1R－8

ハードウェアによるTCP実装方式に関する一考察

田上 敦士† 佐藤 友実‡ 井戸上 彰† 加藤 聰彦†
†KDD研究所 ‡電気通信大学

1. はじめに

近年、インターネットの広帯域化が目覚しく、それに伴い端末におけるプロトコル処理の高速化が強く要求されている。これに対し、ソフトウェアによるプロトコル処理では限界があるとして、ハードウェア技術を適用して処理の高速化を図るアプローチの検討が開始されている[1]。筆者らもTCPなどの複雑な手順を有するプロトコルをFPGAにより実装する方法を検討している。これまでに、TCPのデータ転送部分を対象としてハードウェア記述言語VHDLを用いた実装を進め、送信部分の実装を完了した。本稿では、筆者らの実装のアプローチ、VHDLを用いた実際の実装法などについて述べる。

2. 方針

(1) 送信すべきユーザデータの発生と受信したユーザデータの消費を行うユーザモジュールと、TCPの送受信を行うTCPモジュールを、FPGA内に実現する。また、メモリと物理回線にデータを送受信する回線制御モジュールをFPGA外部に用意する。

(2) 個別のTCPコネクションの状態を管理するコネクションテーブルは、FPGA内部に用意されたRAM上に実現し、外部のメモリは送受信TCPセグメントのバッファリングのみに使用する。さらに、TCPの送受信処理を並行して実行できるよう、外部のメモリを送信メモリと受信メモリに区分する。

(3) 処理速度を向上させるために、メモリまたは回線制御モジュールへのアクセスと、プロトコル処理とを並行に行わせる。

(4) FPGAによりメモリ内のバッファ管理を行うため、できるだけ平易で高速なバッファ管理方式を採用する。よって間接アドレス指定やポインタを用いたリストなどメモリアクセス回数の多い方式ではなく、バッファを連続領域に用意し、ベースアドレス指定でアクセス可能な構造とする。

(5) FPGA内のTCPの実装については、処理の高速化のために、状態遷移を行う順序回路部分(クロックに同期)と出力信号を決定する組合わせ回路部分(クロックと非同期)の定義を独立に行う。

3. 実装の詳細

3.1 ハードウェア構成

上記の方針に従って設計したハードウェアのブロック構成を図1に示す。

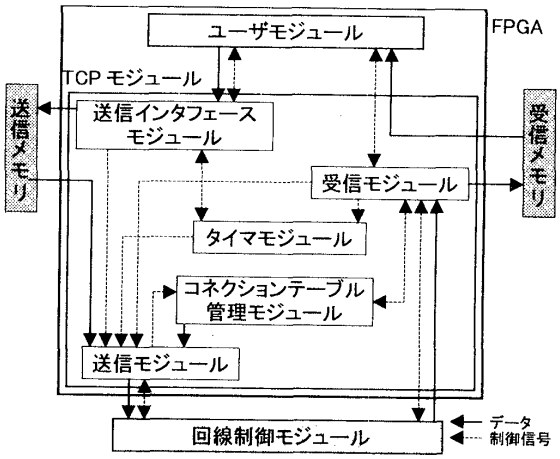

図1 ブロック構成図

ユーザデータの送信は、ユーザモジュールがTCPセグメントのユーザデータサイズ分のデータをメモリに書き込み、それにTCPのヘッダを付加して回線制御モジュールに送出するという処理手順を用いる。このため、送信のためには次の2つのモジュールを用意した。

・送信インタフェースモジュール：ユーザモジュールからの送信要求に対して、FPGA内のRAMにあるコネクションテーブルを検索し、対応する送信バッファの位置を求め、ユーザデータを送信バッファに書き込む。ここで、ユーザデータの書き込みに伴い、ユーザデータ部分のチェックサムを計算する。

・送信モジュール：TCPのフロー制御によりデータセグメントが送出可能な場合、ヘッダの作成、状態遷移、回線制御モジュールへの出力を行う。ここでヘッダの作成と状態遷移は、回線制御モジュールへの出力と並行して行う。

TCPセグメントを受信した場合は、受信モジュールにおいて、ヘッダを解析し、対応するコネクションテーブルを検索し、シーケンス番号に対応するユーザデータの格納位置を求め、受信バッファに書き込む。書き込みと並行にチェックサムを計算し、チェックサムが正しい場合に限りデータを有効とする。そのユーザデータがユーザモジュールに渡せる場合は、ユーザモジュールに受信を通知し、そのデータを読み込ませる。チェックサムが正しい場合には、状態やシーケンス番号などのコネクションテーブル情報の更新、応答された送信ユーザデータの解放などを行う。

タイマモジュールでは、定期的な時間計測を行い、RTTの測定、タイムアウトの検出と再送要求の生成を行う。また、コネクションテーブルへのアクセ

A Stady on Hardware Implementation of TCP Protocol
Atsushi TAGAMI†, Tomomi SATO‡, Akira IDOUE† and Toshihiko KATO†
† KDD R&D Laboratories, Inc.
‡ The University of Electro-Communications

スにおける調停、テーブルの検索などは、コネクションテーブル管理モジュールにより行われる。

3.2 バッファ管理

TCP コネクション毎に、送信待ちと確認待ちのセグメントを確保する送信用バッファおよび受信データの処理とユーザモジュールへの通知待ちのセグメントを確保する受信用バッファを、それぞれ用意する。これらのバッファは、管理の簡単化のために連続的な固定領域を使用する。

