FUNDAMENTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

CONSTRUCTION COST PLANNER

GROUP BP-02

FARUQ SAMI RAMADHAN	**2206026675**
KEVIN RAIHAN	**2206059704**
MARIO MATTHEWS GUNAWAN	**2206810452**
YASMIN DEVINA SINURAYA	**2206817244**

PREFACE

Pertama-tama, kami mengucapkan rasa syukur terhadap Tuhan Yang Maha Esa karena dapat menyelesaikan proyek akhir yang berjudul "Construction Cost Planner". Kami ingin mengucapkan terima kasih kepada asisten laboratorium kami, Laode Alif Ma'sum yang turut memberikan bimbingan kepada kami. Kami juga ingin mengucapkan terima kasih kepada seluruh pihak yang turut membantu dalam penulisan laporan proyek akhir ini.

Tujuan dari penulisan laporan ini yaitu untuk memenuhi tugas proyek akhir pada mata kuliah Perancangan Sistem Digital (PSD).

Dalam penyusunan makalah ini, semua pihak baik yang terkait langsung maupun tidak terhadap pembuatan laporan ini telah memberikan kontribusi yang baik dan bermakna. Oleh karena itu, kami berharap laporan ini dapat memberikan manfaat dengan memberikan pengetahuan tambahan kepada pembaca.

Kami menyadari bahwa penulisan laporan dan proyek akhir ini tidak sempurna, oleh karena itu kami sangat menerima kritik dan saran yang diberikan agar dapat membantu kami meningkatkan pengetahuan kami kedepannya. Akhir kata, semoga laporan ini dapat memberikan manfaat yang berarti bagi para pembaca.

Depok, December 23, 2023

Group BP-02

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Industri konstruksi dihadapkan pada tekanan untuk memenuhi tuntutan kecepatan dan akurasi dalam estimasi biaya proyek. Metode konvensional seperti perhitungan manual dan perangkat lunak berdaya proses rendah seringkali tertinggal di belakang laju pembangunan yang kencang. Keterlambatan estimasi dan ketidakakuratannya berakibat pada pembengkakan anggaran, hilangnya kepercayaan pemangku kepentingan, dan menghambat kemajuan proyek. Proyek ini mengeksplorasi solusi potensial melalui Field-Programmable Gate Arrays (FPGA), menawarkan akselerasi berbasis perangkat keras untuk estimasi biaya konstruksi.

1.2 PROJECT DESCRIPTION

Proyek ini bertujuan untuk mengembangkan dan mengimplementasikan sistem berbasis FPGA untuk estimasi biaya konstruksi yang cepat dan akurat. Sistem akan dirancang untuk membantu user menghitung biaya konstruksi untuk 4 tipe bangunan berbeda, yaitu *commercial building, industrial building*, *infrastructure building*, dan *residential building*.

1.3 OBJECTIVES

Tujuan dari pembuatan program yaitu:

1. Membuat program menggunakan VHDL dengan mengimplementasikan microprogramming untuk mengelola perhitungan biaya pembangunan
2. Mengembangkan kinerja program agar dapat melakukan proses perhitungan dengan cepat dan efisien.

1.4 ROLES AND RESPONSIBILITIES

Berikut adalah pembagian tugas dan tanggung jawab tiap anggota kelompok:

Roles	Responsibilities	Person
All Roles	Membuat code calculator, Membuat testbench, Membantu membuat laporan.	Faruq Sami Ramadhan
Role 2	Merancang konsep awal, Mendesain formula tiap tipe konstruksi, Membuat laporan	Kevin Raihan
	Membuat PPT	Mario Matthews Gunawan
Role 4	Membuat kode untuk industrial, Membuat laporan Membantu membuat PPT Membuat README	Yasmin Devina Sinuraya

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Tools yang digunakan dalam pembuatan program yaitu:

- VS Code
- ModelSim
- Quartus
- GitHub

2.2 IMPLEMENTATION

Program ini mengimplementasikan 8 buah modul yang dipelajari selama praktikum Perancangan Sistem Digital(PSD), berikut adalah penjelasan implementasi tiap modul pada program:

A. Modul 2 (Dataflow Style Programming In VHDL)

Statement yang ada pada dataflow style yaitu concurrent signal assignment, conditional signal assignment, dan selected signal assignment. Pada program ini statement yang digunakan hanya 2, yaitu conditional signal assignment dan selected signal assignment.

