SiTCP ライブラリ
サンプルコード解説書

第 1.0.3 版

2022 年 6 月 1 日

㈱Bee Beans Technologies

改版履歴

版数	日付	内容
1.0.3	2022 年 6 月 1 日	初版

【目次】

1. 概要

本文書は、内田智久博士が作成した文書「SiTCP 説明書」「SiTCP ライブラリ」の補足資料です。SiTCP ライブラリを使用した Xilinx 社製評価ボード KC705 用 GMII インタフェースのサンプルソースコード「SiTCP_Sample_Code_for_KC705_GMII」（以下、本サンプル）について解説しています。

SiTCP は内田智久博士が開発し、（株）Bee Beans Technologies（以下 BBT）が管理・配布を行っているネットワーク・プロセッサです。

2. サンプルソースコード 概要

本サンプルは Xilinx 社が提供するソフトウェア「Vivado Design Suite」（以下 Vivado）での使用を前提とし、回路記述には Verilog を使用しています。本サンプルは Github からダウンロードすることができます。表 2-1 にファイル一覧を、図 2-1 に階層構造を示します。

表 2-1 ファイル一覧

ファイル種別	ファイル名	説明
トップモジュール	kc705sitcp.v	—
SiTCP 関連モジュール	WRAP_SiTCP_GMII_XC7K_32K.v	SiTCP ラッパーモジュール
	SiTCP_XC7K_32K_BBT_V110.v	SiTCP ポート宣言ファイル
	SiTCP_XC7K_32K_BBT_V110.ngc	SiTCP 本体（ngc ファイル）
	TIMER.v	SiTCP 用タイマモジュール
RBCP サンプルモジュール	RBCP.v	—
FIFO モジュール	fifo_generator_v11_0.v	FIFO ラッパーモジュール
	fifo_generator_v11_0.ngc	FIFO 本体（ngc ファイル）
EEPROM 関連モジュール	AT93C46_IIC.v	AT93C46 エミュレーション用モジュール
	BRAM128_9B9B.v	ブロック RAM 定義用モジュール
	PCA9548_SW.v	IIC スイッチ制御用モジュール
	IIC_CTL.v	IIC インタフェース用モジュール
	IIC_CORE.v	IIC シーケンスジェネレータ用モジュール
制約ファイル	kc705sitcp.xdc	KC705 制約ファイル
	SiTCP.xdc	SiTCP 制約ファイル