図 2 に送信用バッファの構成を示す。バッファは、バッファ制御情報、送信するセグメントの制御情報（シーケンス番号、データ部分のチェックサム、ユーザデータ長）を格納する領域、ユーザデータ自身を格納する領域から構成される。このうち、後者 2 つについては、連続領域を巡回して使用する。制御情報格納領域の先頭等については、バッファ制御情報により与えられ、ユーザデータについてはシーケンス番号によりアクセス可能である。

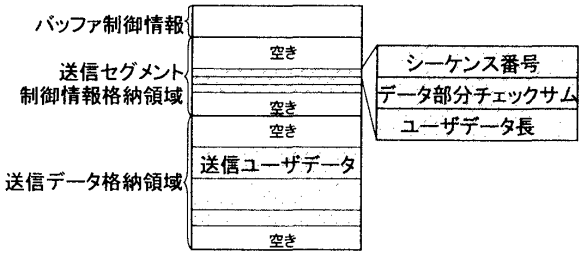

図 2　送信用バッファの構成

3.3 送信処理

図 3 に送信処理の流れをブロック図で示す。

送信インタフェースモジュールの制御部は、ユーザモジュールより送信要求(IP アドレス、PORT 番号、ユーザデータ長)を受け、コネクションテーブルを検索し、対応する送信バッファの位置に格納する。また、チェックサムモジュールは制御モジュールがユーザデータを格納するのと並行してチェックサムを計算する。 制御部は送信バッファにユーザデータを格納し終わると、その送信を送信モジュールに要求する。

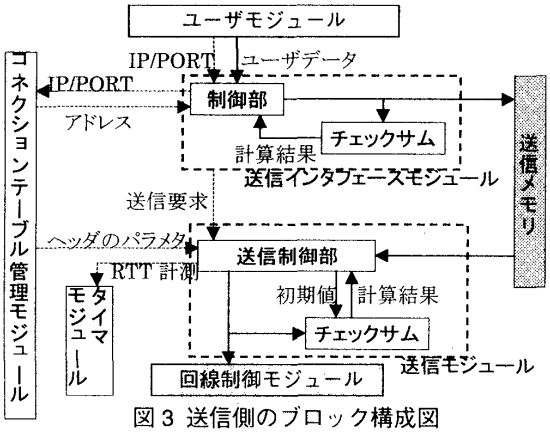

図 3 送信側のブロック構成図

送信モジュールの送信制御部は、送信バッファよりシーケンス番号を取得すると共に、コネクションテーブルよりその他のパラメタを取得しヘッダを生成する。チェックサムモジュールはあらかじめ送信バッファからユーザデータのチェックサムを取得し、ヘッダ送信と並行してヘッダ部分のチェックサムを計算し、ヘッダの最後に計算結果を追加する。

3.4 VHDL での実装

VHDL による実装例として送信インターフェイスモジュール(send_interface)の一部を図 4 に示す。 本モジュールは、クロックに同期して IDLE, REQ_ADDRESS(コネクションテーブルの検索中)などの状態(state)間で状態遷移を行う順序回路部分と、状態に応じて出力(outConnectiontableRequest 等)を決定する組合せ回路部分に分けて実装されている。

```
-- 使用するライブラリ
library ieee
  use ieee.std_logic_1164 ....
-- 入出力ピンの設定
entity send_interface
  port (clock  : in std_logic; reset : in std_logic; ...
        outAddress : out std_logic_vector(19 downto 0));
end entity send_interface;
-- モジュールの実体
architecture send_interface_architecture of send_interface is
-- 局地変数(register)
  type interface_state_type (IDLE, REQ_ADDRESS, .. );
  signal state : interface_state_type; ...
begin
-- クロックに同期する順序回路部分
process(clock, reset)
begin
-- 状態遷移
  if(reset = '1') then
    state <= IDLE -- 初期状態
  elsif(clock'event and clock = '1') then
    case state is
    when IDLE =>
    -- 開始シグナルが on('1')なら状態遷移を始める
      if (start = '1') then state <= REQ_ADDRESS
    when REQ_ADDRESS =>
    -- connection table の検索結果を register に代入する
      baseAddress <= inBaseAddress
    -- Connection Table の検索の終了を待つ
      if (found_connectiontable) then state <= REQUEST_SEQ...
    end case;
end process;
-- クロックに同期しない組合せ回路部分
-- コネクションテーブルに検索を要求する
 outConnectionTableRequest <= '1'
   when state=REQUEST or (state = IDLE and start='1')
                          else '0'; ...
end architecture send_interface_architecture;
```

図 4　VHDL によるモジュールの実装例

4. おわりに

本稿ではハードウェアによる高速な TCP の実装方式について述べた。 今後、実際にこの設計をもとに作成したハードウェアを用いて、 性能評価をおこなう予定である。最後に日頃ご指導頂く KDD 研究所村谷所長、鈴木副所長に感謝する。

参考文献

[1] 陣崎, 中村, 村井 “並列ネットワークサーバ comet のアーキテクチャとその応用,” 信学技報, CPSY98-62, Aug. 1998.