TOE10G-IP 標準(CPU 制御)デザイン説明書 (Intel 版)

Rev1.0J 2018/05/07

1 TCP/IP プロトコル概要

TOE10G-IP コアを使って実装するネットワーク・システムにおいて、TCP/IP は 4 つの層(レイヤ)から成るネットワーク・アプリケーションのインターネット・プロトコル群において中核となるプロトコルです。4 つの層とはすなわちアプリケーション層、トランスポート層、インターネット層、ネットワーク・アクセス層です。ただし各層を説明する図 1-1 においては、TOE10G-IP コアによる FPGA でのハードウエア実装と 1 対 1 に合致させるために 5 層で示しています。 ネットワーク・アクセス層はリンク層と物理層に分割して示しています。

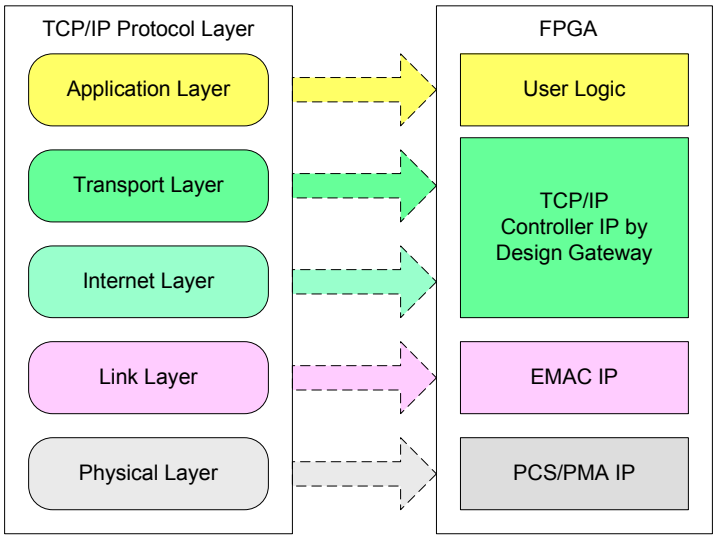


図 1-1: TCP/IP プロトコルのレイヤ図

TOE10G-IP コアは TCP/IP プロトコルにてトランスポート層とインターネット層を実装します。 送信機能において TOE10G-IP コアは、ユーザ回路からの TCP データをパケットのフォーマットに変換し IP ヘッダを生成して EMAC から外部に送信します。 受信機能において TOE10G-IP コアは、IP パケットから TCP データとヘッダを抽出しユーザ回路からリードするために TCP データだけを正しく抜き出してバッファに格納します。

プロトコルの下位層は Intel 社の EMAC-IP コアおよび PCS/PMA-IP コアにより実装されます。

本デモ・デザインは TOE10G-IP コアによるデータの送受信をシンプルなユーザ回路と合わせて実装し実機評価を可能とするデザインです。ユーザ・インターフェイス用として CPU システムが使われ JTAG UART 経由で通信します。ファームウエアはベアメタル OS でデザインされています。本デモでは 2 つのテスト・アプリケーションが使われます、一つは半二重通信用の”tcpdatatest”でもうひとつは全二重通信用の”tcp_client_txrx_10G”です。 本デザインでは Intel 社 FPGA 評価ボードで実機動作し超高速通信のパフォーマンスやデータ信頼性を評価できます。より詳細は以下で説明します。

2　ハードウエアの説明

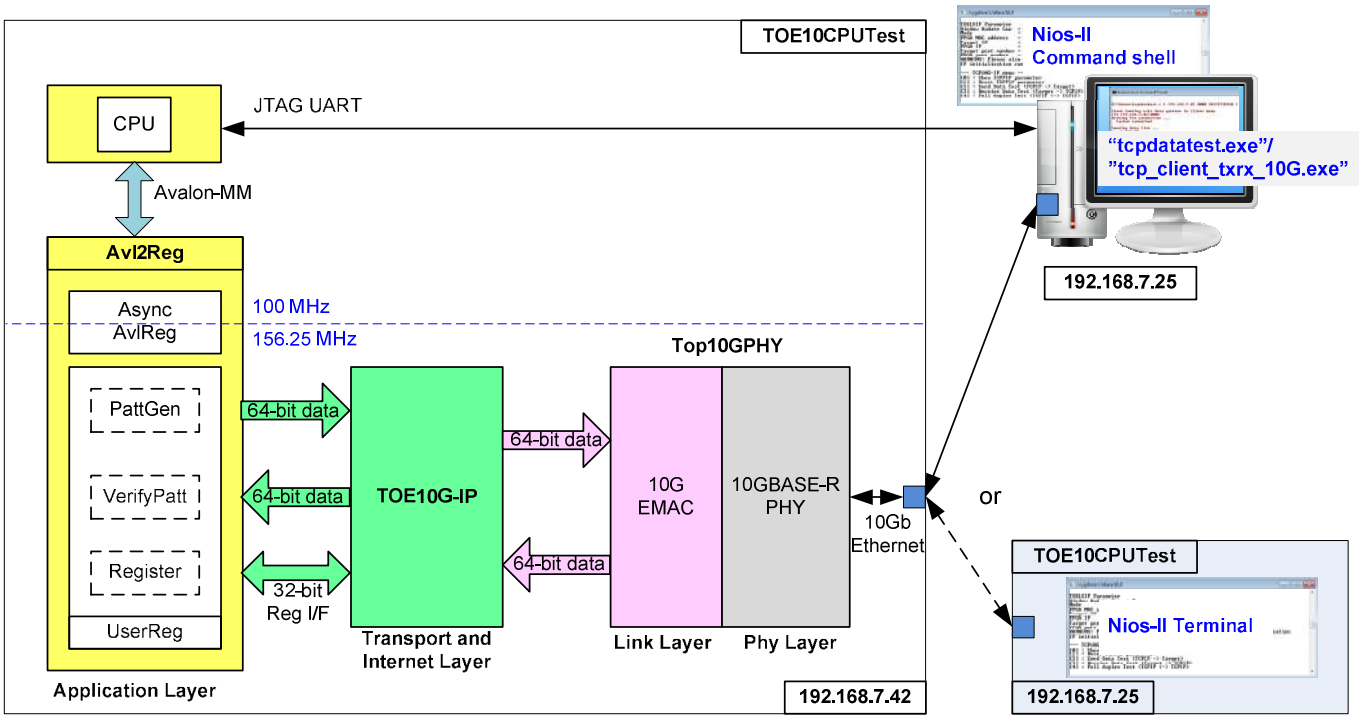


図 2-1: デモ・デザイン全体のブロック図

テスト環境としては 10Gb イーサネット転送に 2 つのネットワーク・デバイスが使われます。一つはクライアント・モードで動作しもう一つはサーバー・モードで動作します。 TOE10G-IP コアをそれぞれのモードで動作検証するため、デモでは図 2-1 に示すように 2 種類の環境が使われます。ひとつは 2 枚の FPGA ボードを使った環境です（片方はクライアントでもう片方はサーバーで使います）、そしてもうひとつは 1 枚の FPGA ボードと PC を使った環境です。