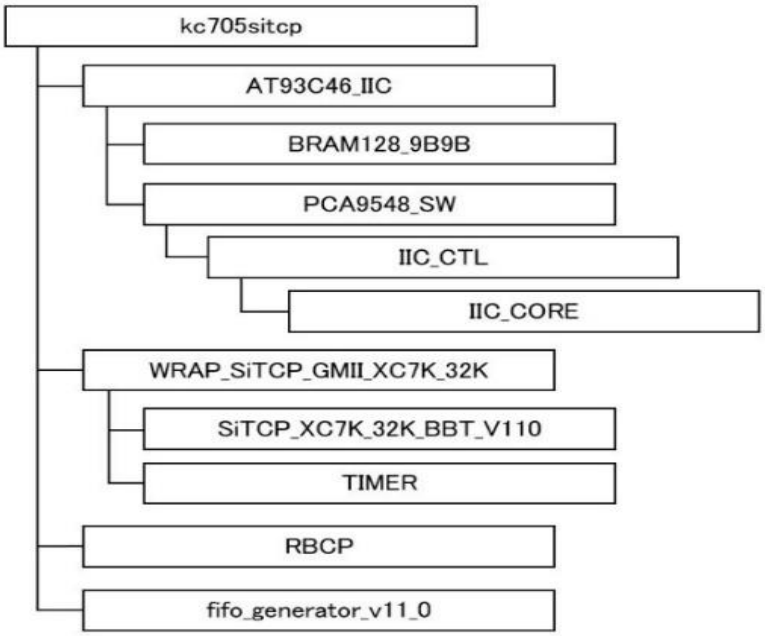

図 2-1 ファイル階層構造

3. サンプルソースコード　各種モジュール・回路

3.1. トップモジュール（kc705sitcp.v）

クロックの生成、各種 wire 定義、モジュールのインスタンシエーション等を行うモジュールです。

3.1.1. クロックの生成

外部入力の 200MHz 差動クロックを IBUFDS(①)に入力し 200MHz のシングルエンドクロックを生成後、PLL(②)を介して 200MHz のシステムクロック(③)および GMII 用の 125MHz クロック(④)を生成しています。

```
IBUFDS #(.IOSTANDARD ("LVDS"))
        LVDS_BUF(.O(BUF_CLK_200M), .I(SYSCLK_200MP_IN), .IB(SYSCLK_200MN_IN));  // ①

PLLE2_BASE #(
        .CLKFBOUT_MULT              (5),
        .CLKIN1_PERIOD              (5.000),
        .CLKOUT0_DIVIDE             (8),
        .CLKOUT0_DUTY_CYCLE         (0.500),
        .CLKOUT1_DIVIDE             (5),
        .CLKOUT1_DUTY_CYCLE         (0.500),
        .DIVCLK_DIVIDE              (1)
)
PLLE2_BASE(                                     // ②
        .CLKFBOUT                   (PLL_CLKFB),
        .CLKOUT0                    (CLK_125M),    // ④
        .CLKOUT1                    (CLK_200M),    // ③
        .CLKOUT2                    (),
        .CLKOUT3                    (),
        .CLKOUT4                    (),
        .CLKOUT5                    (),
        .LOCKED                     (LOCKED),
        .CLKFBIN                    (PLL_CLKFB),
        .CLKIN1                     (BUF_CLK_200M),
        .PWRDWN                     (1'b0),
        .RST                        (1'b0)
);
```

3.1.2. GMII 関連回路

GMII を用いて PHY と通信を行うため、各種信号を接続する必要があります。送信では、MII 動作時は PHY の出力するクロックを使用しますが、GMII 動作時は FPGA で生成した 125MHz クロックを PHY に向け出力する必要があります（受信では MII/GMII 共に PHY からのクロックを使用します）。そのため、BUFGMUX（⑤）を使用して、MII の場合は GMII_TX_CLK に、GMII の場合は CLK_125M に切り替えます。さらに、クロックをピンに出力するために ODDR（⑥）を使用しています。GMII_1000M（⑦）は、通信リンクが 1Gbps の場合 1、100Mbps または 10Mbps の場合 0 となります。

```
// ⑤, ⑦
BUFGMUX GMIIMUX(.O(BUF_TX_CLK), .I0(GMII_TX_CLK), .I1(CLK_125M), .S(GMII_1000M));
// ⑥
ODDR     IOB_GTX
      (.C(BUF_TX_CLK), .CE(1'b1), .D1(1'b1), .D2(1'b0), .R(1'b0), .S(1'b0), .Q(GMII_GTXCLK));
```

PHY から出力される RX_CLK(GMII_RX_CLK)はリンク速度によって周波数が変化するクロックです（1Gbps リンク時：125MHz、100Mbps リンク時：25MHz、10Mbps リンク時：2.5MHz）。本サンプルでは、これを RX_CNT[6:0]によってカウントし、一定周期でリセットすることで GMII_1000M（⑦）の値を決定しています。

```
always@(posedge CLK_200M or negedge SYS_RSTn)begin
      if (~SYS_RSTn) begin
            CNT_CLK[8:0]  <=   9'b0;
            CNT_LD        <=   1'b0;
            CNT_RST       <=   1'b1;
            GMII_1000M    <=   1'b0;
      end else begin
            CNT_CLK[8:0]  <=   CNT_CLK[8] ? 9'd198 : CNT_CLK[8:0] - 9'd1;
            CNT_LD        <=   CNT_CLK[8];
            CNT_RST       <=   CNT_LD;
            GMII_1000M    <=   CNT_LD ? RX_CNT[6] : GMII_1000M;    // ⑦
      end
end

always@(posedge GMII_RX_CLK or posedge CNT_RST)begin
      if (CNT_RST) begin
            RX_CNT[6:0]   <=   7'd0;
      end else begin
            RX_CNT[6:0]   <=   RX_CNT[6] ? RX_CNT[6:0] : RX_CNT[6:0] + 7'd1;
      end
end
```

GMII_MDIO は 3 ステートバッファを使用して SiTCP に接続します（入出力ポート解説書を参照）。なお、本サンプルでは IOBUF プリミティブを使用しています。

```
IOBUF     #(.DRIVE(4), .SLEW("SLOW")) ETH_MDIO_BUF
      (.O(GMII_MDIO_IN), .IO(GMII_MDIO_IO), .I(GMII_MDIO_OUT), .T(~GMII_MDIO_OE));
```

