SYNTHESIZABLE RISC-V (RV32I) MICROPROCESSOR BAGIAN II : CONTROL UNIT (CU), IMMEDIATE SELECTOR, BRANCHER, PROGRAM COUNTER (PC), 4-ADDER, DAN 2-TO-1 MUX GENERIK

William Anthony (13223048)
Asisten : Abraham Pratomo (23223071)
Tanggal Percobaan : 19/11/2025
KELOMPOK 14
EL3011 Arsitektur Sistem Komputer
Laboratorium Sinyal dan Sistem – Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

***Abstrak*—Praktikum ini berfokus pada memahami arsitektur mikroprosesor RISC-V (RV32I) dan merealisasikan beberapa komponen pentingnya dalam HDL yang synthesizable dan dapat disimulasikan. Komponen yang direalisasikan meliputi control unit (CU), immediate selector, brancher, program counter (PC), 4-adder, dan 2-to-1 generic mux. Implementasi dilakukan menggunakan bahasa Verilog HDL dengan fokus pada simulasi fungsional dan timing. Hasil analisis timing menunjukkan worst-case delay sebesar 18.540 ns membatasi frekuensi operasi maksimum hingga sekitar 53.94 MHz.**

***Kata Kunci*—RISC-V, RV32I, Control Unit, Immediate Selector, Brancher, Program Counter, Verilog HDL**

I. Pendahuluan

Praktikum SYNTHESIZABLE RISC-V (RV32I) MICROPROCESSOR BAGIAN II : CONTROL UNIT (CU), IMMEDIATE SELECTOR, BRANCHER, PROGRAM COUNTER (PC), 4-ADDER, DAN 2-TO-1 MUX GENERIK berfokus pada pada perancangan dan implementasi beberapa komponen kunci yang mendukung tahap instruction fetch dan instruction decode, serta kendali aliran data. Pemahaman mendalam tentang arsitektur RISC-V dan datapath eksekusinya adalah prasyarat untuk berhasil merealisasikan komponen-komponen ini. Tujuannya agar Hasil yang diharapkan adalah implementasi fungsional dari komponen-komponen tersebut yang telah disimulasikan secara fungsional dan timing menggunakan Quartus II dan ModelSim.

II. Landasan Teoretis

A. Bahasa Verilog HDL

Verilog HDL adalah bahasa deskripsi perangkat keras (hardware description language) untuk mendesain rangkaian digital. Dalam Verilog, sinyal direpresentasikan sebagai wire (koneksi kombinatorial) dan *reg* (penyimpanan/variabel yang di-assign di blok sekuensial atau kombinatorial). Verilog bersifat *case-sensitive*. Implementasi fungsi dapat dilakukan dengan continuous assignment (*assign*) untuk logika kombinasional atau blok always untuk logika kombinasional atau sekuensial. Setiap entitas desain disimpan dalam file Verilog HDL. Desain ditulis sebagai module yang dapat diinstansiasi oleh module lain untuk membentuk rangkaian yang lebih besar. Setiap desain-desain Verilog ditulis sebagai module yang dapat diinstansiasi oleh module lain untuk membentuk rangkaian yang lebih besar (top-level module). [1][3]

Terdapat tiga jenis concurrent signal assignment (CSA):

I. Simple CSA. Assignment sinyal dilakukan dengan ekspresi logika biasa. Hasil implementasi Simple CSA akan berupa gerbang logika biasa.

II. Conditional CSA. Assignment sinyal dilakukan dengan construct WHEN-ELSE. Hasil implementasi Conditional CSA akan berupa kumpulan 2-to-1 multiplexer yang disusun secara bertahap dengan boolean_expr sebagai selektor dan value_expr sebagai nilai sinyal yang dapat dipilih.

III. Selected CSA. Assignment sinyal dilakukan dengan construct WITH-SELECT. Hasil implementasi Selected CSA akan berupa satu buah n-to-1 multiplexer dengan select_expression sebagai

selektor dan value_expr_3 sebagai nilai sinyal yang dapat dipilih.

Dalam Verilog, elemen memori direalisasikan juga dengan blok always. D Latch bersifat level-sensitive (keluaran mengikuti masukan saat enable aktif) dan tidak direkomendasikan pada FPGA karena menyebabkan latch inference yang merugikan analisis timing. D Flip-Flop (DFF) bersifat edge-triggered dan menjadi dasar implementasi register. Register adalah sekumpulan DFF yang menyimpan suatu vektor data dan banyak digunakan untuk rangkaian sekuensial seperti (FSM) finite state machine maupun register file. Untuk DFF, gunakan non-blocking assignment (<=) di blok always @(posedge clk); untuk logika kombinatorial gunakan always @(*) dengan blocking assignment (=). [1][3]

Contoh Implementasi Mux dalam Verilog adalah pada gambar 1 dalam lampiran. Akan membuat seperti Fig. 1.

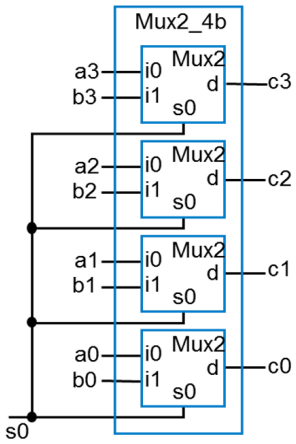


Fig. 1. Mux2_4b di Verilog

B. *Altera® Quartus® II*

Intel® Quartus® Prime (sebelumnya Altera® Quartus® II) adalah perangkat lunak untuk desain FPGA. Proses desain dimulai dengan pembuatan proyek, pemilihan device, dan kompilasi. Kompilasi dapat berupa full compilation (termasuk Analysis & Synthesis, Fitter, dan Assembler) untuk analisis timing, atau hanya Analysis & Synthesis untuk simulasi fungsional. Untuk simulasi, menggunakan Mentor Graphics® ModelSim®. Simulasi bisa dilakukan pengesetan secara fungsional dan timing. Hal ini bertujuan untuk mengetahui delay yang dihasilkan untuk devais yang ditujukan. Frekuensi pada FPGA dapat dihitung dengan rumus f_max < 1 / (worst-case tpd). [2][3]

Fig. 2. Altera® Quartus® II

C. *RISC-V*

RISC-V merupakan arsitektur set instruksi (instruction set architecture, ISA) terbuka yang dikelola oleh RISC V International. RISC-V yang digunakan adalah basis 32-bit (RV32I, base integer ISA).
RISC-V banyak dipakai untuk pembelajaran arsitektur komputer karena spesifikasi intinya ringkas, konsisten, dan mudah diimplementasikan; di dunia nyata ia hadir dari mikrokontroler berdaya rendah hingga SoC tertanam.[1][2][3]

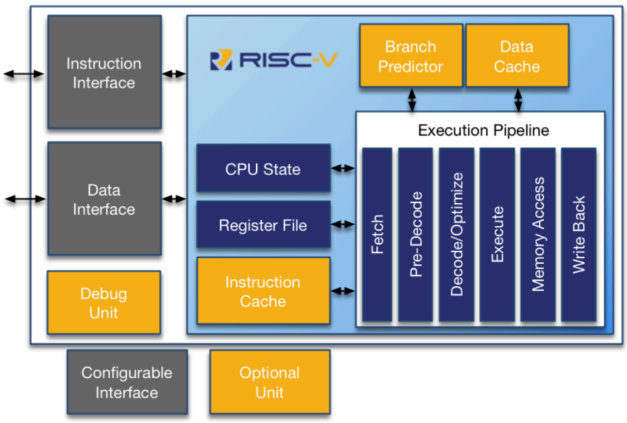


Fig. 3. RISC-V Diagram.

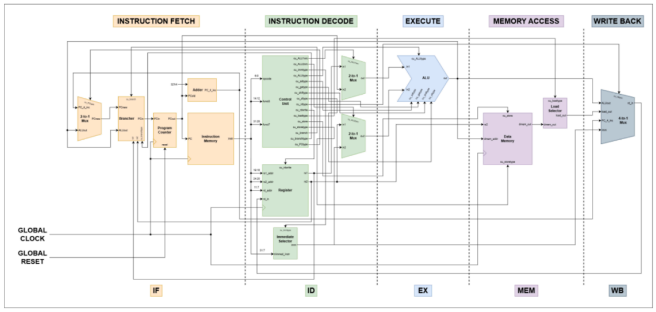


Fig. 4. Tahap-tahap Mikroprosesor RISC-V ketika melakukan eksekusi suatu instruksi.