FPGA 内部ロジックでは TOE10G-IP コアは 10G イーサネット MAC および 10GBASE-R PHY と接続することで TCP/IP レイヤを実装します。 TOE10G-IP コアのユーザ・インターフェイスは Avl2Reg モジュールと接続します。TOE10G-IP コアのデータ・インターフェイスではテスト・パターンを発生する PattGen モジュールを内蔵した UserReg モジュールと接続します。 また、TOE10G-IP コアから受信したデータをベリファイするため VerifyPatt を内蔵します。本デモ・デザインでのテスト・パターンは 32 ビットのインクリメンタル・データです。

制御インターフェイスでは Avl2Reg モジュールがユーザ回路からのテスト・パラメータたとえば転送長や転送方向などを保持するためのレジスタを内蔵します。 ユーザは NiosII コマンド・シェル経由でパラメータを入力します。 CPU ファームウエアは全パラメータを確認し Avalon-MM バスを通してハードウエアにセットします。 CPU システムと TOE10G-IP コアは異なるクロック・ドメインで動作するため、非同期用の回路として AsyncAvlReg モジュールが使われそこでクロック交差機能と CPU システムの Avalon-MM バスをレジスタ・インターフェイスに変換します。本デモにおける CPU はベアメタル OS で動作します。また、CPU システムにはタイマが含まれており転送パフォーマンスを計測します。

本デモにおいて PC 側では 2 種類のアプリケーションが使われます。 ひとつは"tcpdatatest.exe"で TOE10G-IP コアとイーサネットを送信または受信するために使われます。データ転送は半二十通信です。もうひとつのアプリケーションは"tcp_client_txrx_10G.exe"で送受信同時（全二重通信）で同一の TCP ポートを使ってイーサネット・データを送受信します。この全二重通信モードでは UserReg モジュール内の PattGen および VerifyPatt モジュールが送受信両方向で同時に TOE10G-IP コアとデータを転送します。

2.1 10G イーサネット MAC および 10GBASE-R PHY

本デモ・デザインのリンク層と物理層は Intel FPGA の IP コアにて実装されます。Arria10FPGA の場合、低レイテンシ 10G イーサネット MAC が TOE10G-IP コアと 10GBASE-R PHY を接続するために使われます。より詳細については以下リンク先を参照してください。
https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-10gbps-ethernet-mac.html

10GBASE-R PHY は 10G SFP+モジュールと接続するために使われます。その反対側は 10G イーサネット MAC と 156.25MHz で動作する XGMII インターフェイスで接続されます。より詳細については以下リンク先を参照してください。
https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-10gbase-r-pcs.html

2.2 TOE10G-IP コア

TOE10G-IP コアは TCP/IP スタックおよびオフロード・エンジンを実装します。コアの制御ステータス信号はレジスタ・インターフェイスを介してアクセスします。データ・インターフェイスは FIFO インターフェイスを介します。より詳細についてはコアのデータシートを参照してください。

http://www.dgway.com/products/IP/TOE10G-IP/dg_toe10gip_data_sheet_altera_jp.pdf

2.3 Avl2Reg モジュール

このモジュールは他の CPU 周辺回路と同じように Avalon-MM バスを通して CPU と接続します。本モジュール内のレジスタは CPU メモリ・アドレスで表 2-1 に示すようにマッピングされています。CPU からの制御レジスタおよびステータス・レジスタへのアクセスは Avl2Reg モジュール内でデザインされています。

本 Avl2Reg モジュールはデータ・パスおよび制御パスのどちらも TOE10G-IP コアと接続します。図 2-2 に示すように本ブロックには 2 つのクロック・ドメインが存在します、すなわち一つは Avalon-MM バスを通して CPU とインターフェイスする 100MHz(CpuClk)ドメインで、もう一つは TOE10G-IP および EMAC 向けのユーザ・クロック・ドメインとなる 156.25MHz(MacClk)ドメインです。

