XAPP1306 (v1.0) 2017 年 4 月 3 日

LightWeight IP スタックを使用した PS および PL ベース イーサネットの性能

著者: Bhargav Shah、Naveen Kumar Gaddipati、Akhilesh Mahajan、Srini Gaddam

概要

Lightweight IP (lwIP) は、エンベデッド システムに向けたオープン ソースの TCP/IP ネットワーキング スタックです。ザイリンクス ソフトウェア開発キット (SDK) には、Zynq® UltraScale+™ MPSoC に内蔵された ARM® 社のフラッグシップ製品である Cortex®-A53 64 ビット クワッドコア プロセッサまたは Cortex-R5 32 ビット デュアルコア プロセッサ上で動作するようにカスタマイズした lwIP ソフトウェアが含まれます。このアプリケーション ノートでは、lwIP ライブラリを使用して Zynq UltraScale+ MPSoC ベースのエンベデッド システムにネットワーキング機能を追加する方法について説明します。lwIP を使用してエコー サーバー、ウェブ サーバー、TFTP (Trivial File Transfer Protocol) サーバー、および受信/送信性能テスト アプリケーションを作成します。また、PS イーサネット、PL イーサネット (1G)、およびギガバイト イーサネット コントローラー (GEM) を使用した PS-PL イーサネットで lwIP を実行した場合のスループット値も示します。

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。デザイン ファイルの詳細は、「リファレンス デザイン」を参照してください。

はじめに

lwIP はエンベデッド システム用に設計されたオープン ソース ネットワーキング スタックで、BSD (Berkeley Software Distribution) スタイルのライセンスで提供されます。このアプリケーション ノートは、ザイリンクス SDK に付属する lwIP を使用してエンベデッド システムにネットワーキング機能を追加する方法を説明することを目的としています。特に、lwIP [参照 1] を使用してエコー サーバーやウェブ サーバーなどのアプリケーションを作成する一般的な方法について説明します。

Zynq UltraScale+ デバイスは、ARM 社のフラッグシップ製品である Cortex-A53 64 ビット クワッドコア プロセッサと Cortex-R5 デュアルコア リアルタイム プロセッサを含むプロセッシング システム (PS) とプログラマブル ロジック (PL) を 1 つに統合したデバイスです。

PL は、プログラマブル ロジック、コンフィギュレーション ロジック、および関連する内蔵機能で構成されます。PS は、ARM Cortex-A53 MPCore CPU、Cortex-R5 プロセッサ、オンチップ メモリ、外部メモリ インターフェイス、キャッシュ コヒーレント インターコネクト (CCI)、およびペリフェラル コネクティビティ インターフェイスで構成されます。PS には 4 つの GEM があります。外部 PHY との通信する際、RGMII インターフェイスには MIO ピンを介して接続し、その他のインターフェイスには EMIO インターフェイスを使用して接続します。

このアプリケーション ノートで説明するデザインでは、PS-GEM3 は RGMII (Reduced Gigabit Media Independent Interface) 経由で Texas Instruments (TI) PHY に接続されます。ZCU102 ボードではこれがデフォルトのセットアップです。このアプリケーション ノートでは、その他のイーサネット ポートのデザインについても説明します。このアプリケーション ノートで説明するデザインは次のとおりです。

- PS イーサネット (GEM3) を MIO インターフェイス経由で PS の 1G 物理インターフェイスに接続 (「MIO 経由で PS GEM を使用」参照)。
- PS イーサネット (GEM0) を EMIO インターフェイス経由で PL の 1000BASE-X 物理インターフェイスに接続 (「EMIO 経由で PS GEM を使用」参照)。
- PL にソフト ロジックとしてイーサネット (MAC) をインプリメントし、PL の 1000BASE-X 物理インターフェイスに接続 (「PL 1G イーサネットを使用」参照)。

注記: PS イーサネットには GEM1 または GEM2 も使用できます。ハードウェア デザインは、選択した GEM により異なります。

図 1 に、ZCU102 ボードへの各種イーサネット インプリメンテーションを示します。

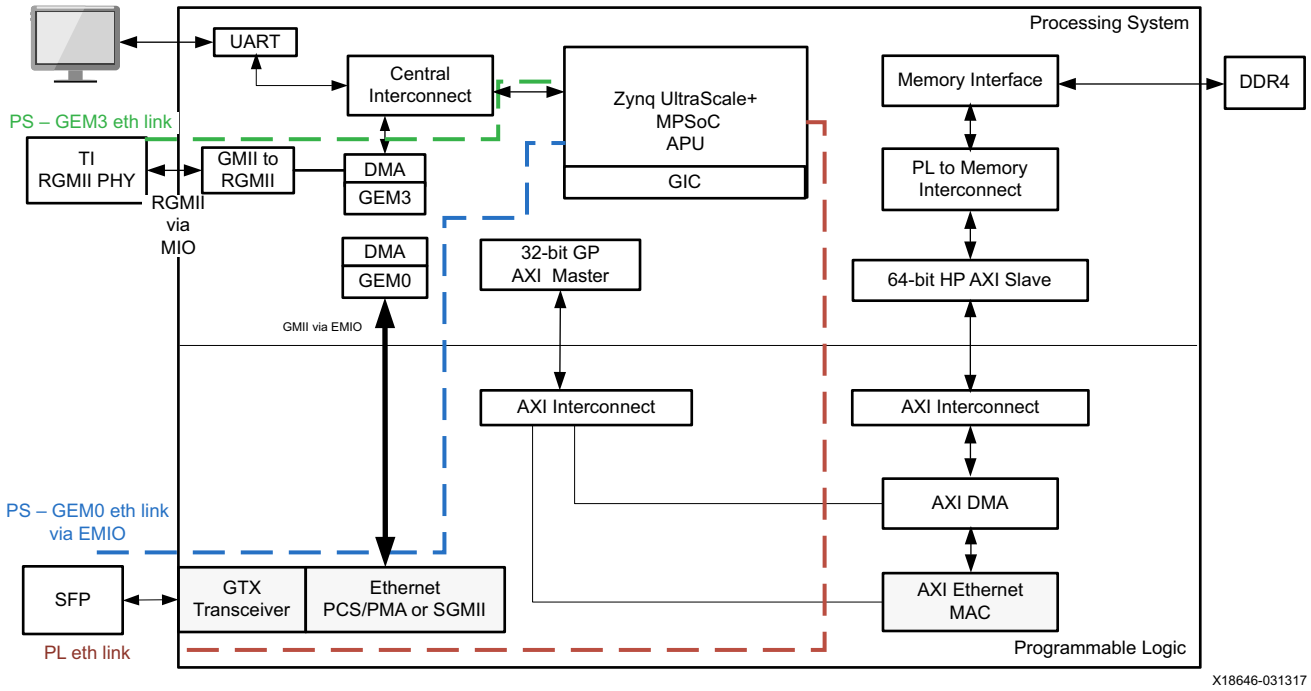

図 1: Zynq UltraScale+ MPSoC のイーサネット インターフェイス

注記: PS-GEM3 は、常に ZCU102 ボード上の Texas Instruments (TI) RGMII PHY に接続されます。1000BASE-X PHY と GTX トランシーバーは、AXI 1G/2.5G Ethernet サブシステム IP コア [参照 2] を使用する 1G PL イーサネット リンク用の AXI Ethernet コアに含まれます。PS-PL イーサネットは PS-GEM0 と 1G/2.5G Ethernet PCS/PMA or SGMII コア [参照 3] を使用します。

lwIP ソフトウェア アプリケーション

このリファレンス デザインには、次に示すソフトウェア アプリケーションが含まれます。これらのアプリケーションは RAW モードとソケット モードの両方で利用できます。

