VHDL 入門編トライアル・コース
演習マニュアル

この演習を完了させるのに必要なアイテム

- ModelSim®-Altera® Edition 10.4b (Quartus® Prime 15.1)

または

- ModelSim-Altera Starter Edition 10.4b (Quartus Prime 15.1)

※ *上記ツールバージョン以外でも演習を行うことはできますが、メニューの位置や操作方法など異なる場合があります。*

2016 年 4 月 Rev.1

VHDL 入門編トライアル・コース
演習マニュアル

目次

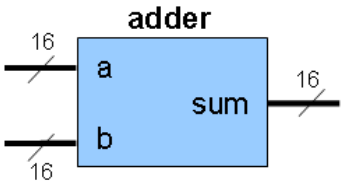
はじめに

- 本マニュアルは、演習用データの保存先を **C:¥lab¥vhdl_lab** として説明しています。もしあなたの保存した先が **C:¥lab¥vhdl_lab** と異なる場合は、マニュアル内のパスを自分の環境に合わせ適応してください。
- この演習では、実際に記述した VHDL の回路が期待した動作をするかを確認するために、ファンクション・シミュレーションを実行します。その際に使用するソフトウェアは、ModelSim-Altera です。
- 演習で使用するためテストベンチ・ファイルは、あらかじめ作成済みのものを使用します。
- この演習におけるツールの操作および設定は、本演習に限定した内容です。
- 本コースは言語のトライアル・コースです。ツール操作を習得するコースではないため、操作の解説は省略しています。ご了承ください。
- 演習データは、別途案内する Web ページからダウンロードすることができます。ダウンロードした ZIP ファイルを解凍すると、EXE ファイルが生成されます。その EXE ファイルを実行して、演習データの保存先を指定してください。（デフォルトの **C:¥lab¥vhdl_lab** でも良いですが、お好みに応じて保存先を変更することもできます。

演習 1

＜目的＞

- 記述の穴埋めをし、加算器を完成させます。
- ファンクション・シミュレーションを実行し、動作を確認します。

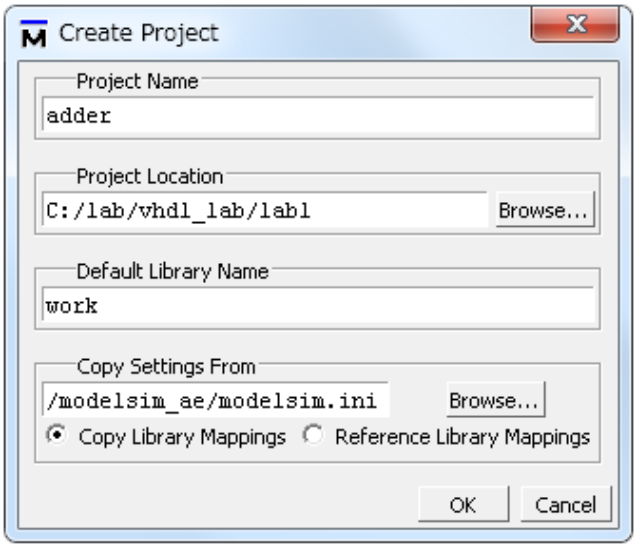


- ◆ エンティティ名: adder
- ◆ 入力ポート: a (16bit), b (16bit)
- ◆ 出力ポート: sum (16bit)
- ◆ データ・タイプ: std_logic_vector
- ◆ 機能: 16ビット加算器
- ◆ 作業ディレクトリ: C:¥lab¥vhdl_lab¥lab1

ステップ 1: シミュレーション用プロジェクトの作成

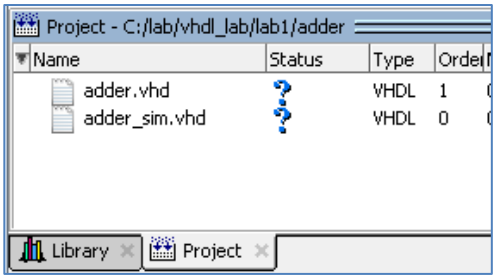
1. ModelSim-Altera を起動します。
2. File メニュー ⇒ New ⇒ Project を選択して、Create Project ダイアログ・ボックスを開きます。
3. Project Name 欄に、**adder** と記述します。Project Location 欄に、**C:/lab/vhdl_lab/lab1**（作業ディレクトリ）を選択します。それ以外はデフォルト状態のままで OK ボタンをクリックします。

   Add items to the Project ウィンドウが表示されますが、今回は使用しません。Close ボタンで閉じます。



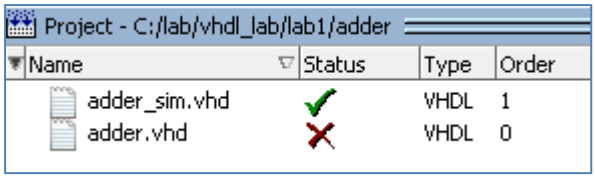
ステップ 2: デザインの作成とプロジェクトへの登録

1. File メニュー ⇒ Open を選択して、作業ディレクトリ内に保存されている **adder.vhd** を選択し開きます。
2. ソースコードを確認します。エンティティ宣言部分が記述されていません。上記の回路仕様を参考にし、加算器のエンティティ名やポート名などの構成を記述して、デザインを完成させてください。
3. 記述終了後、File メニュー ⇒ Save にて、完成させた VHDL ファイルを **adder.vhd** として作業ディレクトリ内に上書き保存します。その後、addrer.vhd ファイルを閉じます。
4. Project メニュー ⇒ Add to Project ⇒ Existing File を選択します。Add file to Project ウィンドウで Brose... ボタンをクリックして、**adder.vhd** と **adder_sim.vhd**（テストベンチ）を選択して OK をクリックします。Project ウィンドウには、下図のように指定した 2 ファイルが登録されます。