AsyncAvlReg モジュールは 100MHz と 156.25MHz の間の非同期接続回路を内蔵します。各モジュールのより詳細を以下に説明します。

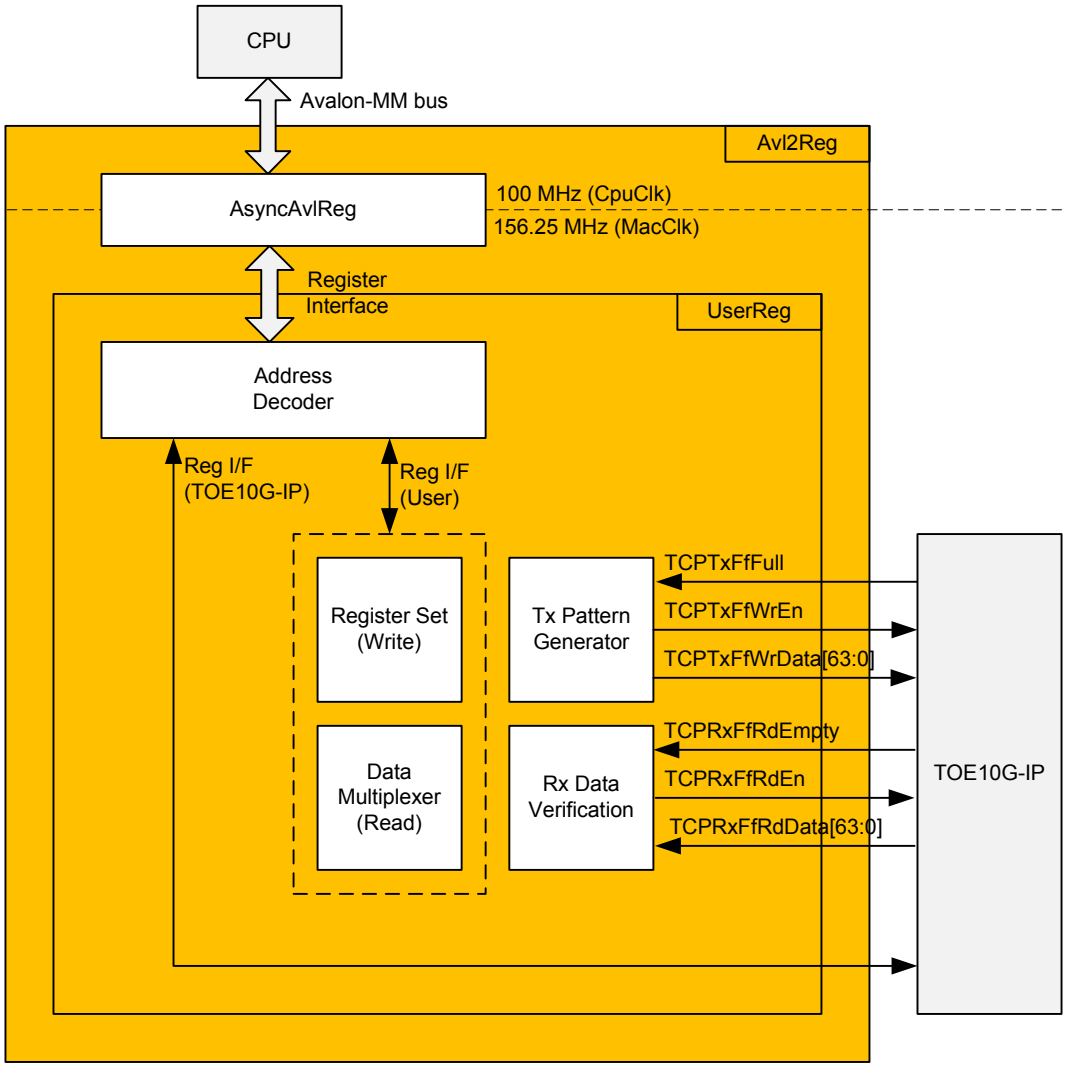


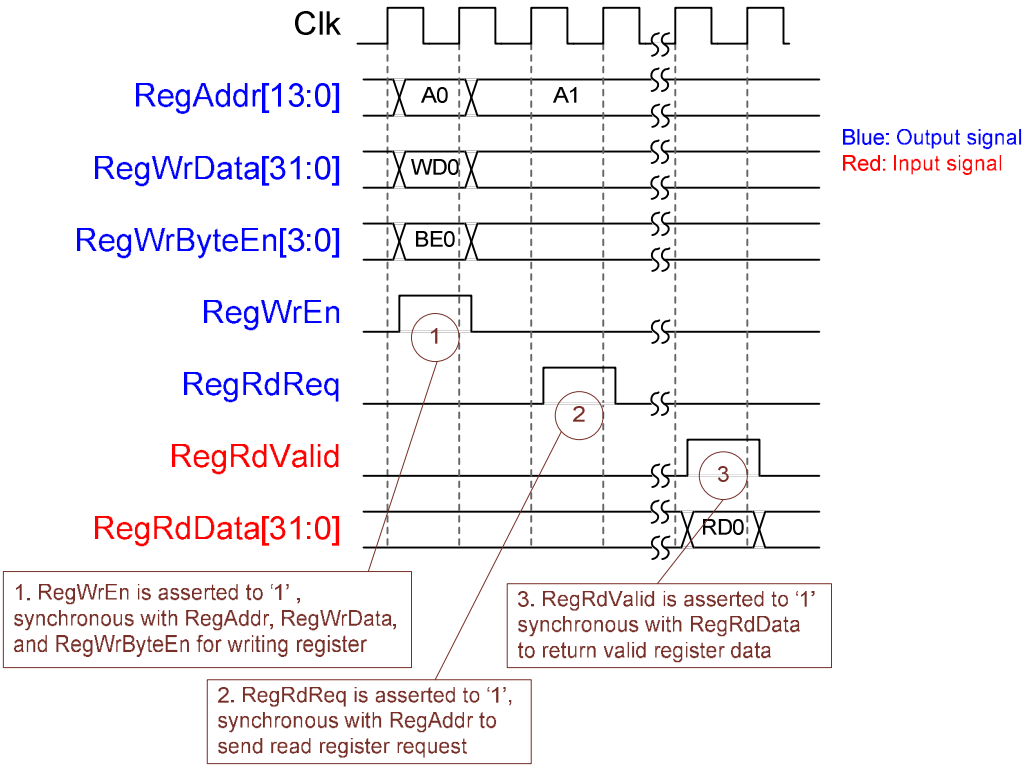
図 2-2: Avl2Reg モジュールのブロック図

2.3.1 AsyncAvlReg モジュール

本モジュールは Avalon-MM インターフェイス信号をレジスタ・インターフェイスに変換します。さらに、100MHz を 156.25MHz のクロック・ドメインに変換します。レジスタ・インターフェイスのタイミング波形を図 2-3 に示します。

レジスタへのライトにおいてのタイミング波形は RAM インターフェイスと同じです。RegWrEn が'1'アサートされるのとともに、有効な RegAddr(32 ビット単位のレジスタ・アドレス)および RegWrData(レジスタへのライト・データ)および RegWrByteEn(アクセスのバイト・イネーブル：bit[0]が RegWrData[7:0]のイネーブルに対応、bit[1]が RegWrData[15:8]のイネーブルに対応、同様に bit[3]が RegWrData[31:24]のイネーブルに対応)がアサートされます。

レジスタからのリードにおいては AsyncAvlReg モジュールが RegRdReq='1'アサートされるとともに、有効な RegAddr(32 ビット単位のレジスタ・アドレス)とともにアサートされます。その後モジュールは RegRdValid が'1'アサートされるのを待機し RegRdData 信号を通してデータを読み取ります。



① レジスタへの書込みにおいて RegWrEn は RegAddr,RegWrData,RegWrByteEn に同期して'1'アサートされる。
② レジスタの読み出しにおいて RegRdReq は RegAddr に同期して'1'アサートされリードが要求される。
③ RegRdValid は RegRdData と同期して有効なレジスタ読み出しデータが出力される。