Untuk conditional signal assignment, implementasinya terletak pada function yang berada di dalam component Calculator. Tujuan penggunaan if-else statement pada function tersebut yaitu untuk menentukan apakah terdapat biaya tambahan yang dikenakan atau tidak. Pada contoh di bawah ini, jika regulations_bit bernilai 1 maka terdapat biaya tambahan untuk safety. Tetapi jika regulations_bit bernilai 0 maka tidak ada biaya tambahan.

```
if regulations_bit = '1' then
    return safety_regulations_cost;
 else
    return 0;
end if;
```

Untuk selected signal assignment, implementasinya terletak pada CostPlanner. Penggunaan switch-case pada component tersebut untuk memilih proses atau instruksi apa yang harus dijalankan ketika sebuah case terpenuhi. Pada contoh di bawah, instruksi yang dijalankan akan dipilih sesuai dengan nilai opcode.

```
case opcode is
      when "01" => -- Residential
         -- Rest of Code
      when "10" => -- Commercial
          --Rest of Code
      when "00" => -- Industrial
          --Rest of Code
      when "11" => -- Infrastructure
       --Rest of Code
      when others =>
          --Rest of Code
      end case;
```

B. Modul 3 (Behavioral Style Programming In VHDL)

Behavioral style programming pada VHDL ditandai dengan penggunaan process statement. Pada program ini, process digunakan pada component Calculator dan CostPlanner. Setiap process ini memiliki sensitivity list yang berbeda-beda. Sebagai contoh, gambar di bawah merupakan contoh penggunaan process pada component CostPlanner. Sensitivity list yang digunakan adalah CPU_CLK, artinya instruksi di dalam block process tersebut akan dijalankan ketika nilai CPU_CLK berubah.

```
process (CPU_CLK)
begin
  if rising_edge(CPU_CLK) then
    -- Rest of code
  end if;
end process;```
```

C. Modul 4 (Testbench)

Pada program ini, testbench dibuat untuk menguji output program berdasarkan input-input yang diberikan.

```
-- Stimulus process
stimulus : process
    constant period: time := 10 ns;
    begin
        -- Initialize inputs
        enable_tb <= '1';
        instruction_tb <= "0010011"; -- Set an example instruction
        Var1_tb <= 10; -- production capacity
        Var2_tb <= 200; -- bulding area
        Var3_tb <= 5; -- equipment quantity
        wait for period;

        enable_tb <= '1';
        instruction_tb <= "0111000"; -- Set an example instruction
        Var1_tb <= 3; -- tingkat bangunan
        Var2_tb <= 200; -- luas tanah
        Var3_tb <= 0; -- dont care
        wait for period;

        enable_tb <= '1';
        instruction_tb <= "1011010"; -- Set an example instruction
        Var1_tb <= 200; -- luas tanah
        Var2_tb <= 10; -- area per customer
        Var3_tb <= 20; -- jumlah customer
        wait for period;

        enable_tb <= '1';
        instruction_tb <= "1100111"; -- Set an example instruction
        Var1_tb <= 500; -- project length
        Var2_tb <= 70; -- unit cost per meter
        Var3_tb <= 100; -- material cost
        wait for period;
        wait; -- End simulation
end process;
```

D. Modul 5 (Structural Style Programming In VHDL)

Penerapan structural style programming pada program ini yaitu penggunaan component-component dan menghubungkannya menggunakan port map. Pada program ini, letak implementasi style ini terdapat pada CostPlanner. Sebagai contoh pada gambar di bawah, isi entity dari component Decoder ditulis terlebih dahulu di dalam architecture CostPlanner.

```
component Decoder is
  port (
      PRG_CNT : in integer; -- Program counter
      instruction : in std_logic_vector(6 downto 0);
      opcode : out std_logic_vector(1 downto 0); -- C
      OP1_ADDR : out std_logic_vector(1 downto 0);
      OP2_ADDR : out std_logic_vector(1 downto 0);
      OP3_ADDR : out std_logic
  );
```