D. Control Unit (CU)

Control Unit (CU) adalah komponen dalam mikroprosesor yang bertanggung jawab untuk mengkoordinasikan semua operasi.[4]

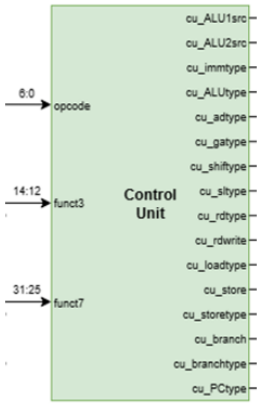


Fig. 5 Control Unit

CU menerima opcode, funct3, dan funct7 dari instruksi yang telah di-decode untuk menghasilkan sinyal-sinyal kontrol yang diperlukan oleh komponen lain dalam datapath. Dalam implementasi ini, CU akan mengeluarkan 16 sinyal kontrol yang bervariasi ukurannya, disesuaikan dengan kebutuhan setiap instruksi. Sinyal-sinyal ini mengendalikan berbagai aspek operasi, seperti pemilihan operand ALU, tipe operasi ALU, penulisan ke register file, mode akses memori data, dan kondisi branch.[4]

E. Immediate Selector

Arsitektur RISC-V (RV32I) menggunakan nilai konstanta (immediate value) sebagai operand dalam banyak instruksi. Nilai immediate ini memiliki format yang berbeda-beda tergantung pada tipe instruksinya (I-type, S-type, B-type, U-type, dan J-type). Immediate selector bertugas mengekstraksi bit-bit immediate dari instruksi yang sudah dipotong (trimmed instruction) dan melakukan sign-extension untuk menghasilkan nilai signed 32-bit yang sesuai. [4]

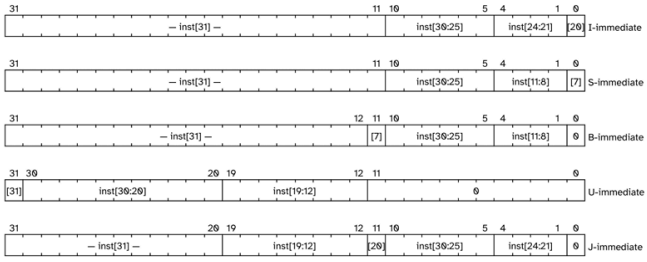


Fig. 6 Nilai immediate yang dihasilkan oleh masing-masing tipe instruksi

F. Brancher

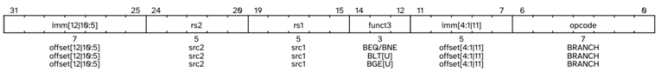


Fig. 7 Format instruksi Brancher

Instruksi branch kondisional (B-type) dalam arsitektur RISC-V (RV32I) akan membuat perubahan aliran kontrol program. Brancher membandingkan nilai dua register (rs1 dan rs2) dan, jika kondisi perbandingan terpenuhi, Program Counter (PC) akan diperbarui dengan alamat target branch (PC + signed offset). Jika kondisi tidak terpenuhi, PC akan di inkremen sebesar 4 (PC+4), melanjutkan eksekusi instruksi secara sekuensial. Terdapat enam jenis instruksi branch: BEQ, BNE, BLT, BLTU, BGE, dan BGEU.[4]

Instruksi BEQ (Branch if Equal) melakukan percabangan jika isi rs1 dan rs2 bernilai sama. Instruksi BNE (Branch if Not Equal) melakukan hal yang berlawanan, yaitu cabang diambil bila kedua register tidak memiliki nilai yang sama. Instruksi BLT (Branch if Less Than) digunakan untuk percabangan berdasarkan perbandingan nilai signed. Instruksi ini akan melakukan cabang jika nilai rs1 lebih kecil dari rs2 dengan mempertimbangkan sign bit. Sementara itu, BLTU (Branch if Less Than Unsigned) melakukan perbandingan yang sama namun dalam nilai unsigned, sehingga sign bit diabaikan dan nilai register dianggap selalu positif. Instruksi BGE (Branch if Greater or Equal) dan BGEU (Branch if Greater or Equal Unsigned), bekerja dengan prinsip kebalikan dari BLT dan BLTU. BGE melakukan cabang jika rs1 lebih besar atau sama dengan rs2 secara signed, sedangkan BGEU digunakan untuk perbandingan unsigned.[4]

III. Hasil dan Analisis

Dari melaksanakan tugas 1, tugas 2, tugas 3. Didapatkan sebagai berikut:

A. Tugas I : Control Unit (CU)

Mikroprosesor RISC-V membedakan jenis instruksi berdasarkan opcode (Operation Code) pada bit [6:0] dari instruksi 32-bit. Untuk instruksi-instruksi dengan opcode yang sama, seperti R-type dan beberapa I-type, perbedaan lebih lanjut ditentukan oleh bidang funct3 (bit [14:12]) dan funct7 (bit [31:25]). Misalnya, opcode 7'h33 mengidentifikasi instruksi R-type, sedangkan kombinasi unik dari funct3 dan funct7 di dalam opcode ini akan menentukan operasi spesifik seperti ADD, SUB, SLL, dan lainnya. Struktur fixed-field decoding ini akan membuat hardware untuk dengan cepat dan efisien mengidentifikasi tipe dan operasi instruksi.

Control Unit (CU) adalah otak dari mikroprosesor yang mengatur aliran data (datapath) dengan menghasilkan sinyal-sinyal kontrol berdasarkan interpretasi instruksi. Sinyal-sinyal ini mengarahkan perilaku berbagai komponen dalam datapath untuk menjalankan operasi yang benar. Komponen-komponen yang diatur oleh CU meliputi ALU (melalui sinyal cu_ALUtype, cu_adtype, cu_gatype, cu_shiftype, cu_ALU1src, dan cu_ALU2src), Register File (melalui cu_rdwrite dan cu_rdtype), Data Memory (melalui cu_store, cu_loadtype, dan cu_storetype), Program Counter (PC) dan Branch Unit (melalui cu_branch, cu_branchtype, dan

cu_PCtype), serta Immediate Generator (melalui cu_immtype).

Bagian-bagian instruksi RISC-V seperti opcode, funct3, funct7, rd, rs1, dan rs2 memiliki posisi bit yang tetap (fixed-field decoding). Keuntungan utama dari pendekatan ini adalah penyederhanaan hardware unit decode instruksi, peningkatan kecepatan decoding karena bidang-bidang dapat diakses secara paralel, dan efisiensi dalam implementasi pipeline yang meminimalkan stall. Meskipun bidangnya tetap, kombinasi opcode, funct3, dan funct7 membuat banyak instruksi yang berbeda untuk diwakili secara efisien.

Dapat diperhatikan saat fungsional (Dari Hasil Lampiran), Control Unit berhasil menghasilkan sinyal kontrol yang sesuai untuk serangkaian opcode dan funct. Misalnya, pada awal simulasi (sekitar 0-10 ns), ketika opcode adalah 33 (R-type) dan funct7 00 dengan funct3 0 (ADD), cu_rdwrite menjadi 1 dan cu_ALUtype tetap 00. Saat opcode menjadi 13 (I-type ALU) sekitar 10 ns, cu_ALU2src menjadi 1 dan cu_rdwrite tetap 1, dengan cu_immtype 000. Kemudian, saat opcode adalah 03 (Load) dan funct3 2 (LW) sekitar 20 ns, cu_rdtype berubah menjadi 01 dan cu_loadtype menjadi 010, menunjukkan akses memori. Pada opcode 23 (Store) dengan funct3 2 (SW), cu_store menjadi 1 dan cu_storetype menjadi 10. Ketika opcode 63 (Branch) dengan funct3 0 (BEQ) aktif, cu_branch menjadi 1 dan cu_branchtype menjadi 000, serta cu_PCtype menjadi 1. Terakhir, untuk opcode 6F (JAL) dan 37 (LUI), sinyal-sinyal kontrol juga berubah sesuai yang diharapkan, seperti cu_immtype yang berubah ke 100 untuk JAL dan 011 untuk LUI, serta cu_rdtype menjadi 10 untuk JAL dan 11 untuk LUI, menegaskan fungsionalitas yang benar dari Control Unit untuk berbagai instruksi.