図 2-3: レジスタ・インターフェイスのタイミング波形

2.3.2 UserReg モジュール

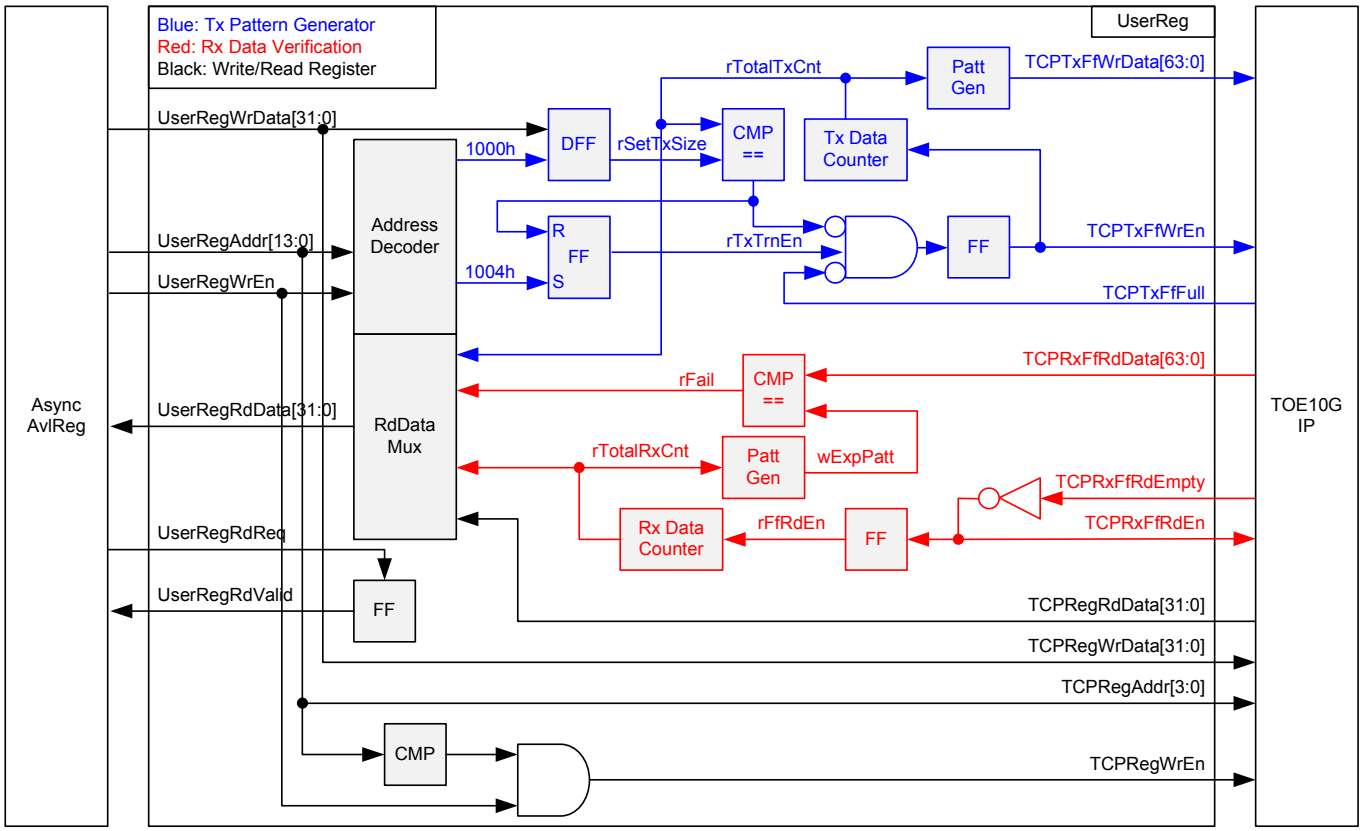


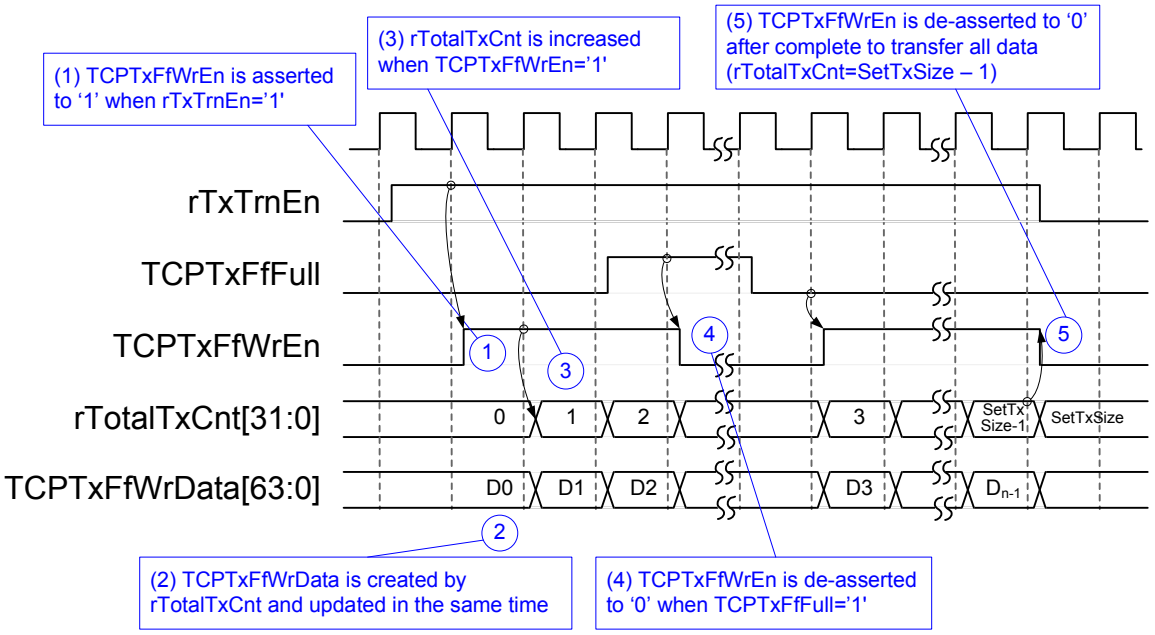
図 2-4: UserReg モジュールのブロック図

UserReg モジュール内の制御およびステータス信号のメモリ・マップを表 2-1 に示します。
0x0000 – 0x00FF は TOE10G-IP コアの内部レジスタにマップされます。
0x1000 – 0x10FF は UserReg モジュール内部レジスタにマップされます。(送信パターン生成および受信データのベリファイに使われます。).

レジスタへのライト動作では、有効な UserRegAddr と合わせて UserRegWrEn が'1'アサートされます。UserRegAddr の上位ビットはアクセス先が TOE10G-IP コアかそうでないかを判断するために使われます。アドレスが TOE10G-IP コア内の場合 TCPRegWrEn が'1'アサートされます。一方 UserReg 内部レジスタの場合、UserRegWrData はアドレスが合致する内部レジスタにロードされます。例えば UserRegAddr=0x1000 の場合 UserRegWrData は rSetTxSize にロードされます。本モジュールでは UserRegWrByteEn 信号は使われないため、CPU ファームウエアは常に 32 ビットでレジスタをアクセスする必要があります。

レジスタのリード動作では、UserRegRdReq が'1'アサートされます。そして RdDataMux が内部レジスタか TOE10G-IP コア内レジスタかのステータス信号を選択し、次クロックで UserRegRdData へと転送されます。UserRegRdData と同期するため RegRdValid は RegRdReq 信号入力を 1 段の D フリップフロップで経由して生成されます。

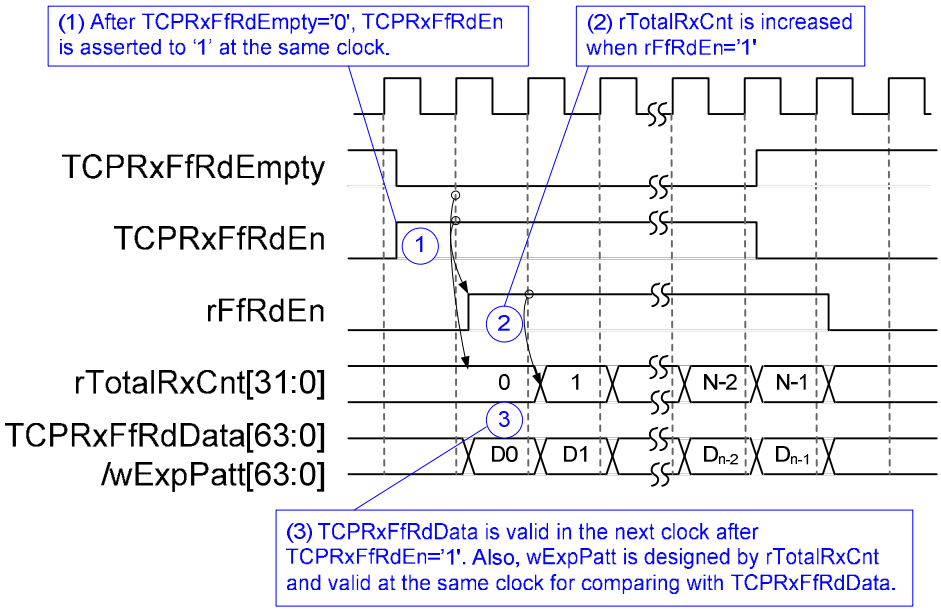
図 2-4 の上側青色のロジックは TOE10G-IP コアへのテスト・データ生成用です。rTxTrnEn はライト・レジスタのアドレスが 1004h の場合に'1'アサートされます。この rTxTrnEn が'1'になると TCPTxFfWrEn は図 2-5 に示すように TCPTxFfFull により制御されます。そして TCPTxFfFull が'1'となると TCPTxFfWrEn は'0'ネゲートされます。rTotalTxCnt はデータ・カウンタで TCPTxFf へ転送する総データ数をカウントします。rTotalTxTxCnt は、また TCPTxFfWrDate への 32 ビット・インクリメンタル・データを生成するためにも使われます。総転送サイズが設定した値(rSetTxSize)に一致すると rTxTrnEn は'0'ネゲートします。



