wallace
p16_2 (semz(1), t(1), bkisy(0), cryisy(0), bkisy(1), cryisy(1), 1
                   bw(1), jum(1));
p16_2 (semz(2), '0', bkisy(1), cryisy(1), bkisy(2), cryisy(2),
                   bw(2), jum(2));
p16_2 (semz(3), t(2), bkisy(2), cryisy(2), bkisy(3), cryisy(3),
                   bw(3), jum(3));
.
.
.
p16_2 (semz(48), '0', bkisy(47), cryisy(47), bkisy(48), cryisy(48),
                   bw(48), jum(48));
```

```
p16_2 (semz(49), '0', bkisy(48), cryisy(48), bkisy(49), cryisy(49),
                  bw(49), jum(49));
p16_2 (semz(50), '0', bkisy(49), cryisy(49), bkisy(50), cryisy(50),
                  bw(50), jum(50));
```

Program 7 hingga 10 boleh digabungkan menjadi Program 11.

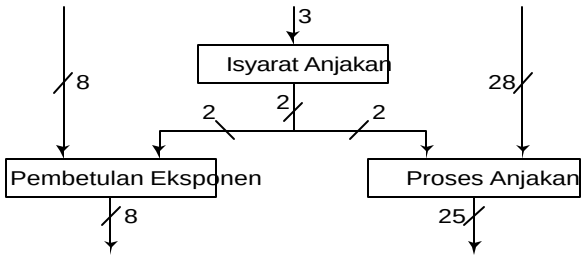
```
--Program 11: Gabungan program 7 hingga 10
dedarab : for k in 1 to n/2 generate
          booth   (r,   s(k*2   downto   k*2-2),   t(2*k-1),   dd(k)(m+k*2-2   downto   k*2-
2));
               lanjtanda :for i in m+n-1 downto m+k*2-1 generate
                          dd(k)(i) <= dd(k)(m+k*2-2);
                          end generate;
          t(2*k) <= '0';
          end generate;
lanjutan : for k in 1 to n/2 - 1 generate
               lanjsifar :for i in 0 to k*2-1 generate
                          dd(k+1)(i) <= '0';
                          end generate;
           end generate;
lanjplkp : for i in 27 to l generate
           t(i) <= '0';
           end generate;

    kolum: for kol in 1 to l generate
      satunit : block
          signal semz : std_logic_vector(13 downto 1);
        begin
          gen : for z in 1 to n/2 generate
                semz(z) <= dd(z)(kol-1);
                end generate;
           p16_2    (semz,    t(kol),    bkisy(kol-1),    cryisy(kol-1),    bkisy(kol),
cryisy(kol),
                  bw(kol), jum(kol));
               end block;
           end generate;
```

2.2 Tahap Kedua

Penambahan Bawaan dan Jumlah Program 12 menunjukkan penambahan jumlah dan bawaan di mana isyarat *jumlah(1)* tidak perlu melakukan penambahan kerana ia sudah merupakan bit hasildarab dan bawaan yang kelimapuluh yakin *bawaan(50)* diabaikan. Isyarat *man_1* merupakan hasildarab (jawapan kepada pendaraban) yang mengandungi 50bit.

```
--Program 12: Penambahan Bawaan dan Jumlah
man_1(50 downto 2) <= jumlah(50 downto 2) + bawaan(49 downto 1);
man_1(1) <= jumlah(1);
```



Rajah 3: Penormalan dan pembetulan eksponen

2.2.1 Penormalan

Setelah mendapatkan hasildarab tersebut, bit48 hingga ke bit50 dalam isyarat *man_1* digunakan untuk mendapat isyarat *anjakan_1* (rujuk Program 13) bagi menentukan anjakan yang diperlukan dalam proses anjakan dan pembetulan eksponen yang dijalankan serentak (lihat Rajah 3). Isyarat *man_norm_1* merupakan isyarat keluaran setelah proses anjakan (rujuk Program 14) dilaksanakan dan bilangan bitnya ialah 25bit, di mana bit ke25 ialah bit-tanda. Program-program ini diguna semula dalam proses penormalan semula pada tahap ketiga.