- エコー サーバー
- ウェブ サーバー
- TFTP サーバー
- TCP/UDP RX スループット テスト
- TCP/UDP TX スループット テスト

エコー サーバー

エコー サーバーは、ネットワーク経由で受信した入力をそのままエコー バックするシンプルなプログラムです。このアプリケーションは、lwIP アプリケーションの作成方法を初めて学ぶ場合の参考として最適です。

ソケット モードのエコー サーバーの構造は次のとおりです。

1. メイン スレッドは指定したエコー サーバー ポートを常時リッスンします。
2. 接続要求があると新規にエコー サービス スレッドを作成します。
3. その後、再びエコー ポートのリッスンを継続します。

```
while (1) {
     new_sd = lwip_accept(sock, (struct sockaddr *)&remote, &size);
     sys_thread_new(process_echo_request, (void*)new_sd, DEFAULT_THREAD_PRIO);
}
```

エコー サービス スレッドは入力として新規ソケット ディスクリプターを受信し、このディスクリプター上で受信データを読み出します。このスレッドが、入力の送信元に対する実際のエコー バックを実行します。

```
while (1) {
       /* read a max of RECV_BUF_SIZE bytes from socket */
       n = lwip_read(sd, recv_buf, RECV_BUF_SIZE));
       /* handle request */
       nwrote = lwip_write(sd, recv_buf, n));
}
```

注記: これらのコード スニペットはコードの主要な構造を示すことのみを目的としており、完全ではありません。

ソケット モードでは、ソケット上の読み出し/書き込みが完了するまで次の読み出し/書き込みをブロックするシンプルな API が提供されます。ただし、ソケット API でこの機能を実装するには、シンプルなマルチスレッド カーネルなど多くのコンポーネントを使用します。このため、この API はあらゆる動作でオーバーヘッドが大きく、低速です。

RAW API では、コールバック スタイルのインターフェイスがアプリケーションに提供されます。RAW API レジスタ コールバックを使用するアプリケーションでは、これらの関数は accept、read、または write などの重大なイベントで呼び出されます。RAW API ベースのエコー サーバーはシングル スレッドで、すべての処理がコールバック関数で実行されます。次に、メイン アプリケーション ループの構造を示します。

```
while (1) {
       if (TcpFastTmrFlag) {
       tcp_fasttmr();
       TcpFastTmrFlag = 0;
       }
       if (TcpSlowTmrFlag) {
       tcp_slowtmr();
       TcpSlowTmrFlag = 0;
       }
       xemacif_input(netif);
       transfer_data();
}
```

TCP TX 処理には TcpFastTmrFlag と TcpSlowTmrFlag が必要で、これらはタイマー ハンドラーでそれぞれ 250ms および 500ms ごとにセットされます。

このアプリケーション ループは、パケットを常時受信し (xemacif_input)、これを lwIP に渡します。このループに入る前に、エコー サーバーはいくつかのコールバックをセットアップします。

```
/* create new TCP PCB structure */
pcb = tcp_new();

/* bind to specified @port */
err = tcp_bind(pcb, IP_ADDR_ANY, port);

/* we do not need any arguments to callback functions */
tcp_arg(pcb, NULL);

/* listen for connections */
pcb = tcp_listen(pcb);

/* specify callback to use for incoming connections */
tcp_accept(pcb, accept_callback);
```

この呼び出しシーケンスにより TCP 接続が作成され、許可された接続に対するコールバックがセットアップされます。接続要求が許可されると、関数 accept_callback が非同期に呼び出されます。エコー サーバーはデータを受信したときだけ応答すればよいため、accept_callback 関数は次のように recv_callback をセットアップします。

```
/* set the receive callback for this connection */
tcp_recv(newpcb, recv_callback);
```

パケットを受信すると、関数 recv_callback が呼び出されます。この関数は、受信したデータを送信元へエコー バックします。

```
/* indicate that the packet has been received */
tcp_recved(tpcb, p->len);

/* echo back the payload */
err = tcp_write(tpcb, p->payload, p->len, 1);
```

RAW API の方がソケット API よりも複雑ですが、オーバーヘッドが小さいため高いスループットが得られます。

ウェブ サーバー

TCP ベース アプリケーションのリファレンスとして、シンプルなウェブ サーバー インプリメンテーションが付属しています。このウェブ サーバーは、HTTP 1.1 プロトコルのサブセットのみをインプリメントします。このようなウェブ サーバーは、ブラウザー経由でエンベデッド プラットフォームを制御または監視するのに使用できます。このウェブ サーバーでは次の機能を試すことができます。

- HTTP GET コマンドを使用してメモリ ファイル システム上のファイルにアクセス
- HTTP POST コマンドを使用して開発ボード上の LED を制御
- HTTP POST コマンドを使用して開発ボード上の DIP スイッチのステータスを取得

開発ボードのメモリには、ザイリンクス メモリ ファイル システム (xilmfs) を使用してファイルを格納します。HTTP GET コマンドでこれらのファイルにアクセスするには、ウェブ ブラウザーで開発ボードの IP アドレスを指定し、目的のファイルを要求します。

ボード上のコンポーネントを制御したり、そのステータスを監視したりするには、これらデバイスにマップする URL に対して POST コマンドを発行します。ウェブ サーバーは、認識可能な URL に対する POST コマンドを受信すると、要求された処理を実行するための関数を呼び出します。この関数の出力は、JSON (JavaScript Object Notation) フォーマットでウェブ ブラウザーに返されます。次に、ウェブ ブラウザーは受信したデータを解釈し、表示を更新します。

ウェブ サーバーの全体的な構造はエコー サーバーに似ています。1 つのメイン スレッドが HTTP ポート (80) をリッスンし、接続の着信を待ちます。接続が着信すると、その接続に対する要求を処理するためのスレッドを新規に作成します。

HTTP スレッドはまず要求を読み出し、その動作が GET なのか POST なのかを判定した後、それに応じた動作を実行します。GET 要求の場合、スレッドはメモリ ファイル システム内の特定のファイルを探します。このファイルが存在する場合は、要求発行元のウェブ ブラウザーにファイルを返します。ファイルが存在しない場合は HTTP 404 エラー コードをブラウザーに返します。

ソケット モードの HTTP スレッドの構造は次のとおりです。

```
/* read in the request */
if ((read_len = read(sd, recv_buf, RECV_BUF_SIZE)) < 0)
return;

/* respond to request */
generate_response(sd, recv_buf, read_len);
```

次に、generate_response 関数の擬似コードを示します。

```
/* generate and write out an appropriate response for the http request */
    int generate_response(int sd, char *http_req, int http_req_len)
{
        enum http_req_type request_type = decode_http_request(http_req, http_req_len);

        switch(request_type) {
            case HTTP_GET:
                    return do_http_get(sd, http_req, http_req_len);
            case HTTP_POST:
```

```
                return do_http_post(sd, http_req, http_req_len);
        default:
                return do_404(sd, http_req, http_req_len);
  }
}
```