3.1.3. システムリセット

KC705 の押しボタン SW2（トップモジュールのポート名 SW_N_IN）を押した場合、またはPLL の位相ロックが外れた場合にリセットが行われ、一定時間経過すると解除されます。なお、SYS_RSTn は 0 でリセット(負論理)の信号です。また、押しボタンの信号（SW_N_IN）はメタ・ステーブル防止回路（⑧）に入力されます。

```
always@(posedge CLK_200M or negedge LOCKED)begin
	if (~LOCKED) begin
		INICNT[29:0]	<= 30'd0;
		SYS_RSTn	<= 1'b0;
	end else begin
		if (RESET_SW) begin
			INICNT[29:0]	<= 30'd0;
			SYS_RSTn	<= 1'b0;
		end else begin
			INICNT[29:0]	<= INICNT[29]?	INICNT[29:0]	:(INICNT[29:0] + 30'd1);
			SYS_RSTn	<= INICNT[29];
		end
	end
end

always@(posedge CLK_200M)begin
	if(~SYS_RSTn)begin
		SW_REG[3:0]		<=	4'b0000;
		RESET_CUNT[26:0]	<=	27'd0;		// 336ms Timer
		RESET_SW		<=	1'b0;
	end else begin
		SW_REG[3:0]		<=	{SW_REG[2:0] ,SW_N_IN};	// ⑧
		RESET_CUNT[26:0]	<=	~SW_REG[3]? 27'd0:	(RESET_CUNT[26:0] +
{26'd0, ~RESET_CUNT[26]});
		RESET_SW		<=	 RESET_CUNT[26];
	end
end
```

3.2. SiTCP 関連モジュール

本サンプルでは、トップモジュールで SiTCP のラッパーモジュールをインスタンシエーションすることで SiTCP の呼び出しを行っています。

3.2.1. IP アドレス、TCP ポート、RBCP ポートの設定

SiTCP は、ラッパーモジュール内の assign 文（⑨等）により、IP アドレス・TCP ポート番号および RBCP ポート番号として DEFAULT から始まる内部レジスタの値を使用します（SiTCP 内部レジスタ解説書を参照）。これらの設定値として任意の値を入力する場合は assign 文を修正するか、SiTCP の IP_ADDR_IN、TCP_MAIN_PORT_IN、RBCP_PORT_IN ポートに直接代入します（⑩）。

FORCE_DEFAULTn の値に関わらず外部の DIP スイッチ等の入力値を適用する場合（⑪）は、SiTCP の上記ポートに EXT から始まる信号を接続した上で、トップモジュールでインスタンシエーションしているラッパーモジュールの EXT から始まる当該ポートに DIP スイッチ等の信号を接続します。

```
// ラッパーモジュール（WRAP_SiTCP_GMII_XC7K_32K. v）での記述
assign      MY_IP_ADDR[31:0] = (~FORCE_DEFAULTn | (EXT_IP_ADDR[31:0]==32'd0) ?
DEFAULT_IP_ADDR[31:0]       : EXT_IP_ADDR[31:0]);   // ⑨
・・・

SiTCP_XC7K_32K_BBT_V110 SiTCP(
      .CLK                          (CLK                    ),      // in  : System clock
・・・
      .IP_ADDR_IN                   (MY_IP_ADDR[31:0]),             // in   : My IP address[31:0]
      .IP_ADDR_DEFAULT              (DEFAULT_IP_ADDR[31:0]),    // out: Default value for…
      .MY_MAC_ADDR                  (),                                  // out: My MAC address[47:0]
      // SiTCPポートに設定値を直接代入（記述例、⑩）
      .TCP_MAIN_PORT_IN             (16'h0018),                     // in: My TCP main-port #[15:0]
      .TCP_MAIN_PORT_DEFAULT (DEFAULT_TCP_PORT[15:0]),// out : Default value ...
・・・
      // 外部の値を入力（記述例、⑪）
          //トップモジュールのラッパーモジュール インスタンスのポートへ代入が必要
      .RBCP_PORT_IN                 (EXT_RBCP_PORT[15:0] ),         // in: My: Default value …
      .RBCP_PORT_DEFAULT        (DEFAULT_RBCP_PORT[15:0]),   // out   : Default …
・・・

// トップモジュール（kc705sitcp. v）での記述
WRAP_SiTCP_GMII_XC7K_32K
・・・
      // DIPスイッチの値を入力（記述例、⑪）
      .EXT_RBCP_PORT                (DIP[15:0]),                                  // in    : RBCP port #[15:0]
・・・
```