```
--Program 13: Isyarat anjakan_1
anjakan_1 <= "10" when (man_1(50) xor man_1(49)) = '1' else
             "01" when (man_1(49) xor man_1(48)) = '1' else
             "00";


--Program 14: Proses anjakan
with anjakan_1 select
    man_norm_1 <= man_1(50) & man_1(48 downto 25) when "10",
                  man_1(49) & man_1(47 downto 24) when "01",
                  man_1(48) & man_1(46 downto 23) when others;
```

2.2.2 Pembetulan Eksponen dan Semakan Limpahan

Pembetulan eksponen (Program 15) merupakan penambahan antara hasiltambah eksponen yang didapati daripada tahap pertama iaitu *hsl_tamb_eksp* dengan nilai *anjakan_1* yang menghasilkan *jeks2*. Seterusnya nilai eksponen yang diperbetulkan disemak semula untuk mengesan limpahan yang wujud. Untuk limpahan atas tahap kedua, ia meyemak samada terdapat isyarat limpahan atas dari tahap pertama atau mengesan kesamaan bit ke lapan bagi isyarat *jeks2* dan *hsl_tamb_eksp* supaya ia berisyarat 0. Manakala limpahan bawah menggunakan teknik songsangan limpahan bawah iaitu ia mengesan satuasi yang bukan menyebabkan limpahan bawah seperti dalam Program 16. Teknik ini juga digunakan dalam pembetulan eksponen dan semakan limpahan pada tahap ketiga.

```
--Program 15: Pembetulan Eksponen
jeks2 := (hsl_tamb_eksp) + ("000000" & anjakan_1);

--Program 16: Semakan Limpahan Tahap Kedua
--lbwheks2 & latseks2 merupakam isyarat limpahan_atas & limpahan _bawah --
daripada tahap pertama
--Limpahan Atas
if (lbwheks2 = '0') and ((latseks2 = '1') or
    ((jeks2(7) /= hsl_tamb_eksp(7)) and (hsl_tamb_eksp(7) = '0'))) then
     lats2 := '1';
--Limpahan Bawah
elsif (((jeks2(7)) /= hsl_tamb_eksp(7)) and
       (hsl_tamb_eksp(7) = '0') and (lbwheks2 = '1')) or
         ((hsl_tamb_eksp = "10000000") and (anjakan_1 /= "00")) then
     lbwh2 := '0';
end if;
```

2.3 Tahap Ketiga

2.3.1 Pembundaran

Proses pembundaran menggunakan skim pembundaran terdekat seperti dalam Program 17.

```
--Program 17: Proses Pembundaran
IF(bit_bundar = '0') THEN
   man_2(24 DOWNTO 1) <= man_norm_1(25 DOWNTO 2);
   man_2(27 downto 25)<= man_norm_1(25)&man_norm_1n(25)&man_norm_1(25);
ELSIF(bit_bundar = '1') THEN
   man_2(27 DOWNTO 1) <= man_norm_1(25)&man_norm_1(25)&man_norm_1(25)&
```

```
                man_norm_1(25 downto 2) + "00000000000000000000000001";
END IF;
```

2.3.2 Penentu Isyarat Jawapan

Program 18 merupakan aturcara untuk perihalan penentu isyarat jawapan. Ia bermula dengan isyarat sifar, jika *nombor_a* atau *nombor_b* adalah sifar maka jawapan yang diberikan ialah sifar iaitu "00000000 00000000 00000000 00000001". Sekiranya wujud limpahan atas, terdapat dua jawapan iaitu untuk nombor negatif dan positif, sekiranya positif, jawapan yang diberikan ialah nilai positif yang maksima iaitu "01111111 11111111 11111111 11111110" dan sekiranya negatif, jawapan ialah nilai negatif yang maksima iaitu "10000000 00000000 00000000 11111110".