RAW モードのウェブ サーバーは主にコールバック関数を使用してタスクを実行します。新しい接続が許可されると、accept_callback 関数が sent_callback および recv_callback 関数をセットアップします。これらの関数は、送信したデータに肯定応答 (ACK) が返されるか、データを受信したときに呼び出されます。

```
err_t accept_callback(void *arg, struct tcp_pcb *newpcb, err_t err)
{
     /* keep a count of connection # */
     tcp_arg(newpcb, (void*)palloc_arg());
     tcp_recv(newpcb, recv_callback);
     tcp_sent(newpcb, sent_callback);
     return ERR_OK;
}
```

ウェブ ページを要求する場合は、recv_callback 関数を呼び出します。この関数はソケット モードの関数と同様のタスクを実行し、要求をデコードして適切な応答を送信します。

```
/* acknowledge that we have read the payload */
tcp_recved(tpcb, p->len);

/* read and decipher the request */
/* this function takes care of generating a request, sending it,
* and closing the connection if all data has been sent. If
* not, then it sets up the appropriate arguments to the sent
* callback handler.
*/
generate_response(tpcb, p->payload, p->len);

/* free received packet */
pbuf_free(p);
```

データ送信は複雑です。ソケット モードでは、アプリケーションは lwip_write API を使用してデータを送信します。TCP 送信バッファーがフルの場合、この関数はデータ送信をブロックします。これに対し、RAW モードではアプリケーションはどれだけのデータを送信可能かを把握し、その量のデータしか送信しません。送信バッファーに空きができるまで、新しいデータは送信されません。送信したデータに対してレシーバー (クライアント コンピューター) から ACK が返されると、送信バッファーに空きができます。この時点で lwIP は sent_callback 関数を呼び出し、データが送信されたことで送信バッファーにデータ格納用の空きができたことを示します。sent_callback の構造は次のとおりです。

```
err_t sent_callback(void *arg, struct tcp_pcb *tpcb, u16_t len)
{
       int BUFSIZE = 1024, sndbuf, n;
       char buf[BUFSIZE];
       http_arg *a = (http_arg*)arg;

       /* if connection is closed, or there is no data to send */
       if (tpcb->state > ESTABLISHED) {
       return ERR_OK;
       }
       /* read more data out of the file and send it */
       sndbuf = tcp_sndbuf(tpcb);
       if (sndbuf < BUFSIZE)
       return ERR_OK;
       n = mfs_file_read(a->fd, buf, BUFSIZE);
       tcp_write(tpcb, buf, n, 1);

       /* update data structure indicating how many bytes
       * are left to be sent
       */
```

```
        a->fsize -= n;
        if (a->fsize == 0) {
              mfs_file_close(a->fd);
              a->fd = 0;
        }
        return ERR_OK;
}
```

sent_callback と recv_callback を呼び出す際の引数は、tcp_arg を使用して設定できます。ウェブ サーバーの場合、この引数は送信待ちのバイト数を格納するデータ構造、およびこのファイルの読み出しに使用できるファイル ディスクリプターを指し示します。

TFTP サーバー

TFTP は、UDP をベースにしたファイル送受信用プロトコルです。UDP ではパケットが確実に転送されることが保証されません。このため、TFTP は転送中のパケット紛失を防ぐプロトコルを規定しています。TFTP プロトコルの詳細は、「RFC 1350 - TFTP プロトコル」[参照 4] を参照してください。

ソケット モードの TFTP サーバーのアプリケーション構造はウェブ サーバーの場合とよく似ています。1 つのメイン スレッドが TFTP ポートをリッスンし、接続要求を受信するたびに新しい TFTP スレッドを作成します。この TFTP スレッドは TFTP プロトコルのサブセットをインプリメントし、読み出しまたは書き込みのいずれかの要求をサポートします。TFTP プロトコルでは一度に 1 つの TFTP データまたは ACK パケットしか転送されません。このため、TFTP プロトコルの実装は非常に簡単です。RAW モードの TFTP サーバーは非常に単純で、タイムアウト処理をしません。このため、パケット損失のないポイント ツー ポイントのイーサネット リンクとしてしか使用できません。デモ用と考えてください。

TFTP コードはウェブ サーバー コードと非常によく似ているため、このアプリケーション ノートでは説明を省略します。ソース コードを見るとわかるように、UDP を使用しているため若干の違いがあります。詳細は、「ウェブ サーバー」を参照してください。

TCP/UDP RX および TCP/UDP TX スループット テスト

TCP/UDP 送信/受信スループット テスト アプリケーションは、lwIP とザイリンクス MAC および PHY を組み合わせた場合に達成可能な最大 TCP 送信/受信スループットを調べるためのシンプルなアプリケーションです。これらのテストは、Iperf [参照 5] と呼ばれるオープン ソース ソフトウェアと通信します。

送信テストでは、lwIP を実行しているボードからホストまでの送信スループットを計測します。このテストでは、lwIP アプリケーションはホスト上で動作する Iperf サーバーに接続し、一定の決まったデータをホストに送信し続けます。ホスト上で動作する Iperf はデータ送信レートを判定し、ホスト ターミナルに出力します。

受信テストでは、ボードでの最大データ受信スループットを計測します。lwIP アプリケーションはサーバーとして動作します。このサーバーは、特定のポートで任意のホストからの接続を受け付けます。そして、このポートに送信されたデータを受信すると、受信データをそのまま破棄します。ホスト上でクライアント モードで動作する Iperf がこのサーバーに接続し、必要な時間だけデータを送信します。サーバーは、送信されたデータ量とその送信スループットを短い期間ごとに区切って計算し、この情報をコンソールに出力します。

MIO 経由で PS GEM を使用

このセクションでは、MIO インターフェイスを経由して PS イーサネット ブロック GEM3 を PS PHY で使用する方法について説明します。

ハードウェア デザイン

PS イーサネット MAC (GEM3) は、RGMII インターフェイスを使用して MIO ピン経由でオンボードの TI PHY に接続します。GEM3 ブロックはハードウェア システムの生成中に有効になります。GEM3 から TI PHY へのリンクを、図 1 では「PS-GEM3 eth link」と記載しています。

基準クロックの生成

このデザインは、Zynq UltraScale+ MPSoC の GTX トランシーバー X0Y4 を ZCU102 ボードの SFP ケージに接続して使用します。GTX トランシーバーの基準クロック (125MHz の差動クロック) は、ZCU102 ボードの Si570 ジッター減衰器から生成されます。クロック分周値は、Si570 プログラマブル オシレーターから 125 MHz が生成されるように調整します。Si570 を I2C インターフェイス経由でプログラムして必要なクロック値である 125 MHz を生成します。Si570 の詳細は、Si570 のデータシート [参照 6] を参照してください。

ソフトウェア デザイン

このデザインは ZCU102 のすべての GEM に対して共通の lwIP ライブラリ (A53 または R5 用) を使用します。lwIP ライブラリは、PS の GEM コントローラーに接続されたダイレクト メモリ アクセス (DMA) コントローラーを使用します。このドライバーは、DMA ディスクリプター リングのセットアップ、割り当て、再利用などの複数の機能を担います。割り込みステータスには DMA イベントも暗黙的に反映されるため、割り込み処理は PS GEM イベントに対してのみ行われます。

lwIP ライブラリ

このデザインには、Cortex-A53 または Cortex-R5 プロセッサ用の lwIP ライブラリが提供されています。図 2 に、PS イーサネット インターフェイスのソフトウェア アーキテクチャを示します。