Agar component Decoder dapat digunakan, maka digunakan port map yang berisi parameter-parameter yang dibutuhkan oleh component Decoder. Dengan menggunakan port map maka instruksi yang terdapat di dalam component Decoder dapat digunakan melalui CostPlanner.

```
DEC : Decoder port map (counter, instruction_input, opcode, OP1_ADDR, OP2_ADDR, OP3_ADDR);
```

E. Modul 7 (Procedure, Function, and Impure Function)

Program ini mengimplementasikan function dalam perhitungan biaya pembangunan tiap jenis bangunan. Sebagai contoh pada gambar di bawah, dibuat sebuah function bernama ApplySafetyCost pada CostPlanner. Function ini membutuhkan parameter input bertipe STD_LOGIC dan parameter output bertipe integer. Function ini berfungsi untuk menentukan apakah terdapat biaya tambahan untuk safety atau tidak. Dengan menggunakan if-else statement, jika parameter input(regulations_bit) bernilai 1, maka function akan me-return biaya safety (safety_regulastions_cost) dan hasil return ini akan digunakan dalam perhitungan total biaya pembangunan. Tetapi jika regulations_bit bernilai 0 maka function akan me-return 0, yang artinya tidak ada biaya tambahan yang ditambahkan dalam perhitungan total biaya pembangunan.

```
function ApplySafetyCost(regulations_bit: std_logic) return integer is
begin
  if regulations_bit = '1' then
      return safety_regulations_cost;
   else
      return 0;
  end if;
end function ApplySafetyCost;
```

F. Modul 8 (Finite State Machine)

Program ini menerapkan finite state machine dengan tipe Moore, karena next state hanya ditentukan berdasarkan current state program. Program ini menggunakan 5 buah state berbeda, yaitu IDLE, FETCH, DECODE, EXECUTE, dan COMPLETE. Berikut penjelasan dari setiap state:

a. State IDLE

State IDLE adalah state dimana program akan menunggu nilai input dari enable. Pada state ini, jika enable bernilai 1, maka program akan berpindah ke state FETCH dan nilai counter akan direset menjadi 0. Tetapi ketika nilai enable tidak bernilai 1, maka program akan tetap berada di state IDLE. Dalam kata lain, program tidak akan bekerja hingga enable bernilai 1.

b. State FETCH

State FETCH adalah state dimana program akan menerima instruction input. State ini akan melakukan increment pada counter, dimana jika counter bernilai 1, maka state akan berubah menjadi DECODE.

c. State DECODE

State DECODE adalah state dimana program akan mengisi variabel parameter yang dibutuhkan oleh component Decode, yaitu variabel counter yang sudah di-increment, dan variabel instruction_input diisi dengan nilai instruction. Setelah variabel tersebut diisi, maka proses decoding instruction_input akan dilakukan oleh component Decode. Dan setelah proses decoding selesai, jika counter bernilai 2 maka state berikutnya adalah EXECUTE.

d. State EXECUTE

State EXECUTE adalah state dimana program akan mengisi variabel-variabel yang dibutuhkan oleh component Calculator. Variabel yang dibutuhkan yaitu counter, opcode_input, operand1_input, operand2_input, dan operand3_input. Setelah variabel diisi, maka proses perhitungan biaya akan dilakukan oleh component Calculator. Setelah proses perhitungan biaya selesai, jika counter bernilai 3 maka state berikutnya adalah COMPLETE.

e. State COMPLETE

State COMPLETE adalah final state dimana program akan memberikan report berupa “Instruction Complete”. State ini menjadi state terakhir dalam program dan state berikutnya adalah IDLE.