Dari Timing Analyzer Summary, terlihat bahwa worst-case propagation delay (tpd) untuk Control Unit adalah 11.505 ns, dari opcode[3] ke cu_loadtype[1]. Nilai tersebut mendeskripsikan *critical path* atau jalur terpanjang dalam simulasi ini. Nilai delay dengan pengesetan interval waktu 10 ns menunjukkan waktu maksimum yang dibutuhkan sinyal untuk merambat dari input ke output dalam kondisi terburuk. Sehingga agar memastikan operasi yang stabil, frekuensi clock maksimum (f_max) yang digunakan harus lebih kecil dari invers dari delay terburuk ini.

$f_{max} < 1 / 11.505\ ns \approx 86.91\ MHz$

Maka FPGA yang menggunakan Control Unit ini harus beroperasi pada frekuensi clock maksimum (f_max) kurang dari sekitar 86.91 MHz. Jika frekuensi clock lebih tinggi dari ini, ada risiko kalau sinyal kontrol tidak akan stabil pada waktu yang tepat sebelum edge clock berikutnya, yang dapat menyebabkan kesalahan fungsional. Ini berarti frekuensi minimal yang harus didukung oleh FPGA adalah 86.91 MHz untuk memastikan unit kontrol dapat bekerja dengan baik pada worst-case delay tersebut.

B. Tugas 2 : Immediate Selector

Program Counter (PC) pada datapath mikroprosesor RISC-V dalam praktikum ini berfungsi sebagai register 32-bit yang menyimpan alamat memori dari instruksi yang akan dieksekusi selanjutnya. PC adalah komponen utama untuk mengontrol aliran eksekusi program. Pada setiap siklus clock, nilai PC saat ini digunakan untuk mengambil instruksi dari instruction memory. Setelah instruksi diambil, nilai PC di inkremen secara otomatis menjadi PC+4 (menggunakan 4-adder) untuk menunjuk ke instruksi berikutnya dalam urutan sekuensial. Namun, jika instruksi yang sedang dieksekusi adalah instruksi branch atau jump (yang ditentukan oleh control unit), nilai PC akan diubah menjadi alamat target branch atau jump yang dihitung oleh brancher atau ALU. Ini membuat program untuk mengubah aliran eksekusinya berdasarkan kondisi atau untuk melompat ke bagian lain dari kode. Pada saat reset, PC diinisialisasi ke nilai awal (misalnya, 32'h0), memulai eksekusi program dari alamat tersebut.

Mikroprosesor RISC-V menggabungkan bagian immediate yang terpecah-pecah melalui komponen immediate selector. Setiap tipe instruksi (I-type, S-type, B-type, U-type, dan J-type) memiliki format immediate yang unik dengan bit-bit yang tersebar di berbagai posisi dalam instruksi 32-bit. Immediate selector mengambil trimmed instruction (25 MSB dari instruksi) dan menggunakan sinyal cu_immtype dari control unit untuk menentukan cara menyusun ulang bit-bit immediate tersebut. Proses ini juga akan secara otomatis melakukan sign-extension, yaitu mereplikasi bit paling signifikan dari immediate ke bit-bit yang lebih tinggi untuk menghasilkan nilai signed 32-bit yang benar. Sebagai contoh, untuk I-type, bit trimmed_instr[24] diperluas ke 20 bit paling signifikan, dan trimmed_instr[24:13] membentuk bagian immediate. Untuk S-type, bit immediate diambil dari trimmed_instr[24:18] dan trimmed_instr[4:0], lalu di-sign-extend dengan trimmed_instr[24]. Untuk nilai signed 32-bit, nilai maksimumnya adalah 2^(31) - 1 = 2,147,483,647 dan nilai minimumnya adalah -2^(31) = -2,147,483,648.

Dapat diperhatikan bahwa pada percobaan fungsional yang ditunjukkan pada hasil waveform di lampiran untuk Immediate Selector, komponen ini berhasil mengekstrak dan memperluas tanda nilai immediate dengan benar untuk berbagai tipe instruksi. Tabel dan Screenshot percobaan dapat diamati pada lampiran. Ketika cu_immtype adalah 000 (I-type) dan trimmed_instr adalah 1FFE505, imm berhasil dihasilkan sebagai -1. Ini menunjukkan sign-extension yang akurat dari nilai immediate negatif. Selanjutnya, dengan cu_immtype 000 dan trimmed_instr 0018249, imm menjadi 12, yang mengkonfirmasi ekstraksi nilai positif yang benar. Ketika cu_immtype beralih ke 001 (S-type) dengan trimmed_instr 1FD2250, imm menjadi -16, menunjukkan penanganan immediate S-type yang benar. Untuk cu_immtype 010 (B-type) dan trimmed_instr 1FC0515, imm menjadi -12, yang sesuai dengan format immediate B-type. Pada

cu_immtype 100 (J-type) dengan trimmed_instr 00A0001, imm adalah 80, mengesahkan penanganan immediate J-type. Yang terakhir, untuk cu_immtype 011 (U-type) dengan trimmed_instr 0000254, imm menjadi 73728, ini menunjukkan ekstraksi dan pergeseran nilai immediate U-type yang tepat. Hasil-hasil ini secara kolektif menegaskan fungsionalitas yang benar dari Immediate Selector.

Dari Timing Analyzer Summary, terlihat bahwa worst-case propagation delay (tpd) untuk Immediate Selector adalah 10.891 ns, dari trimmed_instr[24] ke imm[23]. Nilai delay dengan pengesetan interval waktu 10 ns menunjukkan waktu maksimum yang dibutuhkan sinyal untuk merambat dari input ke output dalam kondisi terburuk, terutama pada jalur yang melibatkan sign-extension dari bit MSB trimmed_instr ke imm. Untuk memastikan operasi yang stabil, frekuensi clock maksimum (f_max) yang dapat digunakan harus lebih kecil dari invers dari delay terburuk ini.

f_max < 1 / 10.891 ns ≈ 91.82 MHz

Oleh karena itu, FPGA yang menggunakan Immediate Selector ini harus beroperasi pada frekuensi clock maksimum (f_max) kurang dari sekitar 91.82 MHz. Jika frekuensi clock lebih tinggi dari ini, ada risiko yakni sinyal keluaran imm tidak akan stabil pada waktu yang tepat sebelum edge clock berikutnya, yang dapat menyebabkan kesalahan fungsional. Ini berarti frekuensi minimal yang harus didukung oleh FPGA adalah 91.82 MHz untuk memastikan unit ini dapat bekerja dengan baik pada worst-case delay tersebut.

C. Tugas 3 : Brancher

Fungsi brancher pada datapath mikroprosesor RISC-V adalah untuk mengimplementasikan mekanisme percabangan kondisional (conditional branch) dan lompatan tanpa syarat (unconditional jump). Komponen ini menentukan alamat instruksi berikutnya (next PC) berdasarkan hasil perbandingan dua register dan tipe instruksi branch yang aktif.

Ketika program mengalami branch, Control Unit (CU) mendeteksi instruksi branch atau jump dan mengaktifkan sinyal cu_branch serta menentukan cu_branchtype yang spesifik. Brancher kemudian menerima dua nilai register (in1 dan in2) dan membandingkannya sesuai dengan cu_branchtype. Secara paralel, ALU menghitung alamat target branch atau jump (ALUout) dengan menambahkan immediate value yang telah di-sign-extend ke nilai PC saat ini. Brancher kemudian mengevaluasi kondisi perbandingan: jika kondisi branch terpenuhi, brancher akan memilih ALUout sebagai nilai PCin (alamat PC berikutnya); jika tidak, brancher akan memilih PCnew (PC+4) sebagai PCin. Proses ini mengatur perubahan aliran eksekusi program secara dinamis. Contoh implementasi loop, if-else statement, dan panggilan fungsi.