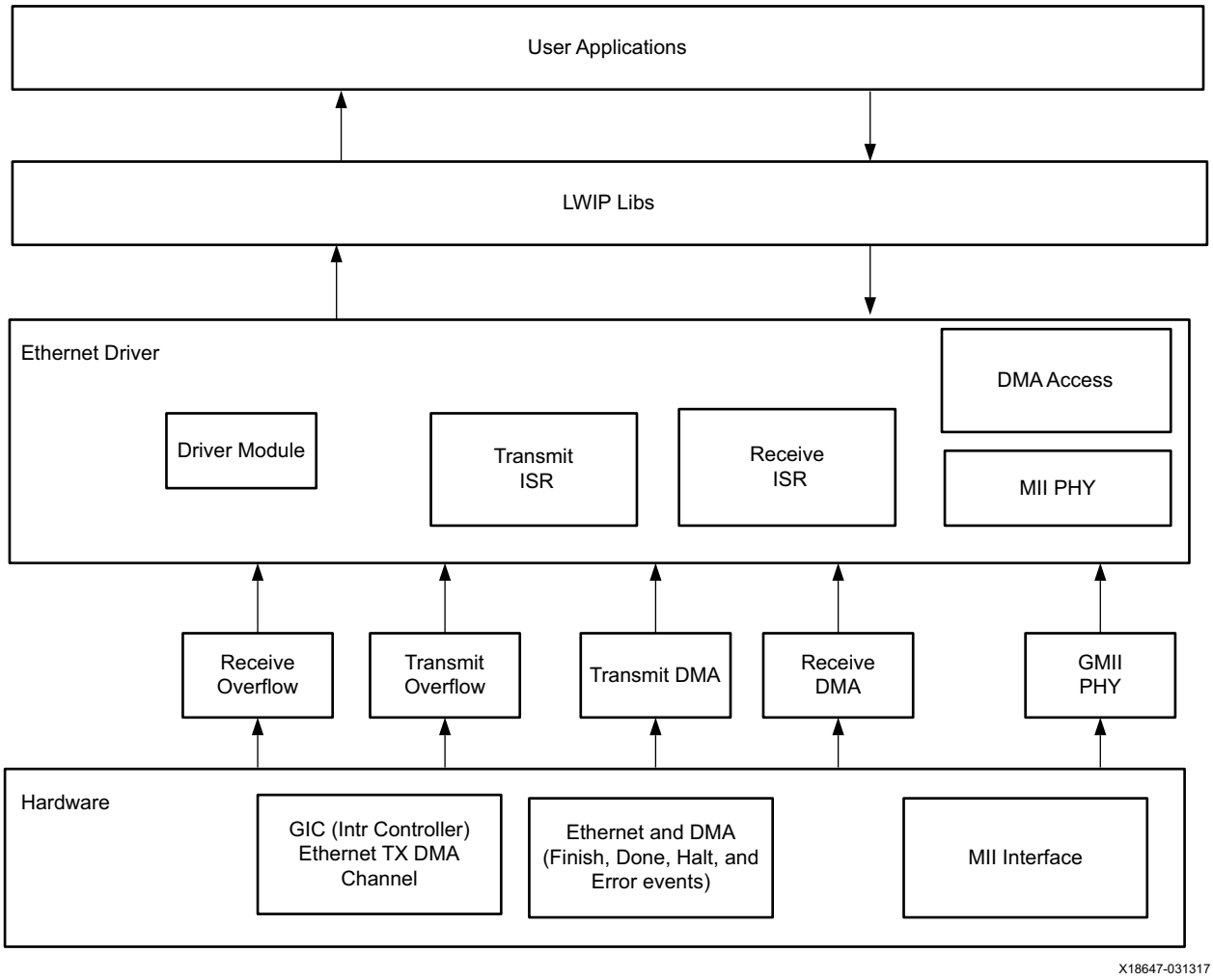

図 2: lwIP ライブラリのレイヤー

リファレンス デザインの実行

このセクションでは、PS イーサネットの実行方法について説明します。

ホスト ネットワークの設定

1. 使用するボードをイーサネット ケーブルでホスト コンピューターのイーサネット ポートに接続します。
2. ホスト コンピューターのイーサネット インターフェイスに IP アドレスを割り当てます。ボードには、ソフトウェア アプリケーションによってデフォルトの IP アドレス 10.10.70.3 が割り当てられます。このアドレスは、各アプリケーションの `main.c` ファイルで変更できます。ホストには、ボードと同じサブネット マスクの IP アドレス (ボードの IP アドレスがデフォルトの場合、10.10.70.9 など) を割り当てます。コンパイルの詳細は、Wiki ページ「Zynq MPSoC の PS および PL ベース イーサネット」[参照 7] を参照してください。

EMIO 経由で PS GEM を使用

このセクションでは、EMIO インターフェイスを経由して PS イーサネット ブロック GEM0 を PL PHY で使用する方法について説明します。PS イーサネット ブロックは、EMIO、GMII、および Management Data Input/Output (MDIO) インターフェイスを介して、PL からアクセス可能になります。イーサネット物理媒体には、1G Ethernet PCS/PMA or SGMII コアを 1000BASE-X モードで使用します。ZCU102 ボードの SFP (Small Form-Factor Pluggable) ケージには、高速シリアル トランシーバーを使用してアクセスします。SFP ケージは SFP-RJ45 コンバーター モジュールを介して標準イーサネット LAN に接続します。SFP を有効にするには、図 5 に示すようにジャンパー J16 をショートします。

ハードウェア デザイン

図 3 に、PS-PL イーサネット デザインを示します。GMII インターフェイスは EMIO ピンを介して PHY と PS GEM を接続します。GEM0 ブロックは Vivado® ツールでハードウェア システムを生成中に有効になります。1G/2.5G Ethernet PCS/PMA or SGMII コアの PHY アドレス ポートには 1 ～ 31 の固定値を割り当てることができます。詳細は、『1G/2.5G Ethernet PCS/PMA or SGMII v16.0 LogiCORE IP 製品ガイド』(PG047) [参照 3] を参照してください。