① TCPTxFfWrEn は rTxTmEn='1'アサートにより'1'アサートされる。
② それと同時に TCPTxFfWrData が rTotalTxCnt から生成され更新される。
③ TCPTxFfWrEn='1'アサートされると rTotalTxCnt がインクリメントされる。
④ TCPTxFfFull が'1'アサートされた場合、TCPTxFfWrEn はネゲートされる。
⑤ 全データの転送が完了（rTotalTxCnt が SetTxSize-1 と一致）すると TCPTxFfWrEn は'0'ネゲートされる。

図 2-5: 送信パターン・ジェネレータのタイミング波形

図 2-4 の赤色で示したロジックは TOE10G-IP コアから受信したデータをベリファイする部分です。TCPRxFfRdEn は TCPRxFfRdEmpty の論理を反転したものを使います。TCPRxFfRdData は TCPRxFfRdEn が'1'アサートされた次クロックで有効となります。そのリード・データ TCPRxFfRdData は rTotalRxCnt で作られた期待値(wExpPatt)と比較されます。rTotalRxCnt はデータ・カウンタで TCPRxFf から転送された総データ数をチェックします。送信パスと同様期待値パターンは 32 ビットのインクリメンタル・パターンです。リード・データが期待値と不一致であった場合は不一致フラグ(rFall)が'1'アサートされます。



① TCPRxFfRdEmpty が'0'となると同じクロック期間で TCPRxFfRdEn が'1'アサートされる。
② rFfRdEn が'1'アサートされると rTotalRxCnt がインクリメントされる。。
③ TCPRxFfRdData は TCPRxFfRdEn が'1'アサートされた次のクロック期間で有効な値が出力される。さらに rTotalRxCnt から wExpPatt が生成され同じクロック期間で TCPRxFfRdData と比較される。

図 2-6: 受信データ・ベリファイのタイミング波形

表 2-1: レジスタ・マップ定義

アドレス Wr/Rd	レジスタ名 (“toe10gtest.c”内のラベル名)	説明
BA+0x0000 – BA+0x00FF: TOE10G-IP レジスタ空間 レジスタの詳細については TOE10G-IP コア・データシートの表 3 を参照してください。		
BA+0x00	TOE10_RST_REG	TOE10G-IP 内 RST レジスタ
BA+0x04	TOE10_CMD_REG	TOE10G-IP 内 CMD レジスタ
BA+0x08	TOE10_SML_REG	TOE10G-IP 内 SML レジスタ
BA+0x0C	TOE10_SMH_REG	TOE10G-IP 内 SMH レジスタ
BA+0x10	TOE10_DIP_REG	TOE10G-IP 内 DIP レジスタ
BA+0x14	TOE10_SIP_REG	TOE10G-IP 内 SIP レジスタ
BA+0x18	TOE10_DPN_REG	TOE10G-IP 内 DPN レジスタ
BA+0x1C	TOE10_SPN_REG	TOE10G-IP 内 SPN レジスタ
BA+0x20	TOE10_TDL_REG	TOE10G-IP 内 TDL レジスタ
BA+0x24	TOE10_TMO_REG	TOE10G-IP 内 TMO レジスタ
BA+0x28	TOE10_PKL_REG	TOE10G-IP 内 PKL レジスタ
BA+0x2C	TOE10_PSH_REG	TOE10G-IP 内 PSH レジスタ
BA+0x30	TOE10_WIN_REG	TOE10G-IP 内 WIN レジスタ
BA+0x34	TOE10_ETL_REG	TOE10G-IP 内 ETL レジスタ
BA+0x38	TOE10_SRV_REG	TOE10G-IP 内 SRV レジスタ
BA+0x1000 – BA+0x10FF: UserReg モジュールの制御/ステータス・レジスタ空間		
BA+0x1000 Wr/Rd	総送信サイズ (USER_TXLEN_REG)	Wr [31:0] – QWord(64 ビット)単位の総送信サイズ、有効な値は 1 - 0xFFFFFFFF Rd [31:0] – 現在の送信済みサイズを QWord(64 ビット)単位で示す、この値は USER_CMD_REG がライトされるとクリアされる。
BA+0x1004 Wr/Rd	ユーザ・コマンド (USER_CMD_REG)	Wr [0] – 送信開始、’1’ライトで送信動作を開始する このビットは全転送の送信完了後’0’クリアされる [1] – データ・ベリファイのディスエーブル(無効化) (‘0’: ベリファイをイネーブル, ‘1’: ベリファイをディスエーブル) Rd [0] – 送信ビジー (‘0’: アイドル, ‘1’: 送信モジュールはビジー) [1] – データ・ベリファイ・エラー (‘0’: 通常, ‘1’: エラー発生) このビットはユーザが次のコマンドを開始するかリセットで’0’クリアされる [2] – TOE10G-IP コアの ConnOn 信号にマップされる
BA+0x1008 Wr/Rd	ユーザ・リセット (USER_RST_REG)	Wr [0] – リセット信号、’1’にセットするとロジックをリセットする このビットは自動的に’0’クリアされる [8] – TimerInt ラッチ値を’1’でクリアする Rd [8] – IP コアからの TimerInt 出力のラッチ値 (‘0’: 通常, ‘1’: TimerInt=’1’が検出された) このフラグはシステムがリセットされるか USER_RST_REG[8]=’1’でクリアされる。
BA+0x100C Rd	FIFO ステータス (FIFO_STS_REG)	Rd [2:0]: IP コアの TCPRxFfLastRdCnt 信号にマップされる [15:3]: IP コアの TCPRxFfRdCnt 信号にマップされる [24]: IP コアの TCPTxFfFull 信号にマップされる
BA+0x1010 Rd	総受信サイズ (TRN_RXLEN_REG)	Rd [31:0] – 現在の総受信サイズを QWord(64 ビット)単位で示す、この値は USER_CMD_REG がライトされるとクリアされる

3 CPU ファームウエアのシーケンス

FPGA がブートアップした後にユーザは FPGA の動作モードをクライアントかサーバーかで選択する必要があります。この動作モードは TOE10_SRV_REG レジスタでセットする値です。クライアント・モードの場合 FPGA は初期化シーケンス中に通信相手デバイスの MAC アドレスを取得するため ARP 要求を送信します。 サーバー・モードの場合 FPGA は初期化シーケンス中に通信相手デバイスからの ARP 要求を待ち ARP 応答を返送します。

2 枚の FPGA ボード間でテストを走らせる場合、各 FPGA の動作モードは異なるよう設定しなくてはなりません、つまり片方がクライアントでもう片方はサーバーとします。FPGA が PC と動作する場合 FPGA はクライアント・モードに設定することが推奨されます。その理由は PC にとって PC から ARP 要求を強制的に出力させるよりも ARP 要求を受け取ったときに ARP 応答を返送する方が容易なためです。