Untuk simulasi fungsional (Dari Hasil Lampiran), Brancher berhasil memvalidasi semua skenario yang diwajibkan oleh Asisten pada branch kondisional. Ini bisa dilihat dari waveform simulasi. Ketika cu_branch adalah 1 dan cu_branchtype adalah 000 (BEQ - Branch if Equal), dengan in1 sama dengan in2 (misalnya, in1 = 5 dan in2 = 5), maka PCin akan memilih ALUout (00000004), menunjukkan bahwa branch diambil. Sebaliknya, ketika in1 dan in2 tidak sama, PCin akan memilih PCnew (00000008), menunjukkan bahwa branch tidak diambil dan eksekusi berlanjut secara sekuensial.

Selanjutnya, saat cu_branchtype berubah ke 011 (BLT - Branch if Less Than), dan in1 kurang dari in2 (misalnya, in1 = -5 dan in2 = 5), PCin juga memilih ALUout (00000004), mengonfirmasi kalau branch diambil untuk kondisi kurang dari. Jika in1 tidak kurang dari in2, PCin akan kembali ke PCnew. Pengujian ini tervalidasi untuk semua tipe branch (BEQ, BNE, BLT, BGE, BLTU, BGEU) yang tunjukkan Brancher membuat keputusan branching yang tepat berdasarkan perbandingan signed dan unsigned.

Dari Timing Analyzer Summary, terlihat bahwa worst-case propagation delay (tpd) untuk Brancher adalah 18.540 ns, dari in2[29] ke PCin[4]. Nilai delay tersebut menunjukkan waktu maksimum yang dibutuhkan sinyal untuk merambat dari input ke output dalam kondisi terburuk, khususnya pada jalur yang melibatkan perbandingan register yang paling signifikan dan penentuan bit PCin yang relevan. Agar memastikan operasi yang stabil, frekuensi clock maksimum (f_max) yang dapat digunakan harus lebih kecil dari invers dari delay terburuk ini.

f_max < 1 / 18.540 ns ≈ 53.94 MHz

Oleh karena itu, FPGA yang menggunakan Brancher ini harus beroperasi pada frekuensi clock maksimum (f_max) kurang dari sekitar 53.94 MHz. Jika frekuensi clock lebih tinggi dari ini, ada risiko bahwa sinyal keluaran PCin tidak akan stabil pada waktu yang tepat sebelum edge clock berikutnya, yang dapat menyebabkan kesalahan fungsional. Ini berarti frekuensi minimal yang harus didukung oleh FPGA adalah 53.94 MHz untuk memastikan unit ini dapat bekerja dengan baik pada worst-case delay tersebut.

IV. SIMPULAN

- Praktikan telah memahami arsitektur mikroprosesor RISC-V (RV32I) dan datapath eksekusinya dalam versi single-cycle.
- Control Unit (CU) untuk RV32I telah berhasil dibuat dalam HDL yang synthesizable dan dapat disimulasikan. Terbukti berhasil karena mampu menghasilkan sinyal kontrol yang benar untuk berbagai instruksi.
- Immediate selector untuk RV32I telah berhasil diimplementasikan dalam HDL yang synthesizable dan dapat disimulasikan. Terbukti secara akurat mengekstraksi dan melakukan sign-extension pada nilai immediate dari berbagai format instruksi.
- Brancher untuk RV32I telah berhasil dibuat dalam HDL

yang synthesizable dan dapat disimulasikan. Berhasil secara tepat mengarahkan aliran program berdasarkan kondisi branch.

- Komponen-komponen pendukung seperti program counter (PC), 4-adder, dan 2-to-1 mux generik untuk RV32I juga telah berhasil dibuat dalam HDL yang synthesizable dan dapat disimulasikan. Komponen-komponen tersebut digunakan untuk mendukung tahapan instruction fetch dan instruction decode.
- Simulasi fungsional mengkonfirmasi logika desain Verilog HDL, dan analisis timing memberikan informasi worst-case delay paling kritis adalah 18.540 ns (dari Brancher). Sehingga agar operasi stabil dan fungsionalitasnya benar, FPGA yang digunakan untuk mengimplementasikan mikroprosesor harus memiliki kemampuan untuk beroperasi pada frekuensi clock maksimum setidaknya sekitar 53.94 MHz (invers dari 18.540 ns). Jika diperlukan frekuensi operasi yang lebih tinggi, misalnya di atas 90 MHz, optimasi timing lanjut pada *Critical Path* Brancher sangat diperlukan.

Referensi

[1] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 5th ed. Waltham, MA, USA: Morgan Kaufmann, 2014

[2] D. Harris & S. Harris, Digital Design and Computer Architecture: RISC-V Edition.

[3] S. Sutandi, dkk., "M1 rev2 Modul Praktikum EL3011 Arsitektur Sistem Komputer Semester I Tahun Akademik 2025 / 2026,". Ms Teams (Diakses 19 November 2025).

[4] Sutandi, Stefen, dkk. “Modul 2 Praktikum EL3011 Arsitektur Sistem Komputer Edisi 2025 SYNTHESIZABLE RISC-V (RV32I) MICROPROCESSOR BAGIAN II : CONTROL UNIT (CU), IMMEDIATE SELECTOR, BRANCHER, PROGRAM COUNTER (PC), 4-ADDER, DAN 2-TO-1 MUX GENERIK Semester I Tahun Akademik 2025 / 2026. 2025”. Ms Teams (Diakses 20 November 2025).

[5] Venus. (n.d.). [Online]. Available: https://github.com/61c-teach/venus/wiki (Diakses 19 November 2025).

Lampiran

1. Source code untuk studi pustaka Bahasa Verilog HDL

```
module Mux2_4b (
    input  wire [3:0] A_IN,
    input  wire [3:0] B_IN,
    input  wire       S_IN,
    output wire [3:0] C_OUT
);
genvar i;
generate
    for (i=0; i<4; i=i+1) begin : G
    Mux2 u(.A(A_IN[i]), .B(B_IN[i]), .S(S_IN), .D(C_OUT[i]));
    end
    endgenerate
endmodule
```