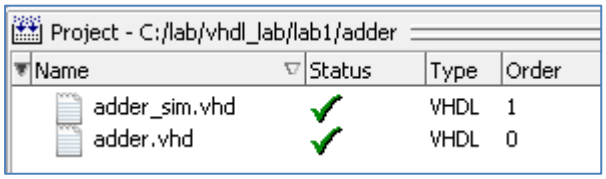
ステップ 3: ソースコードをコンパイル

1. Compile メニュー ⇒ Compile Order を選択し、コンパイルの実行順序を設定します。Compile Order ウィンドウの上段に **adder.vhd**、下段に **adder_sim.vhd** を配置後、OK ボタンをクリックして順番を確定します。
2. Compile メニュー ⇒ Compile All により、ソースコードのコンパイルを実行します。エラーが出る場合には、記述において文法的な間違いがあります。エラーメッセージをヒントに記述を修正してください。修正後は再びコンパイルを実行し、エラーが無くなるまで記述のデバッグを行います。



コンパイルエラーの例。

コンパイルに成功すると、ステータスが ✔ マークに変わり、成功したという内容のメッセージも表示されます。

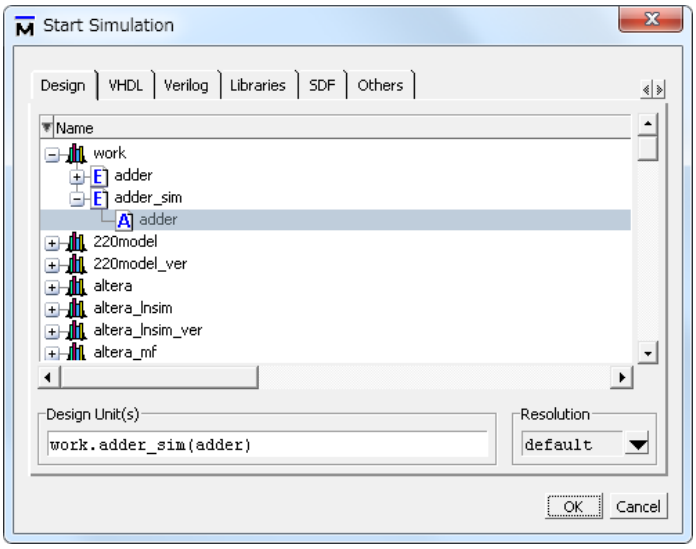


エラーなし。

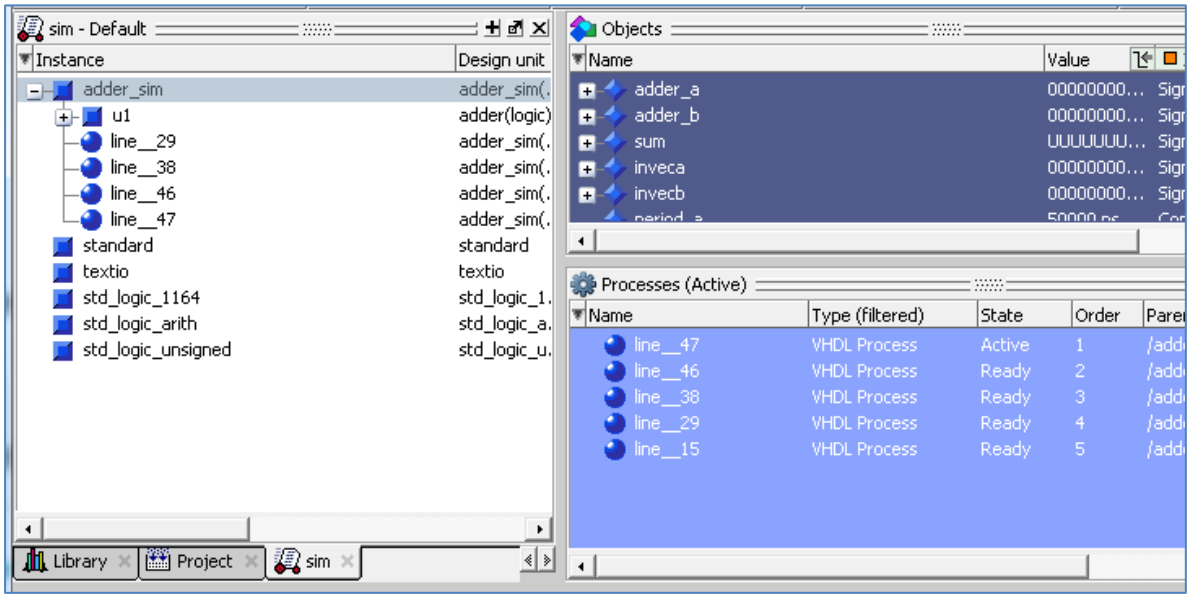
```
# Compile of adder.vhd was successful.
# Compile of adder_sim.vhd was successful.
# 2 compiles, 0 failed with no errors.
```

ステップ 4: デザインのロード

1. Simulator メニュー ⇒ Start Simulation を選択し、Start Simulation ダイアログ・ボックスを起動します。
2. Design タブより work ディレクトリを展開して、**adder_sim** の下位にある **adder** を選択し OK ボタンをクリックします。デザインのロードが開始されます。