ファームウエアでは各動作モードそれぞれにデフォルトのパラメータがあります。 システムのブートアップ後の初期化シーケンスは以下となります。

1) CPU はユーザから動作モードを指定され、コンソール上にデフォルトのパラメータを表示します。
2) ユーザが'x'を入力しデフォルトのパラメータで初期化シーケンスを完了するか、または他のキーでパラメータを変更します。 パラメータを変更する場合、動作シーケンスは以下 3.2 章で説明する TOE10G-IP コアのリセットのシーケンスと同じです。
3) CPU は TOE10G-IP コアが初期化シーケンスを完了する(TOE10_CMD_REG[0]='0')まで待機します。
4) 5 つのメニュー(以下に詳細を説明します)があるメイン・メニューが表示されます。

3.1 パラメータ現在値の表示

本メニューは TOE10G-IP コアに設定したパラメータ例えば動作モード,自分側 MAC アドレス,相手/自分側 IP アドレス,相手/自分側ポート番号などの現在値を表示します。パラメータの表示シーケンスは以下となります。

1) ファームウエアの各変数よりネットワークのパラメータを読み出します
2) 各変数を表示します

3.2 TOE10G-IP コアのリセット

本メニューは IP アドレスや相手/自分側ポート番号などの TOE10G-IP コア・パラメータを変更する場合に使われます。 TOE10G-IP コアのレジスタをセットした後 CPU は IP コアをリセットし新しいパラメータで再初期化を行います。 CPU はコアのビジー・フラグをモニタし初期化が完了するのを待ちます。リセット・シーケンスを以下に説明します。

1) コンソール上に現在のパラメータを表示します
2) ユーザからのパラメータ入力を受け付け、入力値が有効な範囲内であるかをチェックします。 無効な値が入力された場合、その入力値は無視されます。
3) TOE10_RST_REG[0]='1'として IP コアを強制的にリセット状態とします
4) TOE10_SML_REG, TOE10_DIP_REG.など TOE10G-IP コアの全パラメータ・レジスタをセットします
5) TOE10_RESET_REG[0]='0'として IP コアのリセット状態を解除します
6) ユーザ回路へのリセット(USER_RST_REG[0]='1')を発行しユーザ・ロジックを初期状態にします
7) IP コアのビジー・フラグ(TOE10_CMD_REG[0])をモニタします、このフラグが'0'ネゲートされ IP コアの初期化プロセスが完了したことを確認します。

3.3 データ送信テスト

本テストでは 3 つの入力を必要とします、すなわち総送信サイズ、パケット･サイズ、コネクション・モード（クライアント動作の場合アクティブ･オープンでサーバー動作の場合パッシブ・オープン）です。 入力した値が無効な場合テスト動作はキャンセルされます。テスト実行中、32ビットのインクリメンタル・データが内部回路で生成され接続先の PC または FPGA へ送信されます。PC 側テスト・アプリケーションまたは FPGA 内部のベリファイ・モジュールにより受信データをベリファイできます。FPGA より接続先の PC または FPGA へ全データの送信が完了すると一連のテスト・シーケンスは終了します。このテストのシーケンスを以下に説明します。

1) 送信データ・サイズ、パケット・サイズ、コネクション・モードがユーザから入力され有効な値であるかチェッします。
2) UserReq レジスタに各入力値をセットします、すなわち転送サイズの設定(USER_TXLEN_REG)、テスト・パターンのクリア(USER_RST_REG)、データ・パターンの生成開始(USER_CMD_REG=0)を行います。 その後 UserReg 内のテスト・パターン発生回路は TOE10G-IP コアへデータを送信します。
3) システムに設定された現在のパラメータ値をもとに、PC 側で実行するアプリケーションの推奨パラメータ(引数)を表示します。
4) コネクション・モードに従ってコネクションをオープンします。
   a. アクティブ・オープンの場合 CPU は TOE10_CMD_REG=2 をセットし ConnOn が'1'となる(USER_CMD_REG[2])のをモニタします。
   b. パッシブ・オープンの場合 CPU は接続相手の PC 又は FPGA からのコネクションを ConnOn ステータス='1' (USER_CMD_REG[2])でオープンされるまで待機します。
5) TOE10G-IP コア内レジスタ(TOE10_PKL_REG)にパケット・サイズをセットし全転送サイズから繰返しループ回数を計算します。各ループでの最大転送サイズは 4G バイトです。各ループでの動作は以下となります。
   a. ループでの転送サイズをTOE10G-IPコア内レジスタ(TOE10_TDL_REG)にセットします。ここでセットする値は、最後のループでは残りの転送サイズでそれ以外のループでは 4G バイトです。
   b. TOE10G-IP 内レジスタ(TOE10_CMD_REG=0).に送信コマンドを指示します。
   c. TOE10_CMD_REG[0]='0'となり送信動作が完了するのを待ちます。 その待機中 CPU は 1 秒ごとにユーザ回路から現在の転送サイズを(USER_TXLEN_REG および USER_RXLEN_REG)にて読み出しコンソールに表示します。
6) TOE10G-IP 内レジスタ (TOE10_CMD_REG=3)でコネクションをクローズします。
7) パフォーマンスを計算しコンソールに結果を表示します。

3.4 データ受信テスト

ユーザは総受信サイズ、受信データ・ベリファイの有無、コネクション・モード（クライアント動作の場合アクティブ・オープンでサーバー動作の場合パッシブ・オープン）をセットします。 入力した値が無効な場合テスト動作はキャンセルされます。 ベリファイが指定されていた場合はテスト実行中、32 ビットのインクリメンタル・データが内部回路で生成され接続先の PC または FPGA からの受信データと比較されます。このテストのシーケンスを以下に説明します。

1) ユーザ入力の総受信サイズ、受信データのベリファイ有無、コネクション・モードを受け取ります。 全ての入力が有効であることを確認します。
2) UserReg レジスタをセットします、すなわちリセットによりテスト・パターンを初期化(USER_RST_REG)し、データ・ベリファイ・モード(USER_CMD_REG[1]='0' 又は '1')をセットします。
3) PC 側で実行するアプリケーションの推奨パラメータ(引数)を表示します、データ送信テストのステップ 3)と同じです。
4) コネクション・モードに従ってコネクションをオープンします、データ送信テストのステップ 4)と同じです。
5) 接続相手の PC 又は FPGA によりコネクションがクローズされることを Connon ステータス(USER_CMD_REG[2]='0')をモニタすることで待機します。 その待機中 CPU は 1 秒ごとにユーザ回路から現在の転送サイズを(USER_TXLEN_REG および USER_RXLEN_REG)にて読み出しコンソールに表示します。
6) これまでの受信サイズ(USER_RXLEN_REG)が、ユーザが入力した総受信サイズに到達したか、また、ベリファイ結果が合致していたか(USER_CMD_REG[1]) = '0')を確認します。エラー発生を検出した場合エラー・メッセージを表示します。
7) パフォーマンスを計算しコンソールに結果を表示します。