```
--Program 18: Penentu isyarat jawapan
--isyarat sifar
if (a_sifar3 = '1') or (b_sifar3 = '1') then
               isy_sifar <= '1';
               jwp <= "00000000"&"00000000"&"00000000"&"00000001";
--isyarat limpahan atas
elsif (latseks5 = '1') then
               isy_latas <= '1';
                --nombor positif
                if (man_norm_2(23) = '0') then
                        jwp <= "01111111"&"11111111"&"11111111"&"11111110";
                --nombor negatif
                else
                        jwp <= "10000000"&"00000000"&"00000000"&"11111110";
                end if;
--isyarat limpahan bawah
elsif (lbwheks5 = '1')  or ((eks_norm_2 = "00000001") and
       (man_norm_2 /= "00000000"&"00000000"&"00000000")) then
                isy_sifar <= '1'; isy_lbawah <= '1';
                jwp <= "00000000"&"00000000"&"00000000"&"00000001";
--selainnya
else
                jwp <= man_norm_2&eks_norm_2;
end if;
```

3.0 KEPUTUSAN

Jadual 1 menunjukkan pemasaan yang didapati setelah melakukan penempatan dan penghalaan dengan menggunakan peralatan Xilinx Alliance dan pustaka xc4036xl-bg432-2. Jadual 2 pula memberikan maklumat mengenai bilangan komponen TGTM yang digunakan manakala Rajah 4 menunjukkan gambarajah pelan-lantai (*floorplan*) untuk pendaraban titik apungan 32bit bertalian paip.

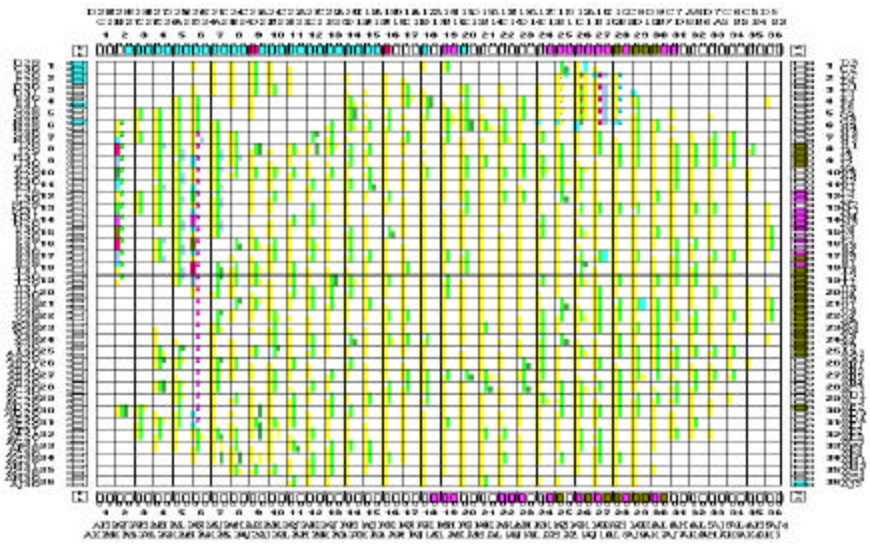
PTA ini lebih laju jika dibandingkan dengan dengan projek di dalam [9] yang menggunakan teknologi 4 Actel A1280 FPGA, di mana ia memerlukan 3 tahap talian paip yang memerlukan masa kitaran 245ns di mana pendarabnya memerlukan enam pendaman iaitu satu untuk eksponen, empat untuk pendaraban dan satu lagi untuk penormalan.

Jadual 1: Pemasaan bagi jam

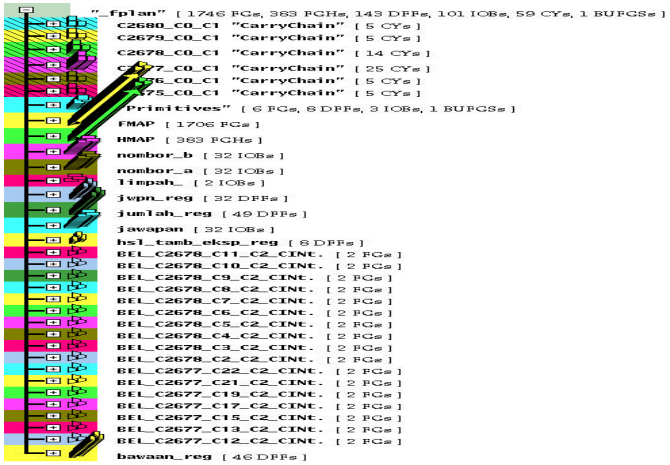
Tempoh minimum	54.135ns
Frekuensi maksima	18.472MHz

Jadual 2: Bilangan komponen yang digunakan

Bilangan IOB luaran	100 IOB daripada 288	34%
Flip flop	99	
Selak	0	
Bilangan *Global Buffer IOB*	1 daripada 8	12%
Flip-flop	0	
Selak	0	
Bilangan CLB	1007 CLB daripada 1296	77%
Jumlah selak	0 daripada 2592	0%
Jumlah CLB flip-flop	143 daripada 2592	5%
LUT 4 masukan	1746 daripada 2592	67%
LUT 3 masukan	383 daripada 1267	29%
Bilangan BUFGLS	1 daripada 8	12%
Bilangan STARTUP	1 daripada 1	100%



(a)



(b)

Rajah 4: Pelan-lantai (a) beserta petunjuk (b) untuk pendaraban titik apungan 32bit bertalian paip

4.0 RUMUSAN

Terdapat dua objektif utama iaitu membina dan merekabentuk pendarab titik-apungan 32bit bertalian paip yang boleh disintesiskan menggunakan VHDL dan ia beroperasi pada kelajuan tinggi. Objektif pertama dapat dicapai dengan terbinanya PTA 32bit bertalian paip (tiga tahap) dengan struktur pendarab selari pepohon Wallace. Ia menggunakan 1007 CLB dan 100 IOB di mana peranti pemetaan yang digunakan ialah xc4036xl-bg432-2 daripada pustaka XC4000 Xilinx FPGA dan boleh disesuaikan dalam UTA untuk enjin grafik 3D, perhitungan saintifik dan aplikasi-aplikasi masa nyata.