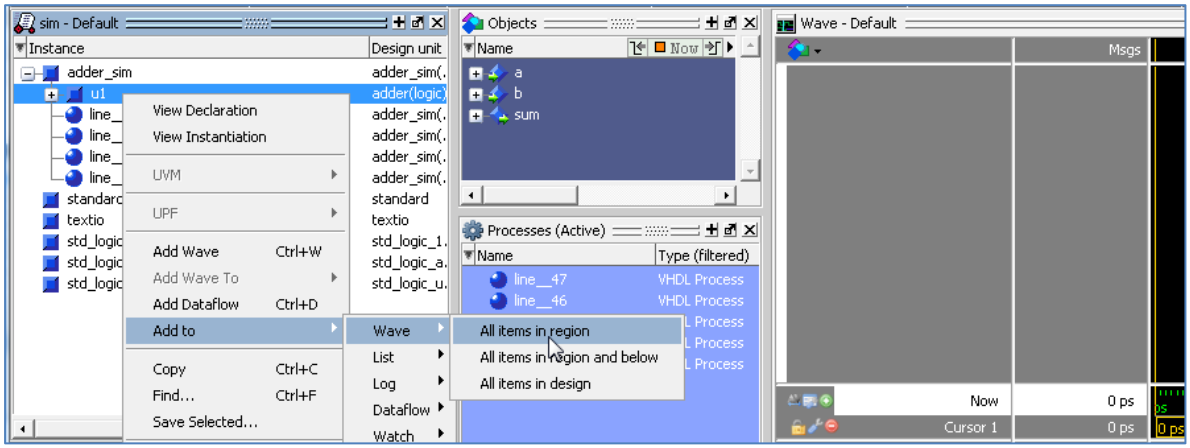
3. ロードが問題なく終了すると、sim タブ（sim ウィンドウ）が追加され、関連ウィンドウが起動します。



ステップ 5: ファンクション・シミュレーションの実行

1. sim ウィンドウ内の **adder_sim** の下位にある **u1** を選択し、右クリック ⇒ Add to ⇒ Wave ⇒ All items in region を選択します。

   VHDL デザインの入出力ピンが登録された状態で Wave ウィンドウが起動します。

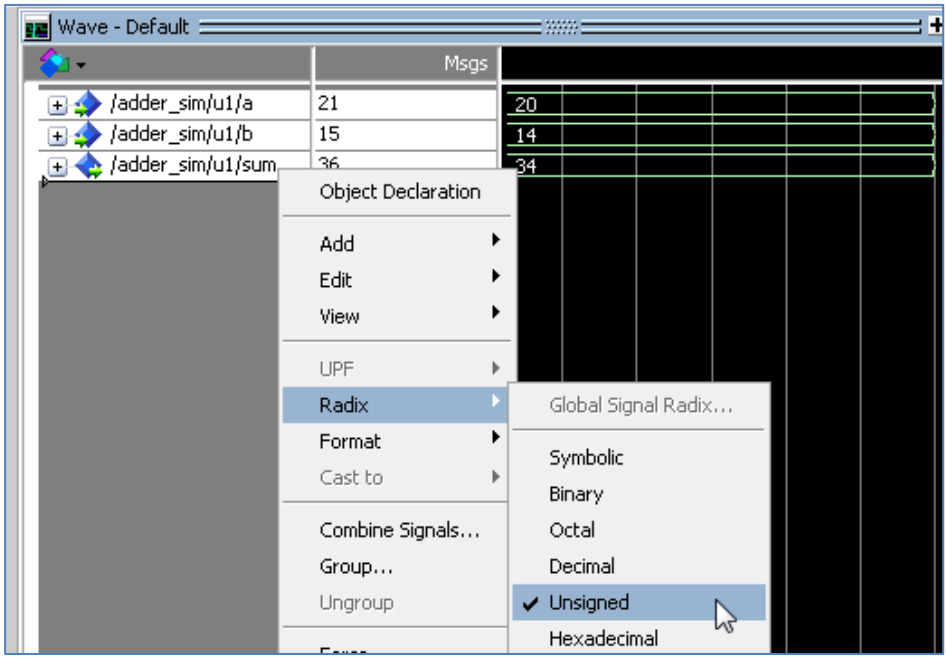


2. Transcript ウィンドウに以下の実行コマンドを入力し、Enter キーでシミュレーション実行を開始します。

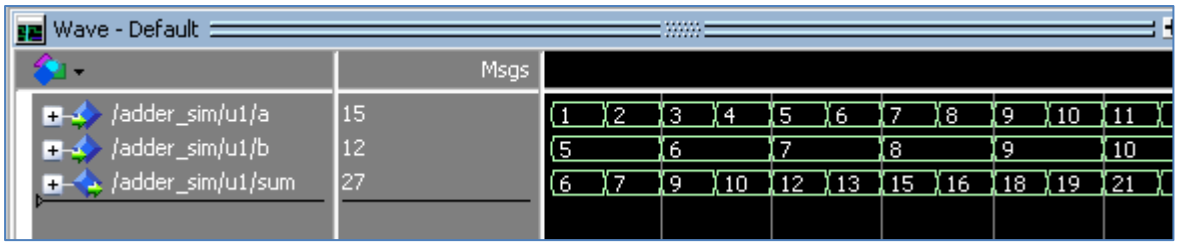
   run␣1us

   ※␣は半角スペース

   Wave ウィンドウにシミュレーション結果が波形表示されます。作成した回路が正しく動作しているか確認しましょう。Wave メニュー ⇒ Zoom ⇒ ･･･、またはツールバーのアイコンにて波形を明確に確認できます。信号の Radix を変更する場合は、信号選択後に右クリック ⇒ Radix ⇒ 目的の表示を選択してください。