G. Modul 9 (Microprogramming)

Microprogramming pada program ini terletak pada bagian penentuan jenis bangunan berdasarkan opcode, dimana opcode ini didapat dari hasil decoding instruction input. Berikut adalah arti dari tiap opcode dari hasil decoding:

a. Opcode 2 bit pertama 00 untuk industrial building
b. Opcode 2 bit pertama 01 untuk residential building
c. Opcode 2 bit pertama 10 untuk commercial building
d. Opcode 2 bit pertama 11 untuk infrastructure building

Bit ke 6 ke 5	Bit ke 4 ke 3	Bit ke 2 ke 1	Bit ke 0
Opcode untuk memilih tipe building	Variabel pilihan pertama	Variabel pilihan kedua	Condition bit penggunaan harga tambahan

Sebagai contoh pada gambar di bawah, perhitungan biaya pembangunan menggunakan rumus yang berbeda tergantung dengan opcode yang ada

```
case opcode is
      when "01" => -- Residential
          biaya <= (cost1 * unit_cost_per_floor) +
                   (cost2 * land_cost_per_sqm) +
                   (cost1 * cost2 * cost_per_sqm_materials(index_fasilitas)) +
                   amenities_cost(index_material);
      when "10" => -- Commercial
          biaya <= (cost1 * land_cost_per_sqm) +
                   (cost2 * cost3 * unit_cost_per_sqm) +
                   ApplyComplianceCost(operand3);
      when "00" => -- Industrial
          biaya <= (cost1 * unit_cost_per_unit(index_unit)) +
                   (cost2 * unit_cost_per_sqm_industrial) + (cost3 * equipment_cost(index_equipment))
                    + ApplySafetyCost(operand3);
      when "11" => -- Infrastructure
          biaya <= (cost1 * (cost2 + cost3)) + CalculateCost(operand3);
      when others =>
          biaya <= 0; -- Invalid opcode
```

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Testing dilakukan menggunakan Test Bench yang telah kami buat.

```
-- Stimulus process
stimulus : process
    constant period: time := 10 ns;
    begin
        -- Initialize inputs
        enable_tb <= '1';
        instruction_tb <= "0010011"; -- Set an example instruction
        Var1_tb <= 10; -- production capacity
        Var2_tb <= 200; -- bulding area
        Var3_tb <= 5; -- equipment quantity
        wait for period;

        enable_tb <= '1';
        instruction_tb <= "0111000"; -- Set an example instruction
        Var1_tb <= 3; -- tingkat bangunan
        Var2_tb <= 200; -- luas tanah
        Var3_tb <= 0; -- dont care
        wait for period;

        enable_tb <= '1';
        instruction_tb <= "1011010"; -- Set an example instruction
        Var1_tb <= 200; -- luas tanah
        Var2_tb <= 10; -- area per customer
        Var3_tb <= 20; -- jumlah customer
        wait for period;

        enable_tb <= '1';
        instruction_tb <= "1100111"; -- Set an example instruction
        Var1_tb <= 500; -- project length
        Var2_tb <= 70; -- unit cost per meter
        Var3_tb <= 100; -- material cost
        wait for period;
        wait; -- End simulation
end process;
```

Fig 1. Snippet Code testbench

Testing dilakukan dengan 4 opcode yang berbeda yaitu 0010011, 0111000, 1011010, 1100111. Masing-masing bertujuan untuk mensimulasikan tiap tipe konstruksi (Industrial, Residential, Commercial, Infrastructure) dengan variabel dan condition bit yang disesuaikan.

