第8章　高速UARTコアを使って相互通信

分散処理対応プラットフォームの構築

横山 雅一 Msakazu Yokoyama

ここでは2枚のZynqボードを相互接続し，より高度な処理を行う事例を紹介します．
まず，ボード間の通信で使う高速UARTコアを使って相互通信が可能なことを確認します．次に，SDSoCのプラットフォームに追加し，センサのデータを送受信する簡単なアプリケーションを作成します．最後に2枚のボードをより効果的に活用する事例として，画像処理を行います．

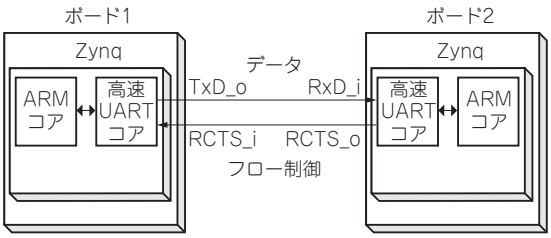

図1　高速UARTコアを実装して2枚のボードを相互接続

準備…ボード間通信対応プラットフォームの作成

2枚のZynqボードで分散処理を行うためのハードウェア構成を**図1**に示します．2枚のボードはシリアル接続します．

● 高速UARTコアのテスト

相互接続で用いる高速UARTコア（詳細は第8章Appendixで解説）を追加します．送信受信のためのI/Oを追加して合成します（**リスト1**）．

まず，ループバック・テストを行います．高速UARTコアは送信側が受信側のRCTS（受信可能）信号を見ながら通信をするハードウェア・フロー制御が実装されています．単純なループバック・テスト・プログラムにより実際に動作することを確認します．

TxACTとRxACTは負論理で動く動作確認用の信号です．送信時と受信時にそれぞれ変化するので外部に引き出しLEDに接続することで実際の通信を目視で確認できます．テスト時にはボード上のLEDに接続しました．

このように，SDSoCのプラットフォーム構築の前にはVivadoでのプロジェクトの作成と単体機能の確認は必須となります．動くシステムを作り上げるには，ひとつひとつ課題をクリアしていくことが重要です．

● 高速UARTコアをプラットフォームに追加する

テストの終わったVivadoのプロジェクトをベースに，SDSoC用のプラットフォームを作成します．

基本的な作成方法は第3章と第4章の通りです．加えて，高速UARTコアのために，**リスト2**のtclコマンドを実行しUIOからアクセス可能な設定としました．

さらにデバイス・ツリーには**リスト3**の記述を追加します．これでLinuxからはUIOを使うことで高速UIOのレジスタ群へとアクセスできます．

リスト1　相互接続のためのピン配置指定（制約ファイル）

```
set_property PACKAGE_PIN N15 [get_ports pin_TxD_o]
set_property PACKAGE_PIN N16 [get_ports pin_RCTS_i]

set_property PACKAGE_PIN H16 [get_ports pin_RxD_i]
set_property PACKAGE_PIN H17 [get_ports pin_RCTS_o]

set_property IOSTANDARD LVCMOS33 [get_ports pin_RCTS_o]
set_property IOSTANDARD LVCMOS33 [get_ports pin_RCTS_i]
set_property IOSTANDARD LVCMOS33 [get_ports pin_RxD_i]
set_property IOSTANDARD LVCMOS33 [get_ports pin_TxD_o]

set_property PACKAGE_PIN G17 [get_ports pin_nRxACT_o]
set_property PACKAGE_PIN M15 [get_ports pin_nTxACT_o]
set_property IOSTANDARD LVCMOS33 [get_ports pin_nRxACT_o]
set_property IOSTANDARD LVCMOS33 [get_ports pin_nTxACT_o]
```