3.2.2. タイマモジュール（TIMER.v）

SiTCP 同様、ラッパーモジュールでインスタンシエーションされているモジュールです。クロックを入力することで 1us,1ms および 1s に CLK の 1 パルス幅 1 となるタイミング信号を出力します。なお、使用するクロック周波数の値（TIM_PERIOD）を上位モジュールからパラメータとして設定する必要があります。

```
// ラッパーモジュール内でのインスタンシエーション
TIMER #(TIM_PERIOD - 8'd2)  TIMER(
        // System
        .CLK        (CLK              ),          // in  : System clock
        .RST        (RST              ),          // in  : System reset
        // Intrrupts
        .TIM_1US (TIM_1US         ),          // out: 1 us interval
        .TIM_1MS(TIM_1MS         ),          // out: 1 ms interval
        .TIM_1S   (TIM_1S           ),          // out: 1 s interval
        .TIM_1M   (TIM_1MIN       )           // out: 1 min interval    // ※
        );

// トップモジュールでのラッパーモジュールのパラメータ設定（200MHzの場合）
WRAP_SiTCP_GMII_XC7K_32K      #(
        .TIM_PERIOD  (8'd200)                 // = System clock frequency(MHz), integer only
        )
・・・
```

※　TIM_1M ポートは現在使用していませんが、将来の互換性確保のため接続してください

3.3. RBCP サンプルモジュール（RBCP.v）

RBCP レジスタの制御を行うモジュールです。このモジュールでは例として RBCP アドレス 0x00000000~0x00000003 の計 4byte の領域のみを設けていますが、下記のアドレス部分の記述を変更することでそれ以降の領域も使用できます。なお、0xFFFF0000~0xFFFFFFFF は SiTCP 内部レジスタ用に予約されているため使用できません。

3.3.1. パイプライン処理

このモジュールでは RBCP 制御信号について 2 段のパイプライン処理を行っています。1 段目では SiTCP が出力した RBCP アドレスが 0x00000000~0x00000003 の範囲かどうか（⑫）、2 段目では 4 つのアドレスのどれに書き込み・読み込みが発生しているか（⑬）を確認します。信号のタイミングを合わせるために RBCP_WE と RBCP_RE を遅延させています（⑭）。

```
always@(posedge CLK)begin
    // 1st
    P0WE                <= RBCP_WE;   // ⑭
    P0RE                <= RBCP_RE;
    P0_WD[7:0]          <= RBCP_WD[7:0];
    P0_ADDR_HI[1]       <= (RBCP_ADDR[31:16] == 16'd0);   // ⑫
    P0_ADDR_HI[0]       <= (RBCP_ADDR[15: 2] == 14'd0);
    P0_ADDR_LO[1:0]     <= RBCP_ADDR[1:0];
    // 2nd
    P1WE                <= P0WE;    // ⑭
    P1RE                <= P0RE;
    P1_WD[7:0]          <= P0_WD[7:0];
    P1_ADDR_HI[1:0]     <= P0_ADDR_HI[1:0];
    REG00_SEL           <= (&P0_ADDR_HI[1:0]) & (P0_ADDR_LO[1:0] == 2'b00);    // ⑬
    REG01_SEL           <= (&P0_ADDR_HI[1:0]) & (P0_ADDR_LO[1:0] == 2'b01);
    REG02_SEL           <= (&P0_ADDR_HI[1:0]) & (P0_ADDR_LO[1:0] == 2'b10);
    REG03_SEL           <= (&P0_ADDR_HI[1:0]) & (P0_ADDR_LO[1:0] == 2'b11);
     ・・・
```

3.3.2. レジスタへの書き込み

アドレス 0x00000000~0x00000003 に対応する x00Reg[7:0]~x03Reg[7:0]へデータを書き込む回路です。x00Reg[7:0]（⑮）には SW11（DIP スイッチ※）の 2 番～4 番の値が常時代入されます。また、PC からアドレス 0x00000001~0x00000003 へ書き込みが発生すると、アドレスが指定される（REG*_SEL）とともに RBCP_WE（P1WE）が 1 となり、RBCP_WD[7:0]（P1_WD[7:0]）のデータが次のクロックで x01Reg[7:0]~x03Reg[7:0]へ反映されます（⑯）。

```
    ・・・
        // Write value to register
    x00Reg[7:0]     <=   {5'd0,DIP[2:0]}; // read DIPswitch    // ⑮
    x01Reg[7:0]     <=   P1WE & REG01_SEL?   P1_WD[7:0]:   x01Reg[7:0];    // ⑯
    x02Reg[7:0]     <=   P1WE & REG02_SEL?   P1_WD[7:0]:   x02Reg[7:0];
    x03Reg[7:0]     <=   P1WE & REG03_SEL?   P1_WD[7:0]:   x03Reg[7:0];
    ・・・
```