3.2 RESULT

Berikut hasil dari testbench yang dijalankan

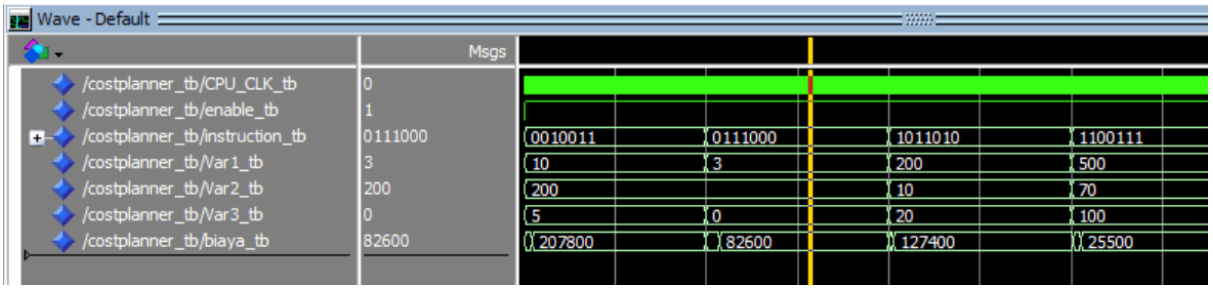


Fig 2. Hasil Testbench

Program membagi instruction input menjadi beberapa bagian, tiap bagian memiliki fungsi yang berbeda. bit 0 dan 1 menentukan tipe konstruksi (industrial, residential, commercial, infrastructure). bit 2 dan 3 menentukan value variable pertama dari pilihan yang sudah ada, bit 4 dan 5 menentukan value variabel kedua dari pilihan yang sudah ada dengan kategori yang berbeda dari variabel pertama. Pada tiap opcode, decoder membagi instruksi lalu melakukan perhitungan berdasarkan tipe konstruksi dan menambahkan variabel berdasarkan bit 2,3 dan 4,5. Opcode pertama 0010011, memiliki spesifikasi tipe konstruksi Industrial, unit cost seharga 80$ per unit, biaya peralatan 1000$ per alat, dan biaya regulasi keamanan dan kesehatan pabrik (42000$). Opcode kedua 0111000, memiliki spesifikasi konstruksi Residential, harga material lantai granit (31$) per meter kuadrat, biaya tambahan halaman (1000$). Opcode ketiga 1011010, memiliki spesifikasi tipe konstruksi Commercial, biaya material lantai granit (31$) per meter kuadrat, dan biaya tambahan patung commercial (1200$). Opcode keempat 1100111, memiliki spesifikasi tipe konstruksi Infrastructure, dengan variabel multiplier kompleksitas (x0), multiplier biaya contingency (x1.5), dan multiplier environmental impact (x2).

Input operasi pertama menghasilkan 207800$, operasi kedua menghasilkan 82500$, operasi ketiga menghasilkan 127400$, dan operasi keempat menghasilkan 25500$.

3.3 ANALYSIS

Hasil output dari desain Cost Planner menggunakan Test Bench lalu kita verifikasi.

$$(10 \cdot 80) + (200 \cdot 800) + (5 \cdot 1000) + 42000 = 207800$$

$$(3 \cdot 1000) + (200 \cdot 300) + (3 \cdot 200 \cdot 31) + 1000 = 82600$$

$$200 \cdot (300 + 31) + (10 \cdot 20 \cdot 300) + 1200 + 0 = 127400$$

$$500 \cdot \left(\left(70 \cdot \frac{10}{10}\right) + 100\right) \cdot 2 \cdot \frac{15}{100} = 25500$$

Fig 1. Verifikasi menggunakan kalkulator Desmos

Dapat kita pastikan bahwa value output program sudah sesuai dari ekspektasi. Tidak terdapat kesalahan logika maupun syntax dalam hasil akhirnya.

Namun terdapat kesalahan dalam output program ketika tengah penjalanan program. Sebelum hasil akhir keluar terdapat value yang tidak diketahui asalnya. Hal ini tidak mempengaruhi tujuan program karena hasil akhir masih valid namun lebih baik jika kita dapat menanggulangi kesalahan ini pada lain waktu.