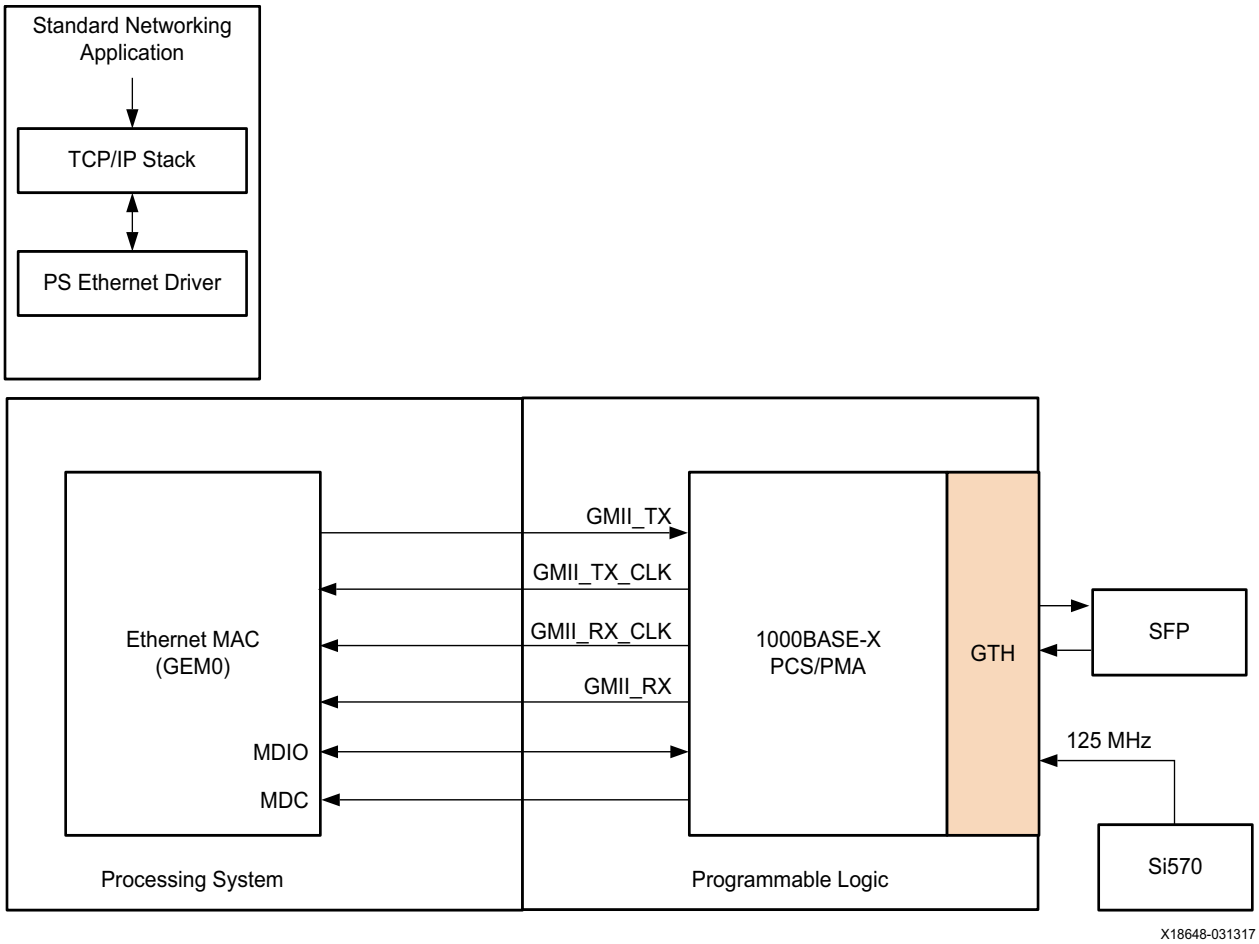

図 3: PS-PL イーサネット デザイン

基準クロックの生成

Zynq UltraScale+ MPSoC の GTX トランシーバー X0Y4 を 1000BASE-X トランシーバー用に ZCU102 ボードの SFP ケージに接続します。GTX トランシーバーの基準クロック (125 MHz の差動クロック) は、ZCU102 ボードの Si570 ジッター減衰器から生成されます。クロック分周値は、Si570 プログラマブル オシレーターから 125 MHz が生成されるように調整します。Si570 を I2C インターフェイス経由でプログラムして必要なクロック値を生成します。Si570 の詳細は、Si570 のデータシート [参照 6] を参照してください。

EMIO インターフェイス経由で GEM0 を利用するには、特定のレジスタをプログラムする必要があります。これは、Zynq UltraScale+ MPSoC の FSBL (第 1 段階ブートローダー) が使用する PS コンフィギュレーション データに含まれます。

受信クロック、データ、および制御信号のソースとして EMIO を選択するには、SLCR.GEM0_CLK_CTRL[SRCSEL] ビットを `3'b1xx` に設定します (x はドントケアで、1 または 0)。

ソフトウェア デザイン

このデザインは、ZCU102 のすべての GEM に共通の lwIP ライブラリ コードを使用します。lwIP ライブラリは、PS の GEM コントローラーに接続された DMA コントローラーを使用します。このドライバーは、DMA ディスクリプター リングのセットアップ、割り当て、再利用などの複数の機能を担います。割り込みステータスには DMA イベントも暗黙的に反映されるため、割り込み処理は PS GEM イベントに対してのみ行われます。

lwIP ライブラリ

このデザインには lwIP ライブラリが提供されています。図 2 に、PS イーサネット インターフェイスのソフトウェア アーキテクチャを示します。

リファレンス デザインの実行

このセクションでは、PS イーサネットの実行方法について説明します。

ホスト ネットワークの設定

1. SFP-to-RJ-45 コンバーターを SFP 0 に接続します。
2. SFP モジュールをイーサネット ケーブルでホスト コンピューターのイーサネット ポートに接続します。
3. ホスト コンピューターのイーサネット インターフェイスに IP アドレスを割り当てます。ボードには、ソフトウェア アプリケーションによってデフォルトの IP アドレス 10.10.70.3 が割り当てられます。このアドレスは、各アプリケーションの `main.c` ファイルで変更できます。ホストには、ボードと同じサブネット マスクの IP アドレス (ボードの IP アドレスがデフォルトの場合、10.10.70.9 など) を割り当てます。コンパイルの詳細は、Wiki ページ「Zynq MPSoC の PS および PL ベース イーサネット」[参照 7] を参照してください。

PL 1G イーサネットを使用

このセクションでは、PL にインプリメントしたイーサネットについて説明します。このデザインは、AXI 1G/2.5G Ethernet サブシステム、AXI DMA、AXI Interconnect の IP コアで構成されます。AXI 1G/2.5G Ethernet サブシステム IP コアは Tri-Mode Ethernet MAC (TEMAC) コアと 1G/2.5G Ethernet PCS/PMA or SGMII コアで構成されます。このデザインでは、PS-DDR メモリへの高速アクセスに HP (High Performance) ポートを使用します。HP ポートをほかのペリフェラルが使用している場合は、汎用スレーブ ポートを使用することもできます。

ハードウェア デザイン

図 4 に、PL にインプリメントしたイーサネットを示します。PL と PS DDR4 メモリ間の高速データ転送には HP ポートを使用しています。このポートから、AXI Interconnect を経由して AXI DMA スキャッター ギャザー S2MM (Stream to Memory Mapped) および MM2S (Memory Mapped to Stream) インターフェイスに接続します。AXI Interconnect は、64 ビット HP ポートを AXI DMA の 32 ビット インターフェイスに接続するためのデータ幅変換も実行します。AXI DMA では、S2MM パスと MM2S パスに対してスキャッター ギャザー オプションおよびデータ リアライメント エンジンの両方が有効です。

AXI DMA のストリーミング インターフェイスは、AXI Ethernet サブシステムに接続されます。AXI Ethernet サブシステムでは、フル チェックサム オフロード (CSO) を有効にし、FIFO 深さを 32K としてジャンボ フレーム転送をサポートします。

AXI Ethernet コアはイーサネット MAC をインプリメントし、1000BASE-X および SGMII PHY インターフェイスをサポートします。このコアは、GTX トランシーバーを介して 1000BASE-X/SGMII インターフェイス経由で SFP に接続します。

