第5章 演算回路の前後にフリップフリップを実装してパイプライン化する

高速化のためのパイプライン化技法

小野 雅晃 Masaaki Ono

第3章～第4章までで，高位合成からFPGAへの実装までの一通りの操作方法が体験できたと思います．第5章では，乗算回路を高速に動作させるためのテクニックについて解説します．もっとも一般的な高速化手法として，回路のパイプライン化があります．ここでは各種指示子の使い方を取り上げ，動作時のタイミング波形の違いなどについても詳しく解説します．最終的に400MHzを超えるクロック周波数で動作する回路を合成できました．

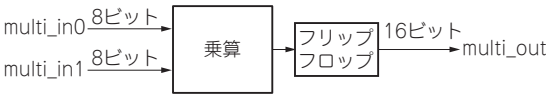

図1　乗算回路の出力にフリッププロップを挿入

● 本章でやること

ここでは，高位合成で生成した回路をより高速に動作させるために，パイプライン化に挑戦してみます．第3章で作成した乗算回路はクロックを必要としない組み合わせ回路でした．これをパイプライン化するための準備として，乗算回路の入力と出力にフリップフロップ(FF)を実装します．パイプラインとは何か，なぜFFが必要なのか不明な場合は，後述している「5.逐次処理とパイプライン処理」(p.63)を先に読んでみてください．乗算回路にFFを実装すると，同時にクロックとリセット信号が追加されます．

次に，乗算回路にPIPELINE指示子を挿入します．乗算回路にFFを追加すると，次のデータを入力するまでに3クロック必要になってしまいますが，PIPELINE指示子を挿入することで，1クロック後に次のデータが入力できるようになります．

最後に，新規にSolutionを生成して動作周波数を変更します．Vivado HLS最大の利点である簡単に動作周波数を変更できる機能を確認してみましょう．

● 作業手順

本章での作業手順の概要を列挙します．今回はC/RTL協調シミュレーション波形を確認するためにVivado 2016.1を使用する以外は，全てVivado HLS 2016.1で作業を行います．

(1) 乗算回路の出力にFFを挿入
- 指示子を変更しCコードの合成を行う
- C/RTL協調シミュレーションを行い波形を観察

(2) 乗算回路の入力と出力にFFを挿入
- 指示子を変更しCコードの合成を行う
- C/RTL協調シミュレーションを行い波形を観察

(3) PIPELINE指示子を挿入
- 指示子を変更しCコードの合成を行う
- C/RTL協調シミュレーションを行い波形を観察
- 新規Solutionを作成し，動作周波数を変更してHDLへの合成を行います．

● 次に進む前にバックアップをとる

これから入力や出力にレジスタを追加すると，ap_clk, ap_rstがHDLに追加され，Vivadoで作った乗算器(multi_ex1)プロジェクトがコンパイルできなくなります．Vivado HLSのプロジェクトをバックアップしておくために，Vivado HLSのプロジェクト(multi_apuint)ごと，どこかのフォルダにコピーしておいてください．

1．乗算回路の出力にフリップフロップを挿入

● 指示子を変更しCコードの合成

乗算回路の出力にFF（レジスタとも呼ぶ）を挿入します．そのブロック図を**図1**に示します．

乗算回路の出力にFFを挿入するには，INTERFACE指示子にregisterオプションを付けます．Vivado HLS 2016.1を開いて，右側のRecent Projectsウィンドウ内の第3章で使用したmulti_apuintプロジェクトをクリックします[**図2(a)**]．

左端のExplorerのSourceをダブルクリックして展開します．その中のmulti_apuint.cppをクリックします[**図2(b)**]．

次にmulti_apuint.cppが開きます．右端のウィンドウでDirectiveタブをクリックして変更します．multi_outの下の指示子を右クリックし，ポップアップ・メニューからModify Directiveを選択します[**図2(c)**]．