3.5 全二重通信テスト

このメニューでは FPGA と PC または FPGA 間で両方向同時かつ同じポート番号で送受信を実行する全二重通信をテストします。ユーザから 4 つのパラメータが入力されますがそれらは双方向での全転送サイズ、FPGA 送信ロジックでのパケット・サイズ、FPGA 受信ロジックでのベリファイ有無、コネクション・モード（クライアント動作でのアクティブ・オープン/クローズまたはサーバー動作でのパッシブ・オープン/クローズ）です。
PC と FPGA 間でのテストでは送信と受信ロジックでの転送サイズは最大サイズとなる 32G バイト固定ですが、そのサイズは PC 側の”tcp_client_txrx_10G” テスト・アプリケーションで設定されるサイズと同じで、またコネクション・モードもパッシブ・モード（サーバー動作）とする必要があります。
テストは PC と FPGA 間での転送の場合は、ユーザが PC 側にて(Ctrl+C キーで)テスト中断が指示されるまでか、あるいは 2 つの FPGA 間での転送の場合は NiosII コマンド・シェルでの中断要求が指示されるまで永久にテスト・ループを繰り返します。本テストの動作シーケンスを以下に説明します。

1) 総転送サイズ、送信パケット・サイズ、受信データ・ベリファイの有無、コネクション・モードをユーザが入力します、その入力値が有効であるかを確認します。
2) システムでの現在パラメータ値をもとに PC 側で実行するアプリケーションの推奨パラメータ(引数)を表示します。
3) UserReg レジスタをセットします、すなわち転送サイズ(USER_TXLEN_REG)をｾｯﾄしリセットによりテスト・パターンを初期化(USER_RST_REG)し、データ・ベリファイ・モード(USER_CMD_REG[1]=’1’ 又は ‘3’)をセットしパターン発生器を開始します。
4) コネクション・モードに従ってコネクションをオープンします、データ送信テストのステップ 4)と同じです。
5) TOE10G-IP コア内レジスタをｾｯﾄします、すなわちパケット・サイズ (TOE10_PKL_REG=ユーザ入力値) から各ループでの総転送サイズを計算します。1 ループでの最大サイズは 4G バイトです。その各ループでの動作は以下となります。
   a. 今回のループでの転送サイズを TOE10_TDL_REG にセットします。最終ループ以外での各ループの転送サイズは最高のパフォーマンスを得るためにパケット・サイズに合わせます。 最終ループの転送サイズは最後の残りデータ量です。
   b. 送信コマンドを TOE10G-IP コアにセット(TOE10_CMD_REG=0)します。
   c. TOE10_CMD_REG[0]=’0’をモニタすることで送信コマンドの完了を待ちます。 その完了待ちの間に CPU はユーザ・ロジックから現在の転送サイズを(USER_TXLEN_REG and USER_RXLEN_REG)から読み取り、1 秒ごとにコンソール上に表示します。
6) 設定されたコネクション・モードに従ってコネクションをクローズします。
   a. アクティブ・クローズの場合 CPU は総受信データ量が設定されていた値に到達するまで待機します。その後 USER_CMD_REG=3 をセットしコネクションのクローズを実行し(USER_CMD_REG[2]=’0’)によりクローズが完了するまで待ちます。
   b. パッシブ・クローズの場合 CPU は ConnOn 信号をモニタし(USER_CMD_REG[2]=’0’)でコネクションがクローズされるのを待ちます。
7) テスト結果やエラーをチェックします（データ受信テストのステップ 6 と同じです）。
8) パフォーマンス結果を計算しコンソール上に表示します。そしてステップ 3 に戻って動作ループを繰り返します。

4 テスト・アプリケーションの動作

4.1 半二重通信テスト向け“tcpdatatest”

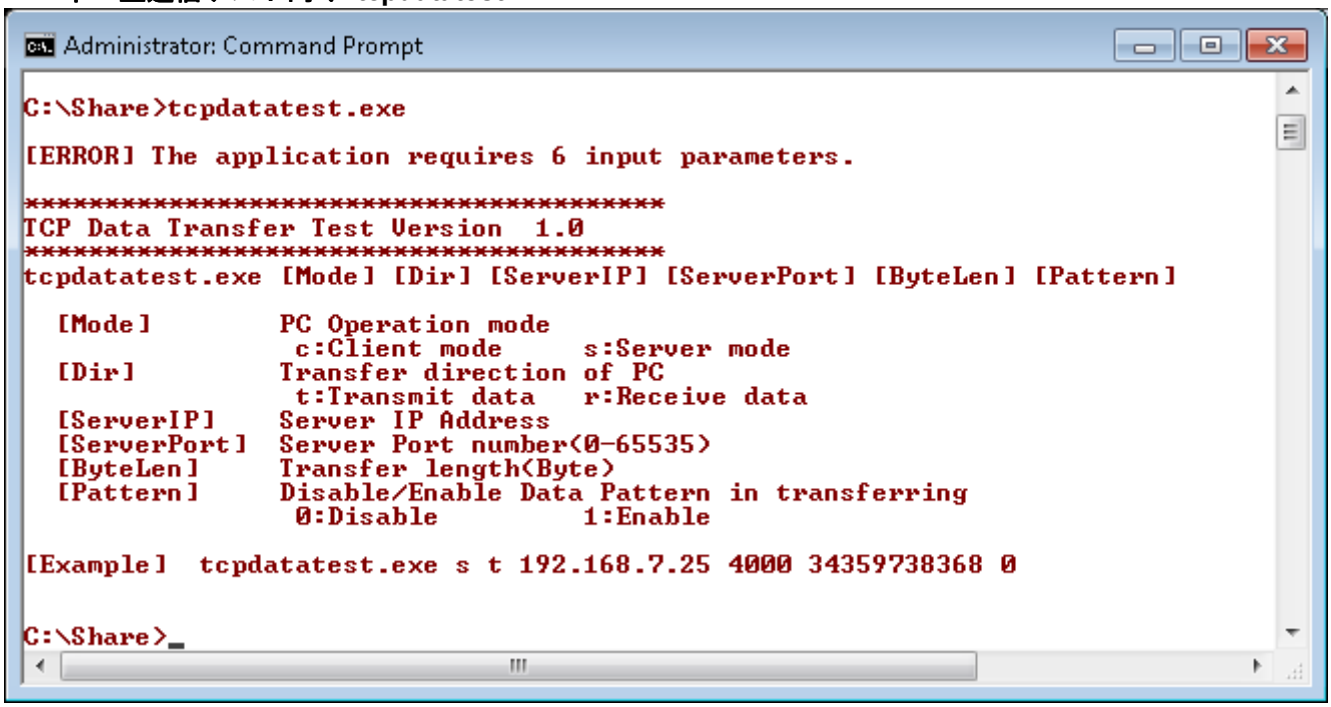


図 4-1: “tcpdatatest”アプリケーションの使い方

“tcpdatatest”は PC 側で動作する DOS アプリケーションであり、イーサネットを経由して TCP データをサーバーあるいはクライアント・モードで送受信します。ただし本デモでは PC 側はクライアント・モードでのみ実行します。ユーザは転送方向やモードなどのパラメータを引数で指定できます。本アプリケーションでは以下 6 つの引数入力を必要とします。