2. Source code untuk tugas I

```
// Praktikum EL3011 Arsitektur Sistem Komputer
// Modul      : 2
// Percobaan  : 1
// Tanggal    : 19 November 2025
// Nama (NIM) 1 : William Anthony (13223048)
// Nama (NIM) 2 : Agita Trinanda Ilmi (13223003)
// Nama File    : ctrl_unit_rv32i.v
// Deskripsi    : Control Unit (CU) untuk Percobaan 1 RV32I
//                [Register terletak pada cite: 141]

module ctrl_unit_rv32i (
    input wire [6:0] opcode,
    input wire [2:0] funct3,
    input wire [6:0] funct7,

    output reg cu_ALU1src,
    output reg cu_ALU2src,
    output reg [2:0] cu_immtype,
    output reg [1:0] cu_ALUtype,
    output reg cu_adtype,
    output reg [1:0] cu_gatype,
    output reg [1:0] cu_shiftype,
    output reg cu_sltype,
    output reg [1:0] cu_rdtype,
    output reg cu_rdwrite,
    output reg [2:0] cu_loadtype,
    output reg cu_store,
    output reg [1:0] cu_storetype,
    output reg cu_branch,
    output reg [2:0] cu_branchtype,
    output reg cu_PCtype
);

always @ (*)
begin
    cu_ALU1src    = 1'b0; // Dari rs1
    cu_ALU2src    = 1'b0; // Dari rs2
    cu_immtype    = 3'b000; // I-type
    cu_ALUtype    = 2'b00; // ADD/SUB
    cu_adtype     = 1'b0; // Addition
    cu_gatype     = 2'b00; // AND
    cu_shiftype   = 2'b00; // SLL
    cu_sltype     = 1'b0; // SLT (Signed)
    cu_rdtype     = 2'b00; // Dari ALU
    cu_rdwrite    = 1'b0; // No Write
    cu_loadtype   = 3'b010; // Load Word (LW) by default
    cu_store      = 1'b0; // No Store
    cu_storetype  = 2'b00; // Store Byte (SB) by default
    cu_branch     = 1'b0; // No Branch
    cu_branchtype = 3'b000; // BEQ
    cu_PCtype     = 1'b0; // PC + 4

    // --- Main Logic Decoder dengan Source Opcode ---
    case (opcode)

        // R-type [Register di cite: 182]
        7'h33:
        begin
            cu_rdwrite = 1'b1;
            case (funct3)
                // ADD/SUB [Register di cite: 190]
```

```
            3'h0:
            begin
                if (funct7 == 7'h20) // SUB [Register di cite: 192]
                    cu_adtype = 1'b1;
            end
            // SLL [Register di cite: 193]
            3'h1:
            begin
                cu_ALUtype = 2'b10; // Shift op [Register di cite: 194]
            end
            // SLT [Register di cite: 196]
            3'h2:
            begin
                cu_ALUtype = 2'b11; // SLT op [Register di cite: 197]
            end
            // SLTU [Register di cite: 199]
            3'h3:
            begin
                cu_ALUtype = 2'b11; // SLT op [Register di cite: 201]
                cu_sltype = 1'b1;  // Unsigned [Register di cite: 203]
            end
            // XOR [Register di cite: 205]
            3'h4:
            begin
                cu_ALUtype = 2'b01; // Gate op [Register di cite: 207]
                cu_gatype = 2'b10; // XOR (template salah, 2'b11 itu tidak ada) [Register di cite: 209, 30]
            end
            // SRL/SRA [Register di cite: 211]
            3'h5:
            begin
                cu_ALUtype = 2'b10; // Shift op [Register di cite: 213]
                if (funct7 == 7'h00)
                    cu_shiftype = 2'b01; // SRL [Register di cite: 214, 216]
                else
                    cu_shiftype = 2'b11; // SRA [Register di cite: 217, 218]
            end
            // OR [Register di cite: 221]
            3'h6:
            begin
                cu_ALUtype = 2'b01; // Gate op [Register di cite: 224]
                cu_gatype = 2'b01; // OR [Register di cite: 224]
            end
            // AND [Register di cite: 225]
            3'h7:
            begin
                cu_ALUtype = 2'b01; // Gate op [Register di cite: 227]
                // cu_gatype = 2'b00 nilainya pasti default
            end
        endcase
    end

    // I-type (ALU) [Register di cite: 228]
    7'h13:
    begin
        cu_ALU2src = 1'b1; // Dari immediate
        cu_rdwrite = 1'b1; // Tulis ke rd
        // cu_immtype = 3'b000 (I-type) bernilai pasti default

        case (funct3)
            // ADDI (default, tidak perlu di-case)
            // 3'h0:

            // SLLI [Register di cite: 232]
            3'h1:
            begin
                cu_ALUtype = 2'b10; // Shift op
                // cu_shiftype = 2'b00 (SLL) bernilai pasti default
            end
            // SLTI [Register di cite: 234]
            3'h2:
            begin
                cu_ALUtype = 2'b11; // SLT op
                // cu_sltype = 1'b0 (Signed) sudah default
            end
            // SLTIU
            3'h3:
            begin
                cu_ALUtype = 2'b11; // SLT op
                cu_sltype = 1'b1;  // Unsigned
            end
            // XORI
            3'h4:
            begin
                cu_ALUtype = 2'b01; // Gate op
                cu_gatype = 2'b10; // XOR
```

```
            end
            // SRLI/SRAI
            3'h5:
            begin
                cu_ALUtype = 2'b10; // Shift op
                if (funct7 == 7'h00)
                    cu_shiftype = 2'b01; // SRLI
                else
                    cu_shiftype = 2'b11; // SRAI
            end
            // ORI
            3'h6:
            begin
                cu_ALUtype = 2'b01; // Gate op
                cu_gatype = 2'b01; // OR
            end
            // ANDI
            3'h7:
            begin
                cu_ALUtype = 2'b01; // Gate op
                // cu_gatype = 2'b00 (AND) bernilai pasti default
            end
        endcase
    end

    // Load (I-type) [Register di cite: 240]
    7'h03:
    begin
        cu_ALU2src = 1'b1; // Dari immediate
        // cu_immtype = 3'b000 (I-type) bernilai pasti default
        cu_rdtype = 2'b01; // Dari memori
        cu_rdwrite = 1'b1; // Tulis ke rd

        case (funct3)
            3'h0: cu_loadtype = 3'b000; // LB
            3'h1: cu_loadtype = 3'b001; // LH
            3'h2: cu_loadtype = 3'b010; // LW
            3'h3: cu_loadtype = 3'b011; // LBU
            3'h4: cu_loadtype = 3'b100; // LHU
        endcase
    end

    // S-type (Store) [Register di cite: 243]
    7'h23:
    begin
        cu_ALU2src = 1'b1; // Dari immediate
        cu_immtype = 3'b001; // S-type
        cu_store = 1'b1;   // Aktifkan store

        case (funct3)
            3'h0: cu_storetype = 2'b00; // SB
            3'h1: cu_storetype = 2'b01; // SH
            3'h2: cu_storetype = 2'b10; // SW
        endcase
    end

    // B-type (Branch) [Register di cite: 247]
    7'h63:
    begin
        cu_ALU1src = 1'b1; // Dari PC
        cu_ALU2src = 1'b1; // Dari immediate
        cu_immtype = 3'b010; // B-type
        cu_branch = 1'b1;   // Aktifkan branch
        cu_PCtype = 1'b1;   // Ambil PC dari ALU (target branch)

        case (funct3)
            3'h0: cu_branchtype = 3'b000; // BEQ
            3'h1: cu_branchtype = 3'b101; // BNE
            3'h4: cu_branchtype = 3'b011; // BLT
            3'h5: cu_branchtype = 3'b001; // BGE
            3'h6: cu_branchtype = 3'b100; // BLTU
            3'h7: cu_branchtype = 3'b010; // BGEU
        endcase
    end

    // LUI (U-type) [Register di cite: 251]
    7'h37:
    begin
        cu_ALU2src = 1'b1; // Dari immediate
        cu_immtype = 3'b011; // U-type
        cu_rdtype = 2'b11; // Dari immediate (ke rd)
        cu_rdwrite = 1'b1; // Tulis ke rd
    end

    // AUIPC (U-type) [Register di cite: 255]
```

```
        7'h17:
        begin
            cu_ALU1src = 1'b1; // Dari PC
            cu_ALU2src = 1'b1; // Dari immediate
            cu_immtype = 3'b011; // U-type
            cu_rdwrite = 1'b1; // Tulis ke rd
            // cu_rdtype = 2'b00 (Dari ALU) bernilai pasti default
        end

        // JAL (J-type) [Register di cite: 259]
        7'h6F:
        begin
            cu_ALU1src = 1'b1; // Dari PC
            cu_ALU2src = 1'b1; // Dari immediate
            cu_immtype = 3'b100; // J-type
            cu_rdtype = 2'b10; // Dari PC+4
            cu_rdwrite = 1'b1; // Tulis ke rd
            cu_branch = 1'b1;   // Aktifkan jump
            cu_PCtype = 1'b1;   // Ambil PC dari ALU (target jump)
        end

        // JALR (I-type) [Register di cite: 263]
        7'h67:
        begin
            // cu_ALU1src = 1'b0 (Dari rs1) bernilai pasti default
            cu_ALU2src = 1'b1; // Dari immediate
            // cu_immtype = 3'b000 (I-type) bernilai pasti default
            cu_rdtype = 2'b10; // Dari PC+4
            cu_rdwrite = 1'b1; // Tulis ke rd
            cu_branch = 1'b1;   // Aktifkan jump
            cu_PCtype = 1'b1;   // Ambil PC dari ALU (target jump)
        end

    endcase
end

endmodule
```