Pada asalnya, sasaran frekuensi ialah 50MHz, namun begitu setelah tamat peringkat sintesis ia hanya berupaya mencapai 7MHz. Maka frekuensi sesaran (frekuensi kekangan) dikurangkan ke 25MHz dengan harapan ia boleh menghampiri sesaran tersebut. Sungguhpun begitu ia berjaya menghampiri 25MHz pada proses penempatan dan penghalaan dengan capaian 18.472MHz. Di sini boleh disimpulkan bahawa walaupun struktur pendarab yang digunakan berprestasi tinggi namun ia bergantung juga kepada teknologi yang digunakan untuk perlaksanaan. Ini bermakna pencapaian-percapaian seperti dalam [4, 10, 11, 12] mahupun lebih daripada itu tidak mustahil diperolehi sekiranya ada pendedahan teknikal atau praktikal untuk merekabentuk cip dalam teknologi canggih yang sama dan kemudahan mendapatkan bahan teknikal atau praktikal supaya persaingan dan perbandingan boleh dibuat dalam situasi yang sama.

Kelebihan utama menggunakan VHDL ialah ia senang untuk menyiasat dan menerokai senibina rekabentuk yang direka. Perihalan domain berstruktur dalam VHDL membolehkan seseorang itu memperihalkan senibina dengan cara semulajadi, yakni dalam ertikata penyambungan set komponen-komponen. Dengan pendekatan ini, ia membolehkan pereka menumpukan sepenuh perhatian kepada pengoptimuman rekabentuk. Terbitan pemasaan juga senang digabungkan ke dalam rekabentuk selagi ia wujud dalam format yang dibolehkan oleh VHDL.

Boleh disimpulkan disini bahawa VHDL merupakan alat untuk membina perlaksanaan dalam silikon dan alat untuk berfikir dengannya, di mana dalam keadaan tertentu ia adalah lebih baik daripada algoritma bahasa komputer lazim.

5.0 KERJA SELANJUTNYA

Sejak kebelakangan ini, System-on-Chip (SoC) memainkan peranan yang penting dalam perdagangan produk elektronik, maka dengan itu, pengalaman ini digunakan dalam penyelidikan dalam keseimbangan keberkesanan penggunaan metodologi-metodologi merekabentuk baru (seperti VHDL), kepenggunaan-semula harta intelek, peralatan automasi rekabentuk dan kepiawaian untuk pembinaan SoC. Paradigma SoC merupakan satu pengembangan rekabentuk ASIC daripada peringkat komponen kepada peringkat sistem ataupun realisasi kefungsian sistem dalam satu cip yang besar seperti *scanner on chip* yang dihasilkan oleh National Semiconductor. Selain daripada paradigma SoC, kajian dalam paradigma SoP (*System on Package*) juga boleh dipertimbangkan. Persamaan dan perbezaan diantara SoP dan SoC boleh didapati daripada [13].

RUJUKAN