1) Mode: c – PC の動作モードで本デモは常に’c’を指定し PC がクライアント、FPGA がサーバーとします。
2) Dir: t – データ送信モード(PC から FPGA へのデータ送信)
   r – データ受信モード(PC が FPGA からのデータを受信)
3) ServerIP: PC がクライアント・モードで動作するときの FPGA 側の IP アドレス (初期値は 192.168.7.42)
4) ServerPort: PC がクライアント・モードで動作するときの FPGA 側のポート番号(初期値は 4000)
5) ByteLen: 総データ転送数をバイト単位で指定します。T ここで入力する値は送信時のみ使われ受信時は虫されます。 受信時はコネクションが破棄された時点でアプリケーション実行が終了します。 送信モードでの ByteLen 値は FPGA と接続したシリアル・コンソールでの設定値(データ受信テストでの設定値)と一致させる必要があります。
6) Pattern:
   0 – 送信モードではダミー・データを出力し、受信モードではデータ・ベリファイを行いません。
   1 – 送信モードではインクリメンタル・データを出力し、受信モードではデータ・ベリファイを実行します。

データ送信モード

送信モードでの本テスト・アプリケーション動作シーケンスを以下に説明します。

1) 送信データ・バッファとして 1M バイトのメモリ空間をアロケートします。
2) ソケットを生成し送信データ・バッファにプロパティを設定します。
3) ユーザから指定された IP アドレスとポート番号でサーバーに対して新たなコネクションを生成します。
4) テスト・パターンが指定された場合インクリメンタル・テスト・パターンをバッファに生成します。 ダミー・パターンが指定された場合はこの処理はスキップされます。
5) データを送信し残り転送サイズ値を減算します。
6) 1 秒ごとに総転送サイズを表示します。
7) 残り転送サイズがゼロになるまでステップ 4) – 6)を繰り返します。
8) ソケットをクローズしそう転送サイズとパフォーマンス結果を表示します。

データ受信モード

受信モードでの本テスト・アプリケーション動作シーケンスを以下に説明します。

1) 受信データ・バッファとして 1M バイトのメモリ空間をアロケートします。
2) ソケットを生成し受信データ・バッファにプロパティを設定します。
3) 送信モードのステップ 3)と同じでコネクションを生成します。
4) 受信データ・バッファからデータをリードし総受信サイズ値を増加します。
5) ベリファイ機能が指定されていた場合はインクリメンタル・テスト・パターンと受信データを比較しデータが合致しない場合はエラー・メッセージを表示します。 ベリファイ機能の指定がない場合このステップはスキップします。
6) 1 秒ごとに総受信サイズを表示します。
7) コネクションがクローズされるまでステップ 4) – 6)を繰り返します。
8) コネクションがクローズされたら総受信サイズとパフォーマンス結果を表示します。

4.2 全二重通信テスト向け“tcp_client_txrx_10G”

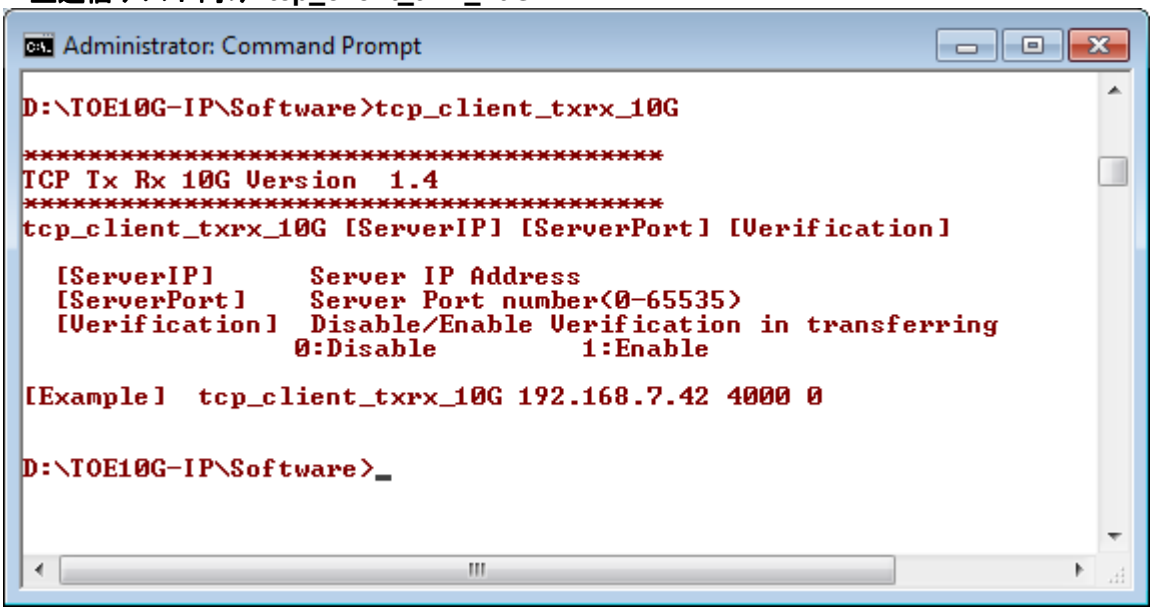


図 4-2: “tcp_client_txrx_10G”アプリケーションの使い方

“tcp_client_txrx_10G”は PC 側で動作する DOS アプリケーションであり、イーサネットを経由して TCP データ同一のポート番号を使って送信と受信を同時に実行します。。このアプリケーションは常にクライアント・モードで実行するため、ユーザは接続先サーバーの情報パラメータを入力する必要があります。
します。ユーザは図 4-2 に示すように以下 3 つの引数入力を必要とします。

1) ServerIP: FPGA(接続先サーバー)側の IP アドレス (初期値は 192.168.7.42)
2) ServerPort: FPGA(接続先サーバー)側のポート番号(初期値は 4000)
3) Verification:
   0 – 送信機能ではダミー・データを生成し受信機能ではベリファイ機能を停止します。このモードは全二重通信での最高パフォーマンスをチェックする場合に指定します。
   1 – 送信機能ではインクリメンタル・データを生成し受信機能ではベリファイ機能を有効にします。このモードは通信データ信頼性を確認する場合に指定します。（アプリケーションの送受信で CPU 処理を必要とするため、パフォーマンスは大きく低下します。）

本テスト・アプリケーションの動作シーケンスを以下に説明します。
(1) 送信および受信データ・バッファとして 60K バイトのメモリ空間をアロケートします。
(2) ソケットを生成しプロパティを設定します。
(3) ユーザから指定された IP アドレスとポート番号で新たなコネクションを生成します。
(4) テスト・パターンが指定された場合インクリメンタル・テスト・パターンを送信バッファに生成します。ダミー・パターンが指定された場合はこの処理はスキップされます。
(5) データを送信し残り転送サイズ値を減算します。
(6) 受信バッファからデータをリードし総受信サイズ値を増加します。
(7) ベリファイ機能が指定されていた場合はインクリメンタル・テスト・パターンと受信データを比較しデータが合致しない場合はエラー・メッセージを表示します。ベリファイ機能の指定がない場合このステップはスキップします。
(8) 1 秒ごとに総受信サイズを表示します。
(9) 総送信/受信データ量が 32G バイトに達するまでステップ 5) – 8)の動作ループを繰り返します。
(10) 総転送サイズとパフォーマンス結果を表示しソケットをクローズします。
(11) 1 ミリ秒待機しハードウエアが現在のテスト・ループを完全に終了するのを待機します。
(12) ステップ 3) – 11)の永久ループを繰り返します。ベリファイ結果がエラーとなった場合本アプリケーションは動作を中断します。

5 更新履歴

Revision	日付	内容
1.0	15-Mar-18	Initial version release
1.0J	2018/05/07	英語版初期バージョンを翻訳し日本語版を作成