3. Screenshot hasil tugas 1

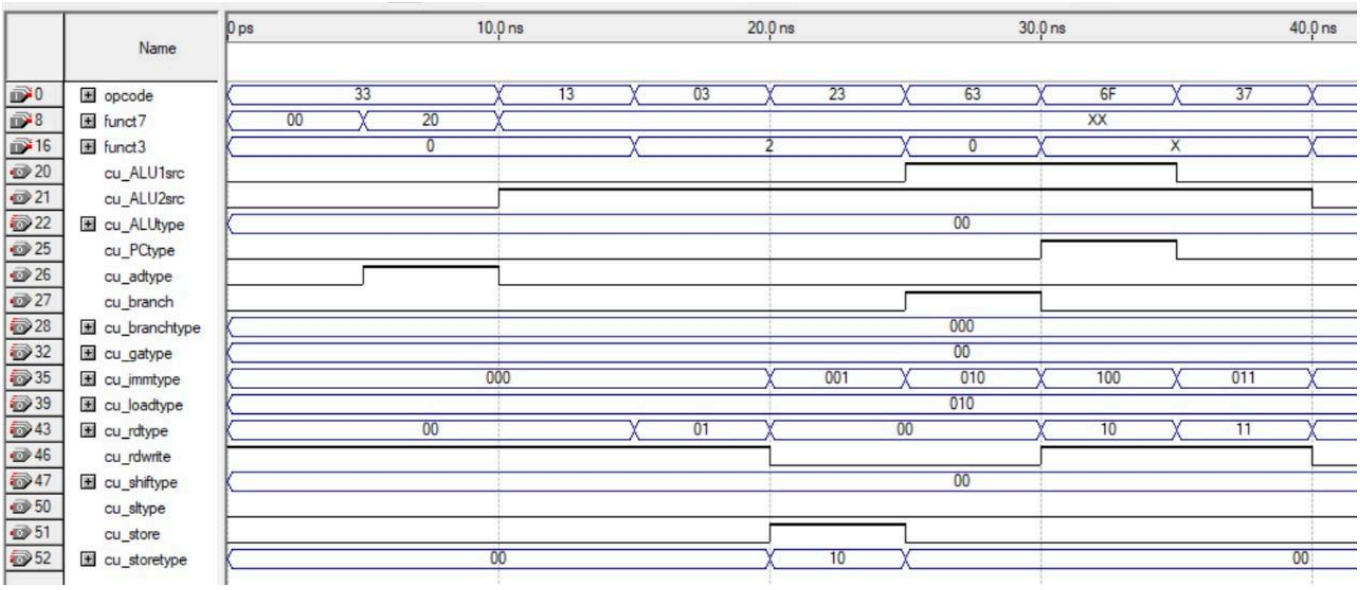
Flow Status	Successful - Wed Nov 19 12:56:49 2025
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	ctrl_unit_rv32i
Top-level Entity Name	ctrl_unit_rv32i
Family	Stratix II
Met timing requirements	Yes
Logic utilization	< 1 %
Combinational ALUTs	41 / 12,480 (< 1 %)
Dedicated logic registers	0 / 12,480 (0 %)
Total registers	0
Total pins	44 / 343 (13 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	0 / 96 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final

Analysis Fitter Tugas 1

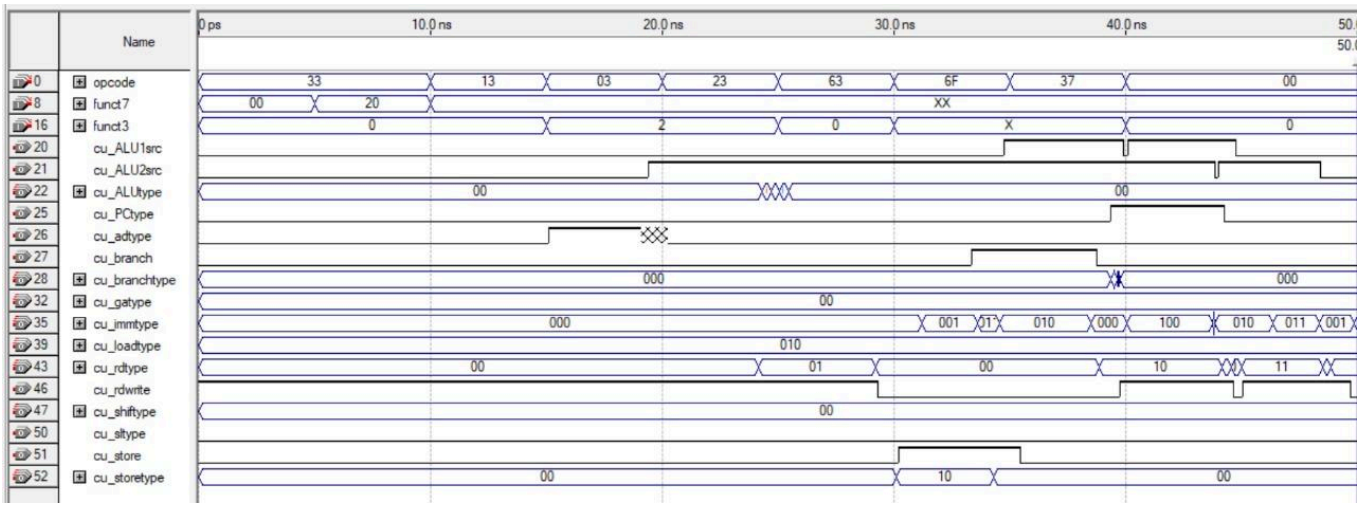
Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	11.505 ns	opcode[3]	cu_loadtype[1]	--	--	0
2	Total number of failed paths								0

Timing Analysis Tugas 1



Fungsional Run Tugas 1



Timing Run Tugas 1

4. Source code untuk tugas II

```
// Praktikum EL3011 Arsitektur Sistem Komputer
// Modul      : 2
// Percobaan  : 2
```

```
// Tanggal    : 19 November 2025
// Nama (NIM) 1 : William Anthony (13223048)
// Nama (NIM) 2 : Agita Trinanda Ilmi (13223003)
// Nama File    : imm_select_rv32i.v
// Deskripsi    : Percobaan 2 yakni Immediate Selector untuk RV32I

module imm_select_rv32i (
    input wire [24:0] trimmed_instr, // 25 MSB dari instruksi
    input wire [2:0]  cu_immtype,    // Tipe immediate (I/S/B/U/J)
    output reg [31:0] imm            // Nilai immediate 32-bit
);

    always @ (*)
    begin
        case (cu_immtype)
            3'b000:  // I - type
                imm <= { {20{trimmed_instr[24]}}, trimmed_instr[24:13] };

            3'b001: // S - type
                imm <= { {20{trimmed_instr[24]}}, trimmed_instr[24:18], trimmed_instr[4:0] };

            3'b010: // B - type
                imm <= { {19{trimmed_instr[24]}}, trimmed_instr[24], trimmed_instr[0],
trimmed_instr[23:18], trimmed_instr[4:1], 1'b0 };

            3'b011: // U - type
                imm <= { trimmed_instr[24:5], 12'h000 };

            3'b100: // J - type
                imm <= { {11{trimmed_instr[24]}}, trimmed_instr[24], trimmed_instr[12:5],
trimmed_instr[13], trimmed_instr[23:14], 1'b0 };

            default:
                imm <= 32'd0;
        endcase
    end

endmodule
```

5. Screenshot hasil tugas 2

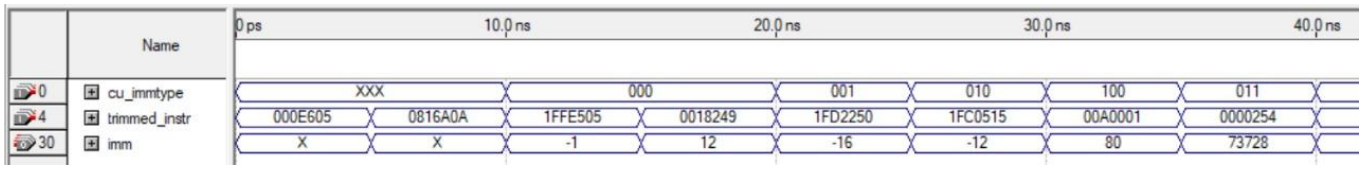
Flow Status	Successful - Wed Nov 19 14:55:45 2025
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	imm_select_rv32i
Top-level Entity Name	imm_select_rv32i
Family	Stratix II
Met timing requirements	Yes
Logic utilization	< 1 %
Combinational ALUTs	32 / 12,480 (< 1 %)
Dedicated logic registers	0 / 12,480 (0 %)
Total registers	0
Total pins	60 / 343 (17 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	0 / 96 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final