制御インターフェイス用に、PS で汎用 (GP) AXI マスター ポートを有効にしています。このポートは、AXI DMA コアと AXI Ethernet コアに接続されます。

1000BASE-X PHY レジスタには、AXI Ethernet コアによって提供される MDIO インターフェイスを介してアクセスします。AXI DMA および AXI Ethernet コアの割り込みポートは、PS 内の汎用割り込みコントローラー (GIC) に接続されます。

各 IP コアの詳細は、『LogiCORE IP AXI DMA 製品ガイド』(PG021) [参照 8] および『AXI 1G/2.5G Ethernet サブシステム v7.0 製品ガイド』(PG138) [参照 2] を参照してください。

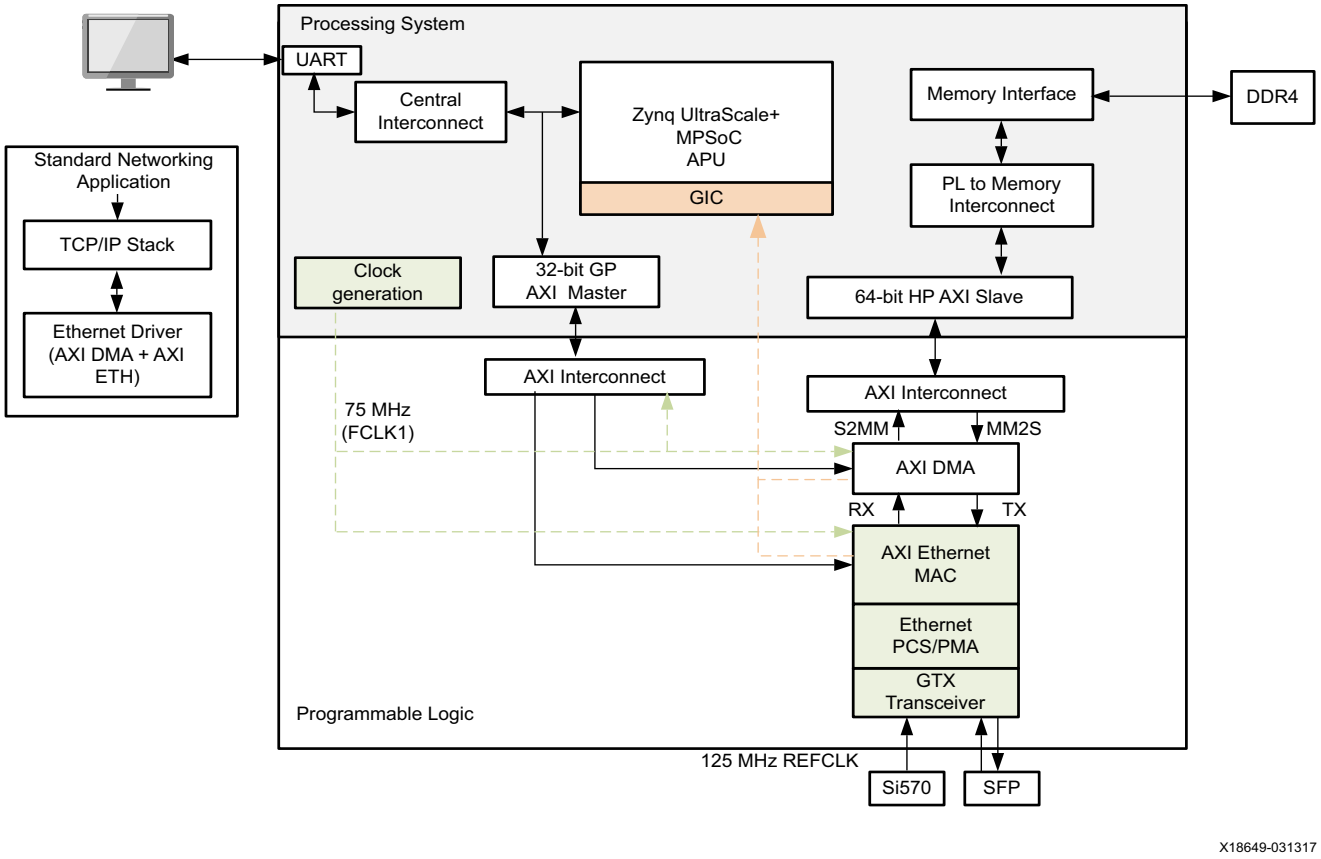

図 4: 1000BASE-X イーサネット デザイン

基準クロックの生成

Zynq UltraScale+ MPSoC の GTX トランシーバー X0Y4 を 1000BASE-X トランシーバー用に ZCU102 ボードの SFP ケージに接続します。GTX トランシーバーの基準クロック (125 MHz の差動クロック) は、ZCU102 ボードの Si570 ジッター減衰器から生成されます。クロック分周値は、Si570 プログラマブル オシレーターから 125 MHz が生成されるように調整します。Si570 を I2C インターフェイス経由でプログラムして必要なクロック値を生成します。Si570 の詳細は、Si570 のデータシート [参照 6] を参照してください。

ソフトウェア デザイン

このセクションでは、デザインのソフトウェアについて説明します。lwIP ライブラリには、次の機能があります。

- PL イーサネット MAC アクセス
- AXI DMA 転送
- 1000BASE-X インターフェイスの物理媒体の初期化

lwIP ライブラリ

このデザインには lwIP ライブラリが提供されています。図 2 に、PL イーサネット インターフェイスのソフトウェア アーキテクチャを示します。

PL イーサネット システムの実行

このセクションでは、PL イーサネット システムの実行方法について説明します。

ホスト ネットワークの設定

1. 使用するボードをイーサネット ケーブルでホスト コンピューターのイーサネット ポートに接続します。
2. ホスト コンピューターのイーサネット インターフェイスに IP アドレスを割り当てます。ボードには、ソフトウェア アプリケーションによってデフォルトの IP アドレス 10.10.70.3 が割り当てられます。このアドレスは、各アプリケーションの `main.c` ファイルで変更できます。ホストには、ボードと同じサブネット マスクの IP アドレス (ボードの IP アドレスがデフォルトの場合、10.10.70.9 など) を割り当てます。コンパイルの詳細は、Wiki ページ「Zynq MPSoC の PS および PL ベース イーサネット」[参照 7] を参照してください。

動作中のソフトウェアとの通信

ソケット モードと RAW モードのアプリケーションでは、エコー サーバー、ウェブ サーバー、TFTP サーバー、および RX/TX スループット テスト サンプルが 1 つの実行ファイルにまとめられています。次に示す出力は、Cortex-A53 プロセッサの場合も Cortex-R5 プロセッサの場合も同じです。

アプリケーションからの出力

実行ファイルの動作が完了すると、シリアル ポートに次のような表示が出力されます。

```
  -----lwIP RAW Mode Demo Application ------
         Board IP: 10.10.70.3
         Netmask: 255.255.255.0
         Gateway: 10.10.70.1
         auto-negotiated link speed: 1000

        Server   Port   Connect With..
--------------------   ------  --------------------
          echo server   7     $ telnet <board_ip> 7
        rxperf server   5001  $ iperf -c <board ip> -i 5 -t 100
        txperf client   N/A   $ iperf -s -i 5 -t 100 (on host with IP 10.10.70.9)
          tftp server   69    $ tftp -i 192.168.1.10 PUT <source-file>
          http server   80    Point your web browser to http://10.10.70.10
```