[1] Cnet Hongkong Glossary, *Floating Point Unit,* http://hongkong1.cnet.com/Briefs/Glossary/Terms/fpu.html Cnet Hong Kong, Nov 1999.

[2] Intel Corporation, *High Performance Floating Point Unit, http://homepage.eusica.fr/~frances/intel/procs/pentium/pptech/floatpt.html, Intel Corporation, 1995.*

[3] Chuck Purdy, *Floating Point Unit, http://www.whatis.com/fpu.html, whatis.com, Oct 1999.*

[4] G. Goto et. al, "A 4.1ns Compact 54×54-bit Multilier Utilizing Sign-Select Booth Encoder". *IEEE J. Solid State Circuit*, Vol. 32, Nov. 1997, pp. 1676-1682.

[5] Peter J. Ashenden, *The Designer's Guide in VHDL.* Morgan Kaufmans, 1996.

[6] *Guidelines and Practices for Successful Logic Synthesis*, Version 1997.08 Synopsis.

[7] *VHDL Compiler Reference Manual*. Version 1997.08, Synopsis.

[8] M. Nagamatsu, S. Tanaka, J. Mori, K. Hirano and T. Noguchi, "A 15ns 32 x 32 bit CMOS Multiplier with Improved Parallel Structure". *IEEE J. Solid State Circuit*, Vol. 25, Apr. 1990, pp. 494-497.

[9] Barry Fagin and Cyril Renard, "Field Programmable Gate Array and Floating Point Arithmetic". *IEEE Trans. On VLSI Systems*, Vol. 2, No. 3, Sept. 1994, pp. 365-367.

[10] N. Ohkubo et al, "A 4.4ns CMOS 54 x 54 -bit Multiplier Using Pass Transistor Multiplexer". *IEEE J. Solid State Circuit*, Vol. 30, Mar. 1995, pp. 251-257.

[11] M. Hanawa et al, "A 4.3ns 0.3µm CMOS 54 x 54-bit Multiplier Using Precharged Pass-Transistor Logic". *ISSCC Dig. Tech. Papers*, Feb. 1996, pp. 364-365.

[12] Y. Hagihara et al, "A 2.7ns 0.25µm CMOS 54 x 54 -bit multiplier". *ISSCC Dig. Tech. Papers*, Feb. 1998, pp. 296-297, 449.

[13] Rao R. Tummala and Vijay K. Madisetti, "System on Chip or System on Package". *IEEE Design and Test Computer* , April-June 1999, pp. 48-54.

BIOGRAFI

Noorzaily Mohamed Noor dilahirkan di Kelantan, Malaysia, pada 29 September 1970. Beliau dianugerahkan Sarjana Muda Sains (Fizik) dan Sarjana Sains Komputer (Senibina Sistem Komputer) daripada Universiti Malaya pada 1995 dan 2000. Beliau pernah terlibat dalam pembangunan 'High Speed FPU' untuk pemproses grafik. Sekarang, beliau terlibat dalam penyelidikan dan pembangunan 'System on Chip (Soc)' dengan Mimos Bhd.

Mashkuri Hj. Yaacob adalah Professor Sains Komputer dan Timbalan Naib Canselor (Akademik), Universiti Malaya. Beliau menyertai Universiti Malaya pada 1976 dan telah menghasilkan lebih daripada 130 kertas penyelidikan dan membentangkannya pada persidangan tempatan dan antarabangsa. Beliau adalah ahli kepada 'IASTED Conference Organising Committee' dan 'IEEE Computer Society'. Bidang penyelidikan beliau ialah kejuruteraan perisian, dan senibina komputer yang meragumi senibina rangkaian ATM, litar pengurusan prestasi maya dan lain-lain lagi.