Analysis Fitter Tugas 2

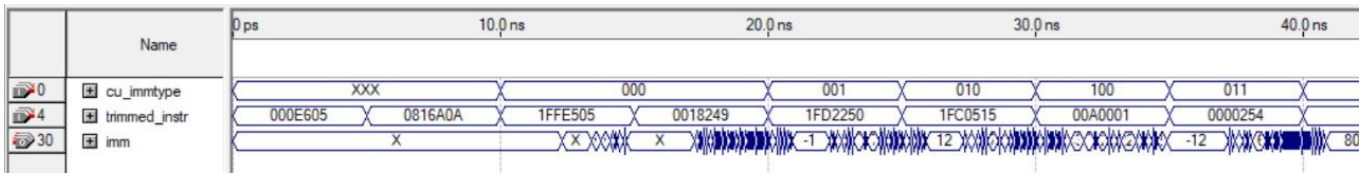
Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	10.891 ns	trimmed_instr[24]	imm[23]	--	--	0
2	Total number of failed paths								0

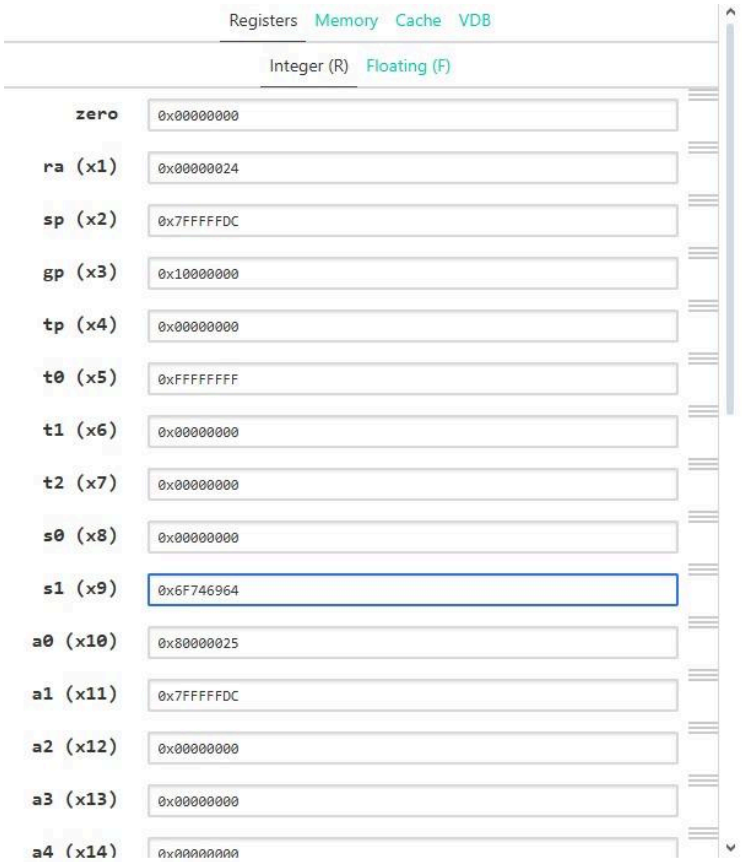
Timing Analysis Tugas 2



Fungsional Run Tugas 2



Timing Run Tugas 2



[5] Venus untuk konfirmasi Alamat

6. Source code untuk tugas III

```
// Praktikum EL3011 Arsitektur Sistem Komputer
// Modul      : 2
// Percobaan  : 3
// Tanggal    : 19 November 2025
// Nama (NIM) 1 : William Anthony (13223048)
// Nama (NIM) 2 : Agita Trinanda Ilmi (13223003)
// Nama File    : brancher_rv32i.v
// Deskripsi    : Brancher untuk RV32I

module brancher_rv32i(
    input wire [31:0] PCnew,   // PC+4 dari 4-adder
    input wire [31:0] ALUout,  // PC+imm dari ALU
    input wire signed [31:0] in1, // Nilai di rs1
    input wire signed [31:0] in2, // Nilai di rs2
    input wire cu_branch,       // Enable branch
    input wire [2:0] cu_branchtype, // Tipe branch
    output reg [31:0] PCin
);

    // N I L A I N Y A          DI B E R I K A N
    // $BEQ=3'b000, $BGE=3'b001, $BGEU=3'b010, $BLT=3'b011,
    // $BLTU=3'b100, $BNE=3'b101 [cite: 379, 382-383]
```

```
    always @ (*)
    begin
        if (cu_branch)
        begin
            case (cu_branchtype)
                // BEQ (Branch if Equal)
                3'b000:
                    PCin <= (in1 == in2) ? ALUout : PCnew;

                // BGE (Branch if Greater or Equal - Signed)
                3'b001:
                    PCin <= (in1 >= in2) ? ALUout : PCnew;

                // BGEU (Branch if Greater or Equal - Unsigned)
                3'b010:
                    PCin <= ($unsigned(in1) >= $unsigned(in2)) ? ALUout : PCnew;

                // BLT (Branch if Less Than - Signed)
                3'b011:
                    PCin <= (in1 < in2) ? ALUout : PCnew;

                // BLTU (Branch if Less Than - Unsigned)
                3'b100:
                    PCin <= ($unsigned(in1) < $unsigned(in2)) ? ALUout : PCnew;

                // BNE (Branch if Not Equal)
                3'b101:
                    PCin <= (in1 != in2) ? ALUout : PCnew;

                // Default: jika tidak ada branch, teruskan PC+4
                default:
                    PCin <= PCnew;
            endcase
        end
        else
        begin
            PCin <= PCnew;
        end
    end

endmodule
```

7. Screenshot hasil tugas 3

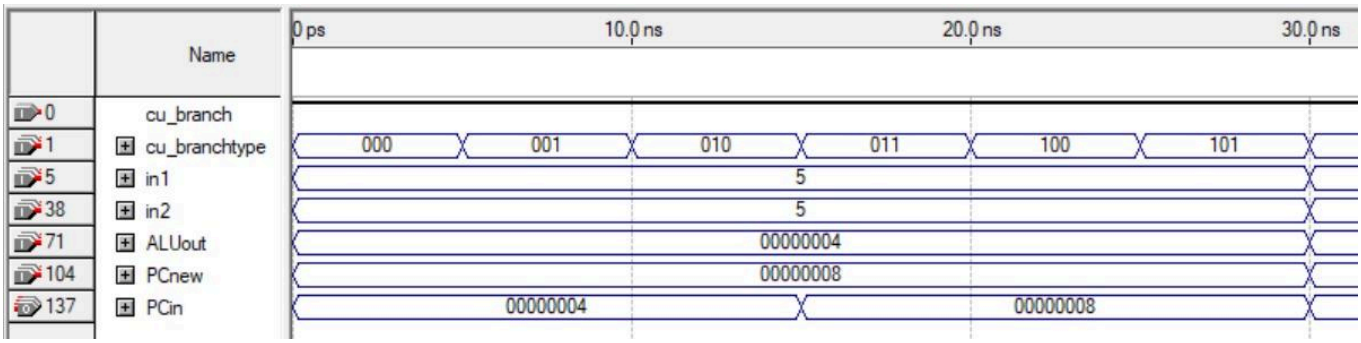
Flow Status	Successful - Wed Nov 19 13:50:41 2025
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	brancher_rv32i
Top-level Entity Name	brancher_rv32i
Family	Stratix II
Met timing requirements	Yes
Logic utilization	< 1 %
Combinational ALUTs	79 / 12,480 (< 1 %)
Dedicated logic registers	0 / 12,480 (0 %)
Total registers	0
Total pins	164 / 343 (48 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	0 / 96 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final