ソケット モード アプリケーションの場合、1 行目にソケット モードのデモ アプリケーションであることが表示されますが、それ以外は同じ内容が出力されます。この時点で、ボード上で動作するアプリケーションにホスト マシンから通信できます。

TCP/UDP 受信スループット テストとの通信

受信スループットを計測するには、適切なオプションを指定して iperf -c コマンドを発行し、iperf クライアントから受信 iperf アプリケーションに接続します。

TCP のサンプル セッションは次のとおりです。

```
[root@localhost xhdpssa]# iperf -c 10.10.70.3 -i 5 -t 50 -w 64k
------------------------------------------------------------
Client connecting to 10.10.70.3, TCP port 5001
TCP window size: 128 KByte (WARNING: requested 64.0 KByte)
------------------------------------------------------------
[ 3] local 10.10.70.3  port 43822 connected with 10.10.70.9 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0- 5.0 sec 560 MBytes 939 Mbits/sec
[ 3] 5.0-10.0 sec 562 MBytes 943 Mbits/sec
```

```
[ 3] 10.0-15.0 sec 562 MBytes 943 Mbits/sec
[ 3] 15.0-20.0 sec 562 MBytes 943 Mbits/sec
[ 3] 20.0-25.0 sec 562 MBytes 943 Mbits/sec
[ 3] 25.0-30.0 sec 562 MBytes 943 Mbits/sec
[ 3] 30.0-35.0 sec 562 MBytes 943 Mbits/sec
[ 3] 35.0-40.0 sec 562 MBytes 943 Mbits/sec
[ 3] 40.0-45.0 sec 562 MBytes 943 Mbits/sec
[ 3] 45.0-50.0 sec 562 MBytes 943 Mbits/sec
[ 3] 0.0-50.0 sec 5.49 GBytes 942 Mbits/sec
```

UDP のサンプル セッションは次のとおりです。

```
[root@localhost xhdpssa]# iperf -c 10.10.70.3 -i 5 -t 50 -u -b 1G
------------------------------------------------------------
Client connecting to 10.10.70.9, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 3] local 10.10.70.3  port 43822 connected with 10.10.70.9 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0-5.0 sec 560 MBytes 955 Mbits/sec
[ 3] 5.0-10.0 sec 562 MBytes 951 Mbits/sec
[ 3] 10.0-15.0 sec 562 MBytes 950 Mbits/sec
[ 3] 15.0-20.0 sec 562 MBytes 952 Mbits/sec
[ 3] 20.0-25.0 sec 562 MBytes 949 Mbits/sec
[ 3] 25.0-30.0 sec 562 MBytes 953 Mbits/sec
[ 3] 30.0-35.0 sec 562 MBytes 952 Mbits/sec
[ 3] 35.0-40.0 sec 562 MBytes 951 Mbits/sec
[ 3] 40.0-45.0 sec 562 MBytes 952 Mbits/sec
[ 3] 45.0-50.0 sec 562 MBytes 954 Mbits/sec
[ 3] 0.0-50.0 sec 5.49 GBytes 952 Mbits/sec
```

注記: 受信スループットを最大にするには、TCP ウィンドウ サイズを 64KB ではなく 8KB とします。

TCP/UDP 送信スループット テストとの通信

送信スループットを計測するには、ホスト上で iperf サーバーを開始した後、ボード上で実行ファイルを起動します。実行ファイルを起動すると、ホスト 10.10.70.9 のサーバーへの接続を試みます。このアドレスは、`txperf.c` ファイルで変更できます。

TCP のサンプル セッション ホスト PC は次のとおりです。

```
[root@localhost xhdpssa]# iperf -s -i 5 -w 64k
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 128 KByte (WARNING: requested 64.0 KByte)
------------------------------------------------------------
[ 4] local 10.10.70.3 port 5001 connected with 10.10.70.9 port 49153
[ ID] Interval Transfer Bandwidth
[ 4] 0.0- 5.0 sec 563 MBytes 944 Mbits/sec
[ 4] 5.0-10.0 sec 566 MBytes 949 Mbits/sec
[ 4] 10.0-15.0 sec 566 MBytes 949 Mbits/sec
[ 4] 15.0-20.0 sec 566 MBytes 949 Mbits/sec
[ 4] 20.0-25.0 sec 566 MBytes 949 Mbits/sec
[ 4] 25.0-30.0 sec 566 MBytes 949 Mbits/sec
[ 4] 30.0-35.0 sec 566 MBytes 949 Mbits/sec
[ 4] 35.0-40.0 sec 566 MBytes 949 Mbits/sec
[ 4] 40.0-45.0 sec 566 MBytes 949 Mbits/sec
[ 4] 45.0-50.0 sec 566 MBytes 949 Mbits/sec
```

UDP のサンプル セッション ホスト PC は次のとおりです。

```
[root@localhost xhdpssa]# iperf -s -i 10 5  -t 100 -u
iperf: ignoring extra argument -- 5
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.5 port 5001 connected with 192.168.1.4 port 42788
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec   958 MBytes   804 Mbits/sec   0.001 ms   58/683527 (0.0085%)
[  3] 10.0-20.0 sec   958 MBytes   804 Mbits/sec   0.001 ms  107/683514 (0.016%)
[  3] 20.0-30.0 sec   958 MBytes   804 Mbits/sec   0.001 ms  231/683499 (0.034%)
[  3] 30.0-40.0 sec   958 MBytes   804 Mbits/sec   0.001 ms  168/683504 (0.025%)
[  3] 40.0-50.0 sec   958 MBytes   803 Mbits/sec   0.001 ms  153/683372 (0.022%)
[  3] 50.0-60.0 sec   958 MBytes   804 Mbits/sec   0.001 ms  144/683404 (0.021%)
[  3] 60.0-70.0 sec   958 MBytes   804 Mbits/sec   0.000 ms   84/683499 (0.012%)
[  3] 70.0-80.0 sec   958 MBytes   804 Mbits/sec   0.002 ms  153/683502 (0.022%)
[  3] 80.0-90.0 sec   958 MBytes   804 Mbits/sec   0.001 ms  155/683512 (0.023%)
[  3] 90.0-100.0 sec   958 MBytes   804 Mbits/sec   0.001 ms  198/683491 (0.029%)
[  3]  0.0-100.0 sec  9.36 GBytes   804 Mbits/sec   0.001 ms 1450/6835471 (0.021%)
```

サーバーを停止するには、Ctrl + C キーを 2 回押します。

必要なハードウェアおよびソフトウェア

このアプリケーション ノートで説明したデザインのテストには、次のハードウェアとソフトウェアが必要です。

- Linux OS が動作する標準 PC
- 1000 Mb/s 対応イーサネット ポート
- ホスト側に 1G 用 SFP モジュール
- iPerf ツール [参照 5]
- テスト用 SFP-RJ45 アダプター モジュールを搭載した Zynq UltraScale+ MPSoC ZCU102 ボード [参照 9]
- Vivado ツール 2016.4 (IP インテグレーター デザイン) [参照 10]
- PetaLinux 2016.4 XSDK [参照 11]