以下の例のように、期待する動作（加算）をしていますか？



様々なツールバーを使って波形を見やすくしてください。主なものを紹介します。

 ： ピン名の表示方法の切り替え

 ： 指定した範囲を拡大して表示

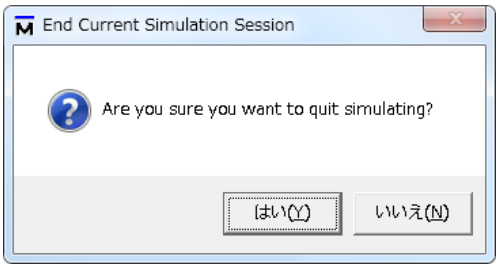
 ： 波形の全体表示

 ： 拡大表示

 ： 縮小表示

ステップ 6: シミュレーションおよびプロジェクトの終了

1. Simulate メニュー⇒ End Simulation よりシミュレーションを終了します。メッセージが表示されますので“はい（Y）”を選択します。



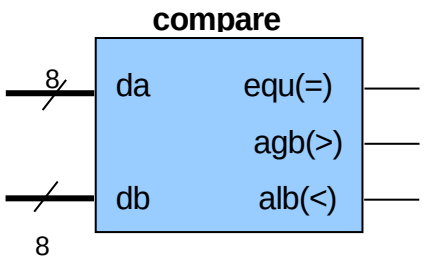
2. メイン・ウィンドウの Project タブをアクティブにし、File メニュー ⇒ Close Project を選択します。メッセージが表示されますので“はい（Y）”を選択し、プロジェクトを終了します。

これで、演習 1 は終了です。

演習 2

＜目的＞

- when-else 文で比較器を作成します。
- ファンクション・シミュレーションを実行し、動作を確認します。



- ◆ エンティティ名: compare
- ◆ 入力ポート: da (8bit), db (8bit)
- ◆ 出力ポート: equ, agb, alb (1bit)
- ◆ データ・タイプ: std_logic, std_logic_vector
- ◆ 機能: 比較器
- ◆ 作業ディレクトリ: C:¥lab¥vhdl_lab¥lab2

＜動作条件＞

入力信号 da と db を比較します。

- ➢ da と db が等しい場合は、equ は High (1) を出力し、それ以外のときは Low (0) を出力
- ➢ da が大きい場合は、agb は High (1) を出力し、それ以外のときは Low (0) を出力
- ➢ db が大きい場合は、alb は High (1) を出力し、それ以外のときは Low (0) を出力

ステップ 1: シミュレーション用プロジェクトの作成

1. ModelSim-Altera が起動していない場合は、ModelSim-Altera を起動します。
2. File メニュー ⇒ New ⇒ Project を選択して、Create Project ダイアログ・ボックスを開きます。
3. Project Name 欄に、**adder** と記述します。Project Location 欄に、**C:/lab/vhdl_lab/lab2**（作業ディレクトリ）を選択します。それ以外はデフォルト状態のままで OK ボタンをクリックします。

   Add items to the Project ウィンドウが表示されますが、今回は使用しません。Close ボタンで閉じます。

ステップ 2: デザインの作成とプロジェクトへの登録

1. File メニュー ⇒ New ⇒ Source ⇒ VHDL よりテキスト・エディタを開きます。
2. ソースコードを記述します。上記の回路仕様を参考にし、比較器を完成させてください。
3. 記述終了後、File メニュー ⇒ Save As より、完成させた VHDL ファイルを **compare.vhd** として作業ディレクトリ内に保存します。その後、**compare.vhd** ファイルを閉じます。
4. Project メニュー ⇒ Add to Project ⇒ Existing File を選択し、**compare.vhd** と **compare_sim.vhd**（テストベンチ）を指定します。Project ウィンドウには、指定した 2 ファイルが登録されます。

ステップ 3: コンパイル

1. Compile メニュー ⇒ Compile Order にて、コンパイルの実行順序を設定します。Compile Order ウィンドウの上段に **compare.vhd**、下段に **compare_sim.vhd** を配置後、OK ボタンをクリックして順番を確定します。
2. Compile メニュー ⇒ Compile All により、コンパイルを実行します。エラーが出る場合には、記述において文法的な間違いがあります。エラーメッセージをヒントに記述を修正してください。修正後は再びコンパイルを実行し、エラーが無くなるまで記述のデバッグを行います。

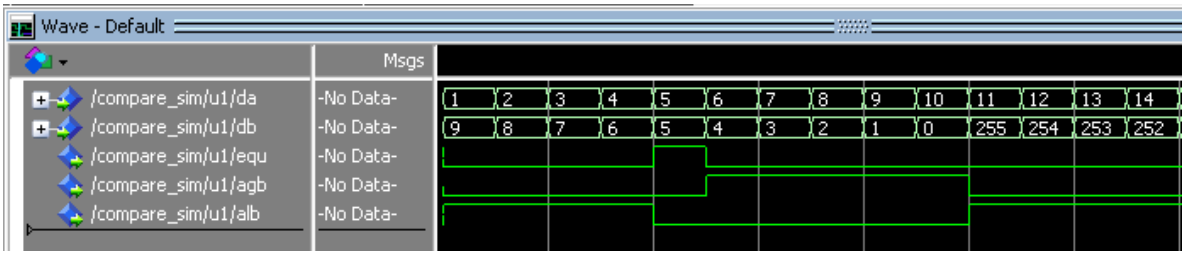
ステップ 4: ロード

1. Simulate メニュー ⇒ Start Simulation を選択し、Start Simulation ダイアログ・ボックスを起動します。
2. Design タブより work ディレクトリを展開し、**compare_sim** の下位にある **compare** を指定後 OK ボタンをクリックします。デザインのロードが開始されます。
3. ロードが問題なく終了すると、sim タブ（sim ウィンドウ）が追加され、関連ウィンドウが起動します。

ステップ 5: ファンクション・シミュレーションの実行

1. Sim ウィンドウ内の **compare_sim** の下位にある **u1** を選択し、右クリック ⇒ Add to ⇒ Wave ⇒ All items in region を選択します。すると VHDL デザインの入出力ピンが登録された状態で Wave ウィンドウが起動します。
2. Transcript ウィンドウに以下の実行コマンドを入力し、Enter キーでシミュレーション実行を開始します。

   run␣1us　　*※␣は半角スペース*
3. Wave ウィンドウにシミュレーション結果が波形表示されます。作成した回路が正しく動作しているか確認しましょう。Wave メニュー ⇒ Zoom ⇒ ･･･、またはツールバーのアイコンにて波形を明確に確認できます。信号の Radix を変更する場合は、信号選択後に右クリック ⇒ Radix ⇒ 目的の表示を選択してください。