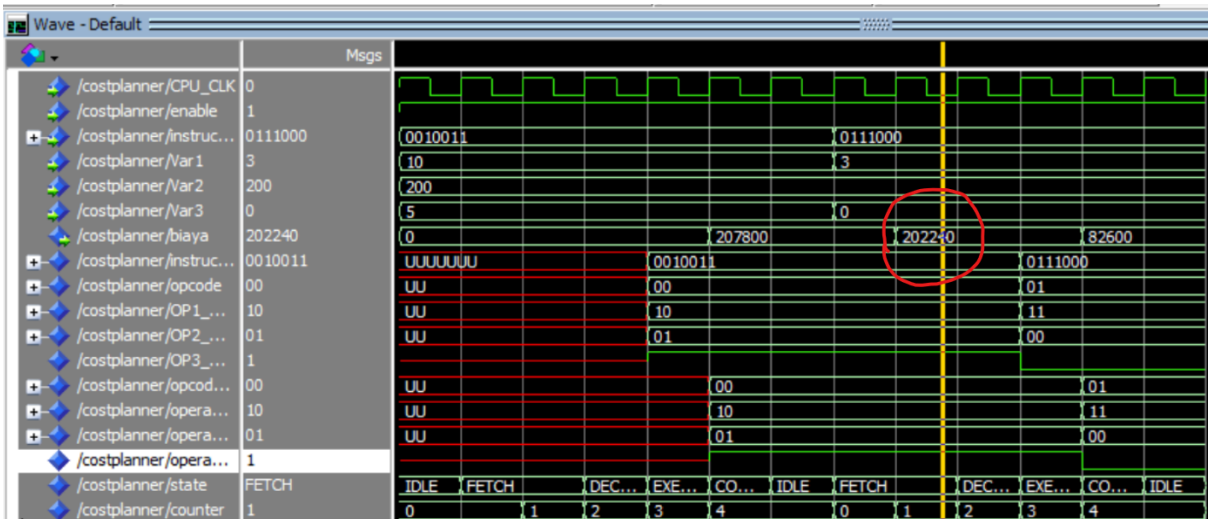


Fig 2. Value sebelum output akhir yang tidak diinginkan

CHAPTER 4

CONCLUSION

penggunaan sistem "Construction Cost Planner" dapat meningkatkan efisiensi secara keseluruhan dalam sebuah proyek konstruksi. Dari penghematan waktu hingga pengurangan kesalahan perhitungan, sistem ini dapat memainkan peran penting dalam meningkatkan kinerja proyek. VHDL adalah bahasa deskripsi perangkat keras yang digunakan untuk mendesain dan mensimulasikan sistem digital. Dalam proyek kami ini, VHDL dapat digunakan untuk mendeskripsikan fungsi-fungsi sistem tersebut dalam level yang lebih rendah, seperti bagaimana sistem akan melakukan perhitungan biaya. Program VHDL yang telah kami buat mengimplementasikan modul-modul yang telah kami pelajari selama praktikum PSD dilakukan.

Proyek ini mengeksplorasi solusi potensial melalui Field-Programmable Gate Arrays (FPGA), FPGA merupakan platform perangkat keras yang dapat diprogram ulang untuk menjalankan fungsi-fungsi yang telah dideskripsikan dalam VHDL. Dalam hal ini, setelah sistem "Construction Cost Planner" didesain dan disimulasikan menggunakan VHDL, FPGA dapat digunakan sebagai target implementasi. Sistem tersebut dapat diprogram ke dalam FPGA untuk dijalankan secara langsung di perangkat keras, memberikan kecepatan dan efisiensi yang diperlukan.

Construction Cost Planner yang kelompok kami buat memiliki input instruction dengan panjang 7 bit dengan bit 6 ke 5 adalah opcode untuk memilih tipe building yang diinginkan, lalu bit 4 ke 3 digunakan untuk memilih variabel yang telah diatur, bit 2 ke 1 juga digunakan untuk memilih variabel yang berbeda dengan variabel sebelumnya, bit terakhir adalah bit condition untuk memilih apakah akan biaya tambahan atau tidak.

Melalui uji coba dan analisis, program yang kami buat telah sesuai dengan apa yang kami harapkan. perhitungan rumus yang kami buat dan variabel-variabel yang kami sediakan berfungsi dengan baik. Namun terdapat kesalahan dalam output program ketika tengah penjalanan program. Sebelum hasil akhir keluar terdapat value yang tidak diketahui asalnya, tetapi value tersebut tidak mempengaruhi hasil dari program.

REFERENCES

[1] ProEst. (2022, December 19). *The 5 types of construction | PRoEST*. https://proest.com/construction/process/types-of-construction/ (accessed 24 December 2023)

[2] Emas2.ui.ac.id

[3] VHDL - Assertion Statement. (n.d.). https://peterfab.com/ref/vhdl/vhdl_renerta/mobile/source/vhd00007.htm (accessed 24 December 2023)

[4] J. J. Jensen, "How to use a function in VHDL," VHDLwhiz, Jul. 08, 2023. Available: https://vhdlwhiz.com/function/ (accessed 24 December 2023)