図 5 に、1G インターフェイスを使用する場合のボードのセットアップを示します。ジャンパー J16 は、SFP を介した伝送を無効にするように設定してください。このデザインは、Cisco GLC-T 1000BASE-X Gigabit Ethernet to Optical SFP モジュールを使用してテストしました。

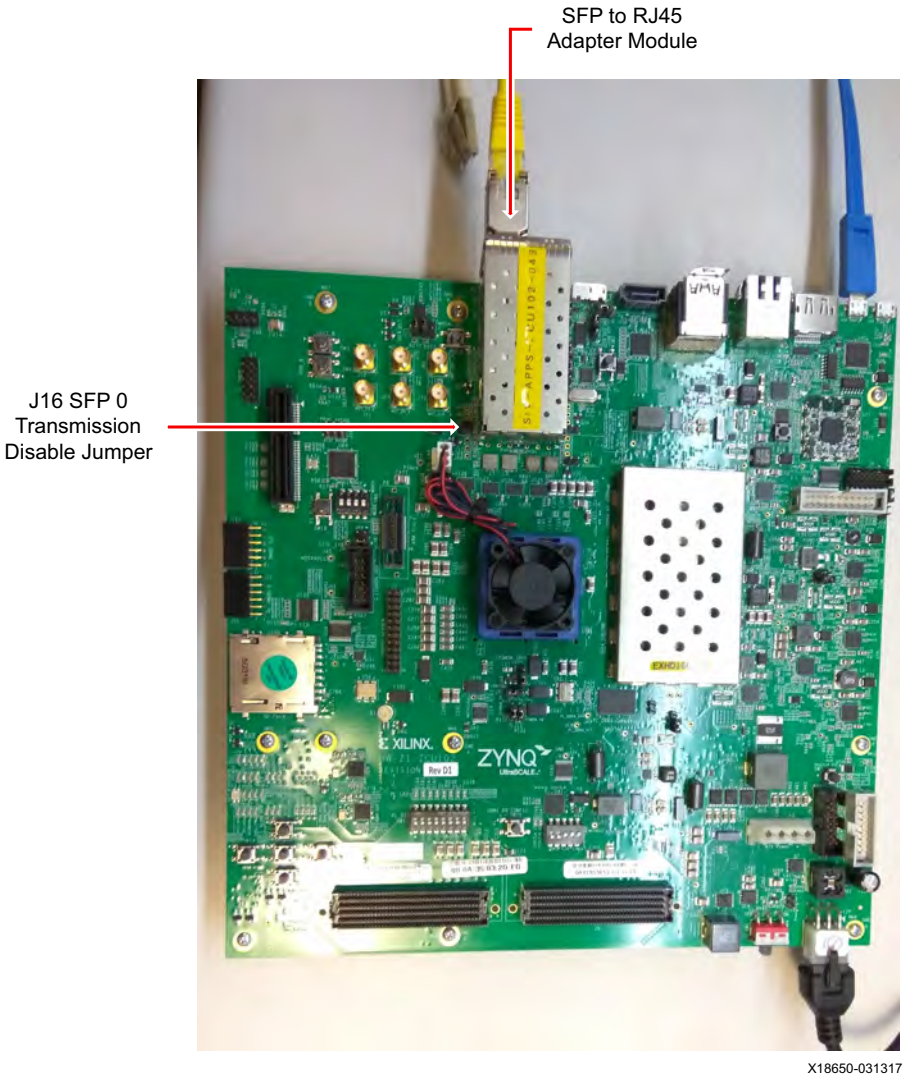

図 5: 1G イーサネットのハードウェア セットアップ

まとめ

このアプリケーション ノートでは、EMIO/MIO を介した PS イーサネット、および PL の 1G イーサネットをインプリメントしたデザインにより、複数のイーサネット リンクをサポートする方法について説明しました。このアプリケーション ノートで説明したデザインの性能ベンチマークの結果は、Wiki ページ「Zynq MPSoC の PS および PL ベース イーサネット」[参照 7] を参照してください。

リファレンス デザイン

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。

表 1 に、リファレンス デザインの詳細を示します。

表 1: リファレンス デザインの詳細

パラメーター	説明
全般	
開発者	Bhargav Shah、Naveen Kumar Gaddipati、Akhilesh Mahajan、Srini Gaddam
ターゲット デバイス	Zynq UltraScale+ デバイス
ソース コードの提供	あり
ソース コードの形式	Verilog、C
既存のザイリンクス アプリケーション ノート/リファレンス デザイン、またはサードパーティからデザインへのコード/IP の使用	あり
シミュレーション	
論理シミュレーションの実施	なし
タイミング シミュレーションの実施	なし
論理シミュレーションおよびタイミング シミュレーションでのテストベンチの利用	なし
テストベンチの形式	N/A
使用したシミュレータ/バージョン	N/A
SPICE/IBIS シミュレーションの実施	N/A
インプリメンテーション	
使用した合成ツール/バージョン	Vivado 2016.4
使用したインプリメンテーション ツール/バージョン	Vivado 2016.4
スタティック タイミング解析の実施	あり
ハードウェア検証	
ハードウェア検証の実施	あり
使用したハードウェア プラットフォーム	ZCU102 評価ボード

参考資料

注記: 日本語版のバージョンは、英語版より古い場合があります。

1. lwIP (https://en.wikipedia.org/wiki/LwIP)
2. 『AXI 1G/2.5G Ethernet サブシステム v7.0 製品ガイド』(PG138)
3. 『1G/2.5G Ethernet PCS/PMA or SGMII v16.0 LogiCORE IP 製品ガイド』(PG047)
4. RFC 1350 - TFTP プロトコル (http://www.faqs.org/rfcs/rfc1350.html)
5. Iperf (http://sourceforge.net/projects/iperf/)
6. Si570 データシート (www.silabs.com/Support%20Documents/TechnicalDocs/Si570.pdf)
7. Wiki ページ「Zynq MPSoC の PS および PL ベース イーサネット」
8. 『LogiCORE IP AXI DMA 製品ガイド』(PG021: 英語版、日本語版)
9. 『ZCU102 評価ボード ユーザー ガイド』(UG1182)
10. ザイリンクス Vivado Design Suite
11. PetaLinux
12. Netperf (www.netperf.org)
13. 『Zynq UltraScale+ MPSoC テクニカル リファレンス マニュアル』(UG1085: 英語版、日本語版)
14. 『UltraScale アーキテクチャ GTH トランシーバー ユーザー ガイド』(UG576: 英語版、日本語版)

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2017 年 4 月 3 日	1.0	初版

お読みください: 重要な法的通知

自動車用のアプリケーションの免責条項

ザイリンクスの製品は、フェイルセーフとして設計されたり意図されてはおらず、また、フェイルセーフの動作を要求するアプリケーション (具体的には、(I) エアバッグの展開、(II) 車のコントロール (フェイルセーフまたは余剰性の機能 (余剰性を実行するためのザイリンクスの装置にソフトウェアを使用することは含まれません) および操作者がミスをした際の警告信号がある場合を除きます)、(III) 死亡や身体傷害を導く使用、に関するアプリケーション) を使用するために設計されたり意図されたりもしていません。顧客は、そのようなアプリケーションにザイリンクスの製品を使用する場合のリスクと責任を単独で負います。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。