ステップ 6: シミュレーションおよびプロジェクトの終了

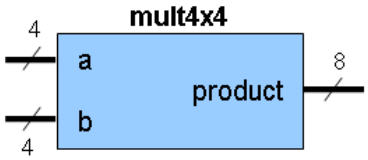
1. Simulate メニュー⇒ End Simulation よりシミュレーションを終了します。メッセージが表示されますので“はい（Y）”を選択します。
2. メイン・ウィンドウの Project タブをアクティブにし、File メニュー ⇒ Close Project を選択します。メッセージが表示されますので“はい（Y）”を選択し、プロジェクトを終了します。

これで、演習 2 は終了です。

演習 3

＜目的＞

- process 文で乗算器を作成します。
- ファンクション・シミュレーションを実行し、動作を確認します。



- エンティティ名: mult4x4
- 入力ポート: a (4bit), b (4bit)
- 出力ポート: product (8bit)
- データ・タイプ: std_logic_vector
- 機能: 4x4bit 乗算器
- 作業ディレクトリ: C:¥lab¥vhdl_lab¥lab3

ステップ 1: シミュレーション用プロジェクトの作成

1. ModelSim-Altera が起動していない場合は、ModelSim-Altera を起動します。
2. File メニュー ⇒ New ⇒ Project を選択して、Create Project ダイアログ・ボックスを開きます。
3. Project Name 欄に、Project Name 欄に、**mult4x4** と記述します。Project Location 欄に **C:/lab/vhdl_lab/lab3**（作業ディレクトリ）を選択します。それ以外はデフォルト状態のままで OK ボタンをクリックします。

   Add items to the Project ウィンドウが表示されますが、今回は使用しません。Close ボタンで閉じます。

ステップ 2: デザインの作成とプロジェクトへの登録

1. File メニュー ⇒ New ⇒ Source ⇒ VHDL よりテキスト・エディタを開きます。
2. ソースコードを記述します。上記の回路仕様を参考にし、乗算器を完成させてください。
3. 記述終了後、File メニュー ⇒ Save As より、完成させた VHDL ファイルを **mult4x4.vhd** として作業ディレクトリ内に保存します。その後、**mult4x4.vhd** ファイルを閉じます。
4. Project メニュー ⇒ Add to Project ⇒ Existing File を選択し、**mult4x4.vhd** と **mult4x4_sim.vhd**（テストベンチ）を指定します。Project ウィンドウには、指定した 2 ファイルが登録されます。

ステップ 3: コンパイル

1. Compile メニュー ⇒ Compile Order にて、コンパイルの実行順序を設定します。Compile Order ウィンドウの上段に **mult4x4.vhd**、下段に **mult4x4_sim.vhd** を配置後、OK ボタンをクリックして順番を確定します。
2. Compile メニュー ⇒ Compile All により、コンパイルを実行します。エラーが出る場合には、記述において文法的な間違いがあります。エラーメッセージをヒントに記述を修正してください。修正後は再びコンパイルを実行し、エラーが無くなるまで記述のデバッグを行います。

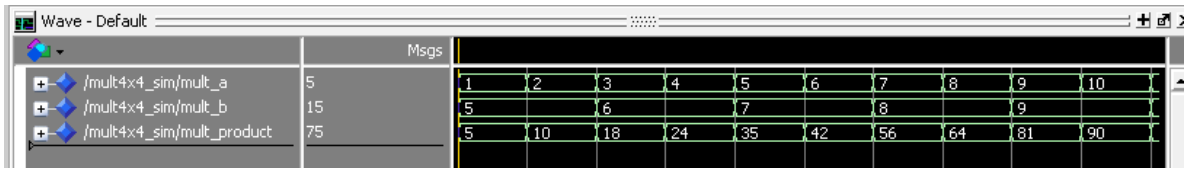
ステップ 4: ロード

1. Simulate メニュー ⇒ Start Simulation を選択し、Start Simulation ダイアログ・ボックスを起動します。
2. Design タブより work ディレクトリを展開し、**mult4x4_sim** の下位にある **mult4x4** を指定後 OK ボタンをクリックします。
3. ロードが開始され問題なく終了すると、sim タブ（sim ウィンドウ）が追加され、関連ウィンドウが起動します。

ステップ 5: ファンクション・シミュレーションの実行

1. sim ウィンドウ内の **mult4x4_sim** の下位にある **u1** を選択し、右クリック ⇒ Add to ⇒ Wave ⇒ All items in region を選択します。すると VHDL デザインの入出力ピンが登録された状態で Wave ウィンドウが起動します。
2. Transcript ウィンドウに以下の実行コマンドを入力し、Enter キーでシミュレーション実行を開始します。

   run␣1us *※␣は半角スペース*
3. Wave ウィンドウにシミュレーション結果が波形表示されます。作成した回路が正しく動作しているか確認しましょう。Wave メニュー ⇒ Zoom ⇒ ･･･、またはツールバーのアイコンにて波形を明確に確認できます。信号の Radix を変更する場合は、信号選択後に右クリック ⇒ Radix ⇒ 目的の表示を選択してください。