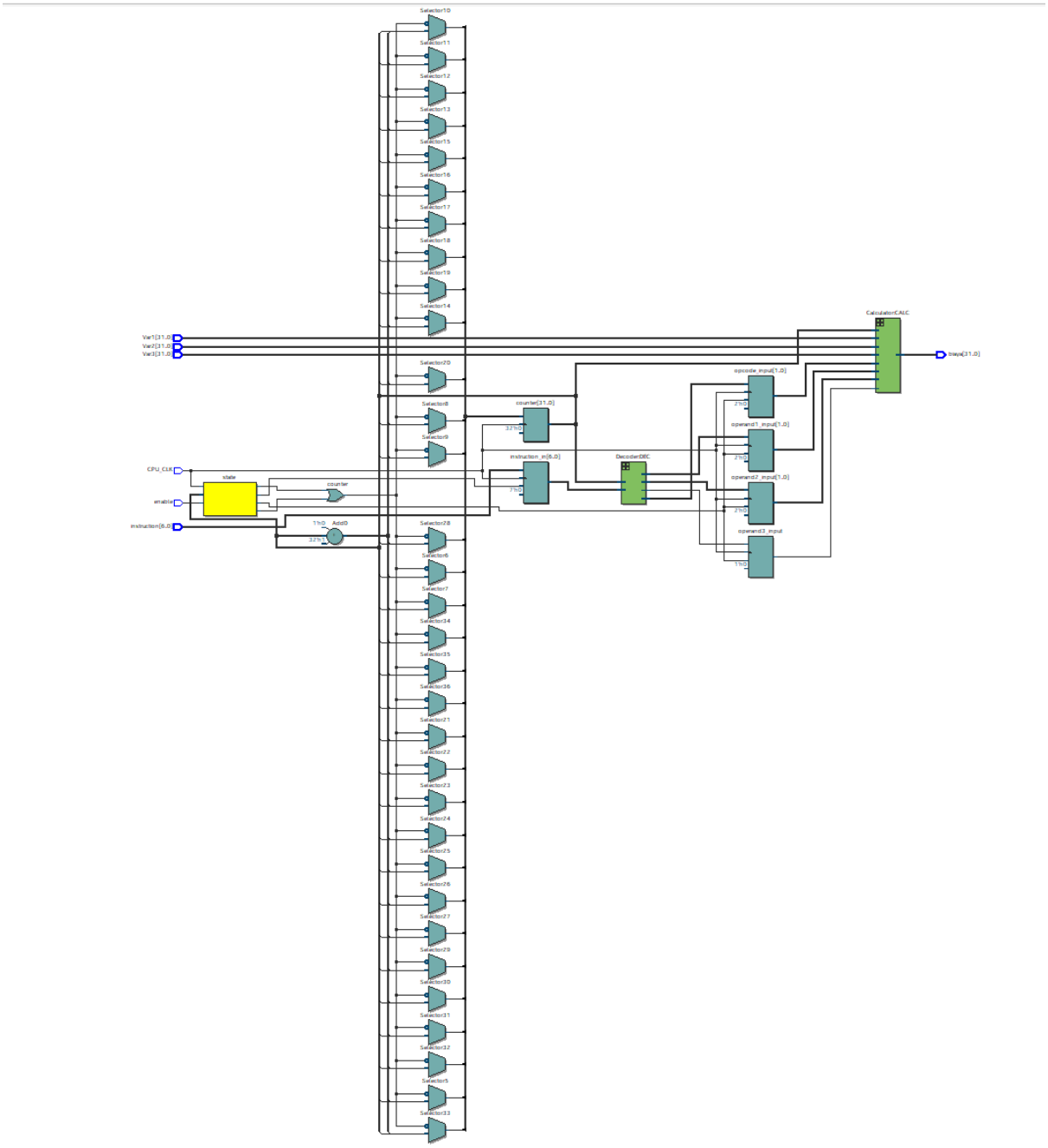
[5] "Finite State Machine - javatpoint," www.javatpoint.com, https://www.javatpoint.com/finite-state-machine (accessed 24 December 2023)

[6] J. J. Jensen, "How to create a signal vector in VHDL: Std_logic_vector," VHDLwhiz, https://vhdlwhiz.com/std_logic_vector/ (accessed 24 December 2023)

APPENDICES

Appendix A: Project Schematic

Put your final project latest schematic here (Quartus)



Appendix B: Documentation

Put the documentation (photos) during the making of the project