Analysis Fitter Tugas 3

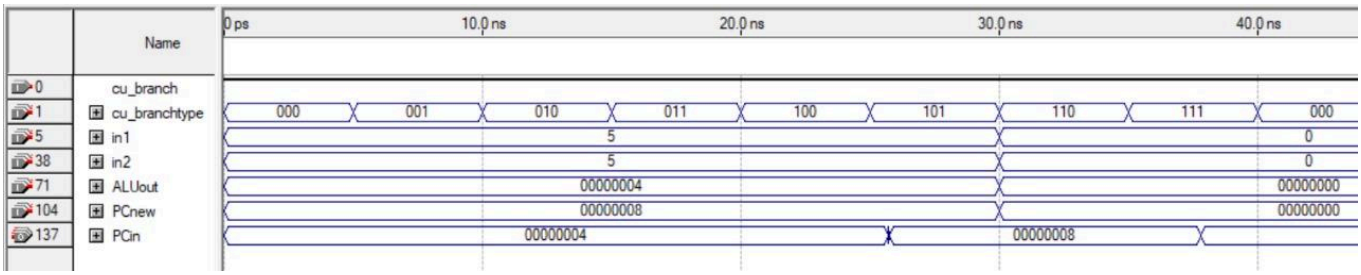
Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	18.540 ns	in2[29]	PCin[4]	--	--	0
2	Total number of failed paths								0

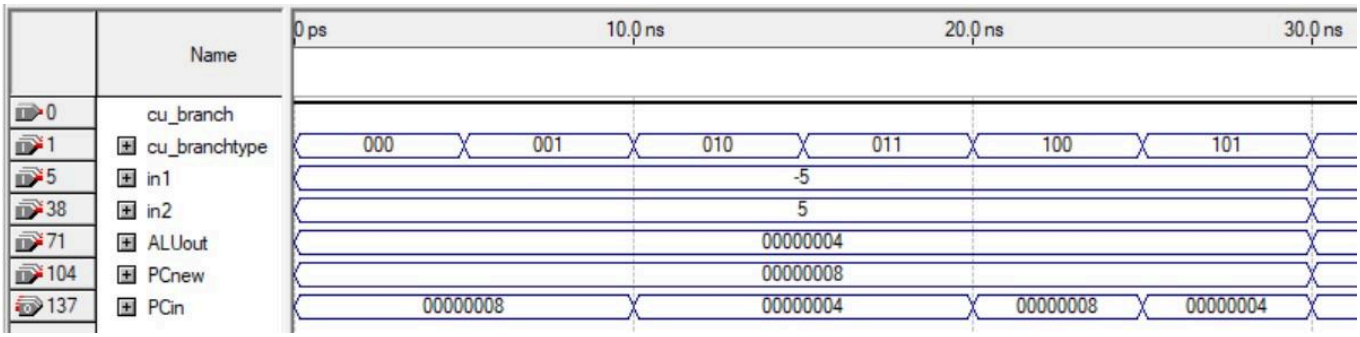
Timing Analysis Tugas 3



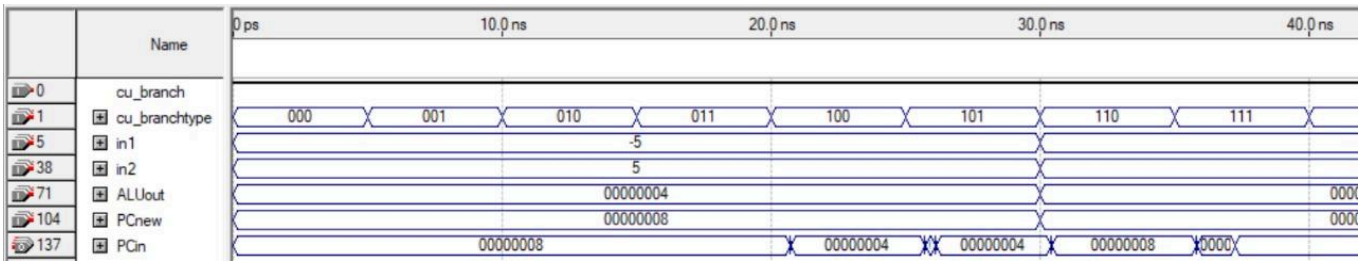
Fungsional Run in1 = in2



Timing Run in1 = in2



Fungsional Run in1 < in2



Timing Run in1 < in2

8. Tabel utnuk Tugas 1

No	Instruksi	Machine code	Sinyal Control Unit															
			cu_ALU1src	cu_ALU2src	cu_immtype	cu_ALUtype	cu_adtype	cu_gatype	cu_shiftype	cu_sltype	cu_rdtype	cu_rdwrite	cu_loadtype	cu_store	cu_storetype	cu_branch	cu_branchtype	cu_Pctype
1	add x5 x6 x7	0x007302b3	0	0	---	00	0	--	--	-	00	1	---	0	--	0	---	0
2	sub x10 x10 x11	0x40b50533	0	0	---	00	1	--	--	-	00	1	---	0	--	0	---	0
3	addi x5 x5 -1	0xfff28293	0	1	000	00	0	--	--	-	00	1	---	0	--	0	---	0
4	lw x9 12(x2)	0x00c12483	0	1	000	00	0	--	--	-	01	1	010	0	--	0	---	0
5	sw x9 -16(x2)	0xfe912823	0	1	001	00	0	--	--	-	--	0	---	1	10	0	---	0
6	beq x5 x0 -12	0xfe028ae3	1	1	011	00	0	--	--	-	--	0	---	0	--	1	000	0
7	jal x1 80	0x050000ef	1	1	100	00	0	--	--	-	10	1	---	0	--	0	---	1
8	lui x20 0x00012	0x00012a37	0	1	011	00	0	--	--	-	11	1	---	0	--	0	---	0

9. Tabel referensi untuk tugas 2

No	Instruksi	Type	Bit [31:25]	Bit [24:20]	Bit [19:15]	Bit [14:12]	Bit [11:7]	Bit [6:0]	Machine code	trimmed_instr
1	add x5 x6 x7	R	***funct7***	***rs2***	***rs1***	***funct3***	***rd***	***opcode***	0x007302b3	0x00E605
			0x00	0x07	0x06	0x00	0x05	0x33		
			000 0000	0 0111	0 0110	000	0 0101	011 0011		
2	sub x10 x10 x11	R	***funct7***	***rs2***	***rs1***	***funct3***	***rd***	***opcode***	0x40B50533	0x816A0A
			0x20	0x0B	0x0A	0x0	0x0A	0x33		
			010 0000	0 1011	0 1010	000	0 1010	011 0011		
3	addi x5 x5 -1	I	***imm***		***rs1***	***funct3***	***rd***	***opcode***	0xFFF28293	0x1FFE505
			0x0FFF		0x05	0x0	0x05	0x13		
			000 1111 1111 111		00101	000	0 0101	001 0011		
4	lw x9 12(x2)	I	***imm***		***rs1***	***funct3***	***rd***	***opcode***	0x00C12483	0x0018249
			0x000C		0x02	0x2	0x09	0x03		
			0000 0000 0000 1100		00010	010	0 1001	000 0011		
5	sw x9 -16(x2)	S	***imm[11:5]***	***rs2***	***rs1***	***funct3***	***imm[4:0]***	***opcode***	0xFE912823	0x1FD2250
			0xFF0 [11:5]	0x09	0x02	0x2	0xFF0 [4:0]	0x23		
			111 1111	01001	00010	010	1 0000	010 0011		
6	beq x5 x0 -12	B	***imm[12][imm10:5]***	***rs2***	***rs1***	***funct3***	***imm[4:1] imm[11]***	***opcode***	0xFE028AE3	0x1FC0515
			0xFF4 [12] 0xFF4[10:5]	0x00	0x05	0x0	0xFF4 [4:1] 0xFF4[11]	0x63		
			1 11 1111	00000	00101	000	1 0101	110 0011		
7	jal x1 80	J	***imm[20] imm[10:5]***	***imm[4:1]imm[11]***	***imm[19:15]***	***imm[14:12]***	***rd***	***opcode***	0x050000EF	0x00A0001
			0x50[20] 0x50[10:5]	0x50[4:1] 0x50[11]	0x50[19:15]	0x50[14:12]	0x01	0x6F		
			0 00 0101	0000 1	00000	000	0 0001	110 1111		
8	lui x20 0x00012	U	***imm[31:25]***	***imm[24:20]***	***imm[19:15]***	***imm[14:12]***	***rd***	***opcode***	0x00012A37	0x0000254
			0x12 [31:25]	0x12 [24:20]	0x12 [19:15]	0x12 [14:12]	0x14	0x37		
			000 0000	0 0000	00010	010	1 0100	011 0111		