ステップ 6: シミュレーションおよびプロジェクトの終了

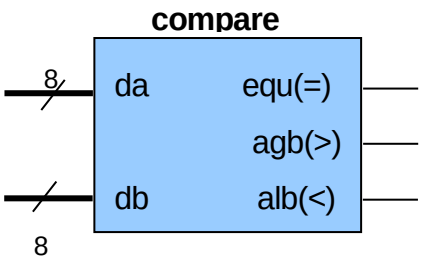
1. Simulate メニュー ⇒ End Simulation よりシミュレーションを終了します。メッセージが表示されますので “はい(Y)” を選択します。
2. Workspace ウィンドウの Project タブをアクティブにし、File メニュー ⇒ Close Project を選択します。メッセージが表示されますので “OK” を選択し、プロジェクトを終了します。

これで、演習 3 は終了です。

演習 4

＜目的＞

- 演習 2 の比較気を if-else 文で作成します。
- ファンクション・シミュレーションを実行し、動作を確認します。



- ◆ エンティティ名: compare_if
- ◆ 入力ポート: da (8bit), db (8bit)
- ◆ 出力ポート: equ, agb, alb (1bit)
- ◆ データ・タイプ: std_logic, std_logic_vector
- ◆ 機能: 比較器
- ◆ 作業ディレクトリ: C:¥lab¥vhdl_lab¥lab4

＜動作条件＞

入力信号 da と db を比較します。

- ➢ da と db が等しい場合は、equ は High (1) を出力し、それ以外のときは Low (0) を出力
- ➢ da が大きい場合は、agb は High (1) を出力し、それ以外のときは Low (0) を出力
- ➢ db が大きい場合は、alb は High (1) を出力し、それ以外のときは Low (0) を出力

ステップ 1: プロジェクトの作成

1. ModelSim-Altera が起動していない場合は、ModelSim-Altera を起動します。
2. File メニュー ⇒ New ⇒ Project を選択して、Create Project ダイアログ・ボックスを開きます。
3. Project Name 欄に、Project Name 欄に、**compare_if** と記述します。Project Location 欄に **C:/lab/vhdl_lab/lab4**（作業ディレクトリ）を選択します。それ以外はデフォルト状態のままで OK ボタンをクリックします。

   Add items to the Project ウィンドウが表示されますが、今回は使用しません。Close ボタンで閉じます。

ステップ 2: デザインの作成とプロジェクトへの登録

1. File メニュー ⇒ New ⇒ Source ⇒ VHDL よりテキスト・エディタを開きます。
2. ソースコードを記述します。上記の回路仕様を参考にし、比較器を完成させてください。
3. 記述終了後、File メニュー ⇒ Save As より、完成させた VHDL ファイルを **compare_if.vhd** として作業ディレクトリ内に保存します。その後、**compare_if.vhd** ファイルを閉じます。
4. Project メニュー ⇒ Add to Project ⇒ Existing File を選択し、**compare_if.vhd** と **compare_if_sim.vhd**（テストベンチ）を指定します。Project ウィンドウには、指定した 2 ファイルが登録されます。

ステップ 3: コンパイル

1. Compile メニュー ⇒ Compile Order にて、コンパイルの実行順序を設定します。Compile Order ウィンドウの上段に **compare_if.vhd**、下段に **compare_if_sim.vhd** を配置後、OK ボタンをクリックして順番を確定します。
2. Compile メニュー ⇒ Compile All により、コンパイルを実行します。エラーが出る場合には、記述において文法的な間違いがあります。エラーメッセージをヒントに記述を修正してください。修正後は再びコンパイルを実行し、エラーが無くなるまで記述のデバッグを行います。

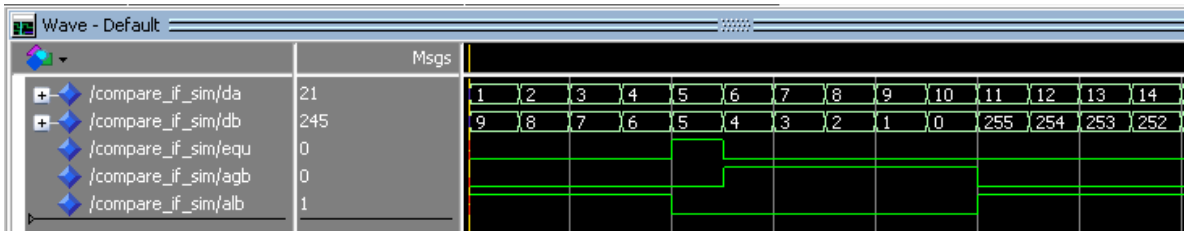
ステップ 4: ロード

1. Simulate メニュー ⇒ Start Simulation を選択し、Start Simulation ダイアログ・ボックスを起動します。
2. Design タブより work ディレクトリを展開し、**compare_if_sim** の下位にある **compare_if** を指定後 OK ボタンをクリックします。
3. ロードが開始され問題なく終了すると、sim タブ（sim ウィンドウ）が追加され、関連ウィンドウが起動します。

ステップ 5: ファンクション・シミュレーションの実行

1. sim ウィンドウ内の **compare_if_sim** の下位にある **u1** を選択し、右クリック ⇒ Add to ⇒ Wave ⇒ All items in region を選択します。すると VHDL デザインの入出力ピンが登録された状態で Wave ウィンドウが起動します。
2. Transcript ウィンドウに以下の実行コマンドを入力し、Enter キーでシミュレーション実行を開始します。

   run␣1us *※␣は半角スペース*
3. Wave ウィンドウにシミュレーション結果が波形表示されます。作成した回路が正しく動作しているか確認しましょう。Wave メニュー ⇒ Zoom ⇒ ･･･、またはツールバーのアイコンにて波形を明確に確認できます。信号の Radix を変更する場合は、信号選択後に右クリック ⇒ Radix ⇒ 目的の表示を選択してください。



ステップ 6: シミュレーションおよびプロジェクトの終了

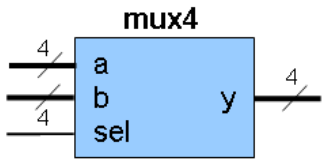
1. Simulate メニュー ⇒ End Simulation よりシミュレーションを終了します。メッセージが表示されますので “はい(Y)” を選択します。
2. Workspace ウィンドウの Project タブをアクティブにし、File メニュー ⇒ Close Project を選択します。メッセージが表示されますので “OK” を選択し、プロジェクトを終了します。

