TP ARCHITECTURE DES ORDINATEURS

Analyse Détaillée des Questions 2, 5 et 8

Rapport de Groupe

Groupe 2
Membre 1 : Nzau Lumendo Ulrich
Membre 2 : Mukabi Ngombo Nelson
Membre 3 : Lungi Selemani Josué
Membre 4 : Mugaruka Tshioka Jeff

16 juin 2025

1ICE

Table des matières

1 Introduction

Ce rapport présente l'analyse, la réalisation et la vérification des modules Verilog répondant aux questions 2, 5 et 8 du TP d'architecture des ordinateurs. Chaque question est traitée par l'ensemble des membres du groupe 2, avec une répartition équitable des tâches de conception, implémentation et test.

2 Question 2 : Multiplieur Binaire

2.1 i) Étapes de la multiplication binaire

1. Génération des produits partiels : ET logique entre chaque bit du multiplicande et du multiplicateur
2. Alignement des produits partiels selon leur poids binaire
3. Addition séquentielle ou parallèle des produits partiels
4. Gestion des retenues propagées sur toute la largeur du résultat

2.2 ii) Implémentation avec portes logiques

L'implémentation utilise trois niveaux d'additionneurs :
— Demi-additionneurs (HA) pour les bits de poids faible
— Additionneurs complets (FA) pour les additions intermédiaires
— Chaînage des retenues entre les étages

```
module multiplier_4x4(
    input [3:0] a, b,
    output [7:0] p
);
    // G n ration des produits partiels
    wire [3:0] pp0 = a & {4{b[0]}};
    wire [3:0] pp1 = a & {4{b[1]}};
    wire [3:0] pp2 = a & {4{b[2]}};
    wire [3:0] pp3 = a & {4{b[3]}};

    // Addition des produits partiels
    // ... (3 niveaux d'additionneurs)
endmodule
```

Listing 1 – Structure du multiplieur 4x4

2.3 iii) Défis pour grandes tailles

— **Latence** : Temps de propagation O(n) pour les multiplieurs en array
— **Complexité** : Nombre de portes $O(n^2)$ pour n bits
— **Consommation** : Puissance dissipée importante
— **Solutions** : Architectures avancées (Wallace Tree), pipeline

2.4 iv) Design Verilog et testbench

Testbench exhaustif vérifiant les 256 combinaisons possibles :

```
initial begin
    for (int i = 0; i < 16; i++) begin
        for (int j = 0; j < 16; j++) begin
            a = i; b = j; #10;
            if (p !== i*j) $error("Erreur %0d*%0d", i, j);
        end
    end
    $display("Tous tests r ussis");
end
```

Listing 2 – Testbench du multiplieur

3 Question 5 : Multiplexeur 4 :1

3.1 i) Fonction et utilisation

Le multiplexeur (MUX) permet de sélectionner une source parmi plusieurs entrées selon un code de sélection. Applications principales :

— Routage de données dans les processeurs entre mémoires et bus de données
— Systèmes de communication multiplexés

3.2 ii) Conception d'un multiplexeur 4 :1

Équation logique : $Y = (\overline{S1} \cdot \overline{S0} \cdot D0) + (\overline{S1} \cdot S0 \cdot D1) + (S1 \cdot \overline{S0} \cdot D2) + (S1 \cdot S0 \cdot D3)$

```
module mux4to1(
    input [3:0] data_in,
    input [1:0] sel,
    output reg out
);
    always @(*) begin
        case(sel)
            2'b00: out = data_in[0];
            2'b01: out = data_in[1];
            2'b10: out = data_in[2];
            2'b11: out = data_in[3];
            default: out = 1'bx;
        endcase
    end
endmodule
```

Listing 3 – Implémentation Verilog

3.3 iii) Applications pratiques

— Sélection d'opérandes dans les ALU
— Gestion de bus de données multiples de périphériques d'E/S
— Conversion parallèle-série

3.4 iv) Design Verilog et testbench

Validation exhaustive avec 4 entrées fixes et 4 sélections :

```
initial begin
    data_in = 4'b1010; // Configuration fixe
    for (int i = 0; i < 4; i++) begin
        sel = i; #10;
        $display("Sel=%b, Out=%b", sel, out);
    end
end
```

4 Question 8 : Convertisseur Binaire vers Décimal

4.1 i) Principe de fonctionnement

Algorithme Double Dabble :

1. Initialisation du registre BCD avec le binaire en LSB
2. Pour chaque bit (MSB en premier) :
   — Décalage à gauche du registre BCD
   — Ajout de 3 aux digits 5
3. Extraction des digits décimaux après 8 itérations

4.2 ii) Implémentation avec circuits logiques

— Registre à décalage 20 bits
— Comparateurs 4 bits pour la condition 5
— Additionneurs +3 pour l'ajustement
— Machine à états pour le contrôle

4.3 iii) Applications pratiques

— Affichage numérique (calculatrices, instruments) utilisateur avec écrans LCD
— Systèmes embarqués basse consommation
— Traitement de données décimales (finances)

4.4 iv) Design Verilog et testbench

Implémentation de l'algorithme Double Dabble :

```
module bin2bcd8(
    input [7:0] bin,
    output reg [3:0] centaines, dizaines, unites
);
    reg [19:0] bcd;
    integer i;

    always @(*) begin
        bcd = 20'd0;
        bcd[7:0] = bin;
        for (i = 0; i < 8; i = i+1) begin
            if (bcd[11:8] >= 5) bcd[11:8] += 3;
            if (bcd[15:12] >= 5) bcd[15:12] += 3;
            if (bcd[19:16] >= 5) bcd[19:16] += 3;
```

```
            bcd = bcd << 1;
        end
        centaines = bcd[19:16];
        dizaines = bcd[15:12];
        unites = bcd[11:8];
    end
endmodule
```

Listing 4 – Convertisseur bin2bcd

Test des valeurs critiques :

```
task test(input [7:0] value);
    bin = value; #10;
    $display("Bin=%0d => %0d%d%d", value, centaines, dizaines, unites);
endtask

initial begin
    test(0);
    test(10);
    test(99);
    test(128);
    test(255);
end
```

5 Répartition des tâches

Membre	Question 2	Question 5	Question 8
Nzau Lumendo Ulrich	Conception	Testbench	Documentation
Mukabi Ngombo Nelson	Testbench	Documentation	Conception
Lungi Selemani Josué	Documentation	Conception	Testbench
Mugaruka Tshioka Jeff	Intégration	Validation	Validation

TABLE 1 – Répartition des tâches au sein du groupe

6 Conclusion

Ce TP a permis au groupe de maîtriser trois composants fondamentaux en architecture des ordinateurs :

— Le multiplieur structurel avec gestion optimisée des retenues
— Le multiplexeur pour le routage de données
— Le convertisseur numérique par algorithme Double Dabble

Les testbenches exhaustifs ont validé le fonctionnement correct des trois modules dans tous les cas limites.

Annexes

Les codes sources complets sont disponibles sur dépôt Git :
`https://github.com/Ulrich930/archi-tp`