これで、演習 4 は終了です。

演習 5

＜目的＞

- Case 文を使用し、マルチプレクサを作成します。
- ファンクション・シミュレーションを実行し、動作を確認します。



- ◆ エンティティ名: mux4
- ◆ 入力ポート: a (4bit), b (4bit), sel (1bit)
- ◆ 出力ポート: y (4bit)
- ◆ データ・タイプ: std_logic, std_logic_vector
- ◆ 機能: 2 to 1 マルチプレクサ
- ◆ 作業ディレクトリ: C:¥lab¥vhdl_lab¥lab5

＜動作条件＞

- ➢ セレクト・コントロール信号 (sel) が Low (0) ならば a[3..0] を出力
- ➢ セレクト・コントロール信号 (sel) が High (1) ならば b[3..0] を出力

ステップ 1: プロジェクトの作成

1. ModelSim-Altera が起動していない場合は、ModelSim-Altera を起動します。
2. File メニュー ⇒ New ⇒ Project を選択して、Create Project ダイアログ・ボックスを開きます。
3. Project Name 欄に、Project Name 欄に、**mux4** と記述します。Project Location 欄に **C:/lab/vhdl_lab/lab5**（作業ディレクトリ）を選択します。それ以外はデフォルト状態のままで OK ボタンをクリックします。

   Add items to the Project ウィンドウが表示されますが、今回は使用しません。Close ボタンで閉じます。

ステップ 2: デザインの作成とプロジェクトへの登録

1. File メニュー ⇒ New ⇒ Source ⇒ VHDL よりテキスト・エディタを開きます。
2. ソースコードを記述します。上記の回路仕様を参考にし、マルチプレクサを完成させてください。
3. 記述終了後、File メニュー ⇒ Save As より、完成させた VHDL ファイルを **mux4.vhd** として作業ディレクトリ内に保存します。その後、**mux4.vhd** ファイルを閉じます。
4. Project メニュー ⇒ Add to Project ⇒ Existing File を選択し、**mux4.vhd** と **mux4_sim.vhd**（テストベンチ）を指定します。Project ウィンドウには、指定した 2 ファイルが登録されます。

ステップ 3: コンパイル

1. Compile メニュー ⇒ Compile Order にて、コンパイルの実行順序を設定します。Compile Order ウィンドウの上段に **mux4.vhd**、下段に **mux4_sim.vhd** を配置後、OK ボタンをクリックして順番を確定します。
2. Compile メニュー ⇒ Compile All により、コンパイルを実行します。エラーが出る場合には、記述において文法的な間違いがあります。エラーメッセージをヒントに記述を修正してください。修正後は再びコンパイルを実行し、エラーが無くなるまで記述のデバッグを行います。

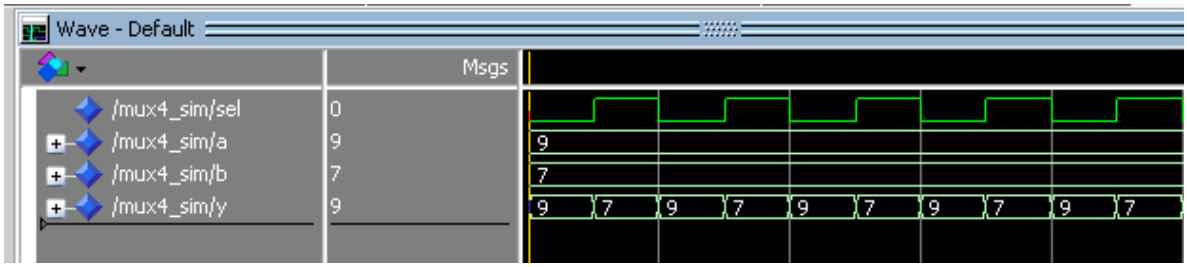
ステップ 4: ロード

1. Simulate メニュー ⇒ Start Simulation を選択し、Start Simulation ダイアログ・ボックスを起動します。
2. Design タブより work ディレクトリを展開し、**mux4_sim** の下位にある **mux4** を指定後 OK ボタンをクリックします。
3. ロードが開始され問題なく終了すると、sim タブ（sim ウィンドウ）が追加され、関連ウィンドウが起動します。

ステップ 5: ファンクション・シミュレーションの実行

1. sim ウィンドウ内の **mux4_sim** の下位にある **u1** を選択し、右クリック ⇒ Add to ⇒ Wave ⇒ All items in region を選択します。すると VHDL デザインの入出力ピンが登録された状態で Wave ウィンドウが起動します。
2. Transcript ウィンドウに以下の実行コマンドを入力し、Enter キーでシミュレーション実行を開始します。

   run␣1us *※␣は半角スペース*
3. Wave ウィンドウにシミュレーション結果が波形表示されます。作成した回路が正しく動作しているか確認しましょう。Wave メニュー ⇒ Zoom ⇒ ･･･、またはツールバーのアイコンにて波形を明確に確認できます。信号の Radix を変更する場合は、信号選択後に右クリック ⇒ Radix ⇒ 目的の表示を選択してください。



ステップ 6: シミュレーションおよびプロジェクトの終了

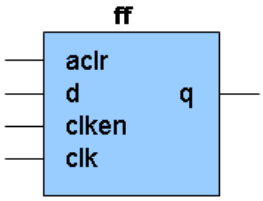
1. Simulate メニュー ⇒ End Simulation よりシミュレーションを終了します。メッセージが表示されますので “はい(Y)” を選択します。
2. Workspace ウィンドウの Project タブをアクティブにし、File メニュー ⇒ Close Project を選択します。メッセージが表示されますので “OK” を選択し、プロジェクトを終了します。

これで、演習 5 は終了です。

演習 6

＜目的＞

- 非同期クリア & クロック・イネーブル付きフリップフロップを作成します。
- ファンクション・シミュレーションを実行し、動作を確認します。



- ◆ エンティティ名: ff
- ◆ 入力ポート: clk (1bit), aclr (1bit), clken (1bit)
- ◆ 入力ポート: d (1bit)
- ◆ 出力ポート: q (1bit)
- ◆ データ・タイプ: std_logic
- ◆ 機能: 非同期クリア&クロック・イネーブル付きフリップフロップ
- ◆ 作業ディレクトリ: C:¥lab¥vhdl_lab¥lab6

＜動作条件＞

- クリア信号(aclr) が Low (0) のとき、フリップフロップは Low (0) を出力する（クリアされる）
- クリア信号(aclr) が High (1)、且つクロック・イネーブル信号 (clken) が High (1) のとき、出力 q は 入力 d を出力する

ステップ 1: プロジェクトの作成

1. ModelSim-Altera が起動していない場合は、ModelSim-Altera を起動します。
2. File メニュー ⇒ New ⇒ Project を選択して、Create Project ダイアログ・ボックスを開きます。
3. Project Name 欄に、Project Name 欄に、**ff** と記述します。Project Location 欄に **C:/lab/vhdl_lab/lab6**（作業ディレクトリ）を選択します。それ以外はデフォルト状態のままで OK ボタンをクリックします。

   Add items to the Project ウィンドウが表示されますが、今回は使用しません。Close ボタンで閉じます。

ステップ 2: デザインの作成とプロジェクトへの登録

1. File メニュー ⇒ New ⇒ Source ⇒ VHDL よりテキスト・エディタを開きます。
2. ソースコードを記述します。上記の回路仕様を参考にし、フリップフロップを完成させてください。
3. 記述終了後、File メニュー ⇒ Save As より、完成させた VHDL ファイルを **ff.vhd** として作業ディレクトリ内に保存します。その後、**ff.vhd** ファイルを閉じます。
4. Project メニュー ⇒ Add to Project ⇒ Existing File を選択し、**ff.vhd** と **ff_sim.vhd**（テストベンチ）を指定します。Project ウィンドウには、指定した 2 ファイルが登録されます。

ステップ 3: コンパイル

1. Compile メニュー ⇒ Compile Order にて、コンパイルの実行順序を設定します。Compile Order ウィンドウの上段に **ff.vhd**、下段に **ff_sim.vhd** を配置後、OK ボタンをクリックして順番を確定します。

2. Compile メニュー ⇒ Compile All により、コンパイルを実行します。エラーが出る場合には、記述において文法的な間違いがあります。エラーメッセージをヒントに記述を修正してください。修正後は再びコンパイルを実行し、エラーが無くなるまで記述のデバッグを行います。

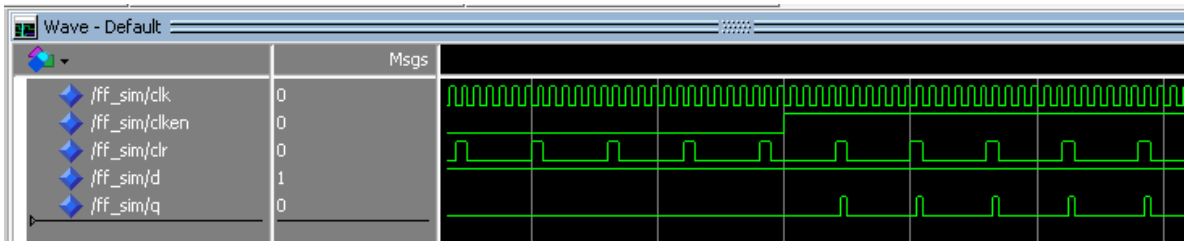
ステップ 4: ロード

1. Simulate メニュー ⇒ Start Simulation を選択し、Start Simulation ダイアログ・ボックスを起動します。
2. Design タブより work ディレクトリを展開し、**ff_sim** の下位にある **ff** を指定後 OK ボタンをクリックします。
3. ロードが開始され問題なく終了すると、sim タブ（sim ウィンドウ）が追加され、関連ウィンドウが起動します。

ステップ 5: ファンクション・シミュレーションの実行

1. sim ウィンドウ内の **ff_sim** の下位にある **ff** を選択し、右クリック ⇒ Add to ⇒ Wave ⇒ All items in region を選択します。すると VHDL デザインの入出力ピンが登録された状態で Wave ウィンドウが起動します。
2. Transcript ウィンドウに以下の実行コマンドを入力し、Enter キーでシミュレーション実行を開始します。

   run␣1us *※␣は半角スペース*
3. Wave ウィンドウにシミュレーション結果が波形表示されます。作成した回路が正しく動作しているか確認しましょう。Wave メニュー ⇒ Zoom ⇒ ･･･、またはツールバーのアイコンにて波形を明確に確認できます。信号の Radix を変更する場合は、信号選択後に右クリック ⇒ Radix ⇒ 目的の表示を選択してください。



ステップ 6: シミュレーションおよびプロジェクトの終了

1. Simulate メニュー ⇒ End Simulation よりシミュレーションを終了します。メッセージが表示されますので “はい(Y)” を選択します。
2. Workspace ウィンドウの Project タブをアクティブにし、File メニュー ⇒ Close Project を選択します。メッセージが表示されますので “OK” を選択し、プロジェクトを終了します。

これで、演習 5 は終了です。

以上で本コースの演習はすべて終了です。お疲れ様でした。

改版履歴

Revision	年月	概要
1	2016 年 4 月	・ 初版（Web 対応）