※ KC705 の仕様上、基板上の DIP スイッチ番号と回路図の信号名の順番が逆転しているため、DIP スイッチの 4 番が DIP[0]に対応しています。

3.3.3. レジスタからの読み込みおよび ACK の応答

x00Reg[7:0]~x03Reg[7:0]の値を RBCP_RD[7:0]に出力する回路です。PC からの RBCP アドレス 0x00000001~0x00000003 への読み込みが発生すると、SiTCP は ACK が 1 の間のみ RBCP_RD[7:0]のデータを取り込みます。そのため、RBCP データの書き込みおよび読み込みが正常に終了した際は、RBCP_ACK を 1 クロックだけ 1 にすることで応答する必要があります。

このモジュールでは、0x00000000~0x00000003 の範囲にアクセスがあった時のみ((&P1_ADDR_HI[1:0])の部分)、RBCP_WE と RBCP_RE を遅延させたもの(P1WE、P1RE)を使用して ACK として応答しています(⑰)。データに関しては、P1RE が 1 かつ指定のレジスタにアクセスがあった時(REG*_SEL)にのみレジスタのデータを RBCP_RD[7:0]へ反映させるようにしています(⑱)。このように、それぞれ遅延させたものを使用することで RBCP_ACK と RBCP_RD[7:0]のタイミングを合わせることができます。

```
・・・
    // Read value from register
RBCP_RD[ 7:0] <=    (                                           // ⑱
    ((P1RE & REG00_SEL)?    x00Reg[7:0]:    8'h00)|
    ((P1RE & REG01_SEL)?    x01Reg[7:0]:    8'h00)|
    ((P1RE & REG02_SEL)?    x02Reg[7:0]:    8'h00)|
    ((P1RE & REG03_SEL)?    x03Reg[7:0]:    8'h00)
);
// ACK reply
RBCP_ACK      <=    (&P1_ADDR_HI[1:0]) & (P1WE | P1RE);    // ⑰
・・・
```

3.4. FIFO モジュール（fifo_generator_v11_0.v/ngc）

このモジュールは Xilinx の IP コアを使用して 4KB の FIFO バッファを生成したものであり、後述「4.6 TCP 通信」でのエコーバックに使用します。本サンプルでは SiTCP の TCP データ受信のためにトップモジュールでインスタンシエーションされており、SiTCP ラッパーモジュールとの間に下記の信号が接続されています（SiTCP の各種制御信号については入出力ポート解説書を参照）。SiTCP が PC 等から受信した TCP データを FIFO バッファに格納し、送信バッファがほぼ満杯であることを示す TCP_TX_FULL が 0 の場合にそのデータを返送する回路です。TCP セッションが確立していない時は FIFO バッファのデータはクリアされます。

なお、上記の 4KB というバッファサイズは SiTCP の受信機能を使用する上で必要最低限のバッファ容量です。SiTCP の受信機能を使用して高帯域の受信を行う場合は容量の大きな FIFO バッファを設けてください（最大バッファサイズは 64KB、入出力ポート解説書を参照）。

```
WRAP_SiTCP_GMII_XC7K_32K     #(
      .TIM_PERIOD  (8'd200)    // = System clock frequency(MHz), integer only
)
SiTCP     (
      .CLK                (CLK_200M           ),     // in  : System Clock (MII: >15MHz, …
・・・
      .TCP_OPEN_ACK   (TCP_OPEN_ACK   ),     // out: Acknowledge for open  (=Socket busy)
      .TCP_ERROR          (                   ),     // out: TCP error, its active period is equal to MSL
      .TCP_CLOSE_REQ (TCP_CLOSE_REQ ),     // out: Connection close request
      .TCP_CLOSE_ACK  (TCP_CLOSE_REQ ),     // in  : Acknowledge for closing
      // FIFO I/F
      .TCP_RX_WC          ({4'b1111,FIFO_DATA_COUNT[11:0]}),     // in  :Rx FIFO write …
      .TCP_RX_WR          (TCP_RX_WR          ),     // out: Write enable
      .TCP_RX_DATA        (TCP_RX_DATA[7:0]),     // out: Write data[7:0]
      .TCP_TX_FULL        (TCP_TX_FULL        ),     // out: Almost full flag
      .TCP_TX_WR          (FIFO_RD_VALID      ),     // in  : Write enable
      .TCP_TX_DATA        (TCP_TX_DATA[7:0]),     // in  : Write data[7:0]
・・・
);

fifo_generator_v11_0 fifo_generator_v11_0(
      .clk                (CLK_200M                   ),//in :
      .rst                (~TCP_OPEN_ACK              ),//in :
      .din                (TCP_RX_DATA[7:0]           ),//in :
      .wr_en              (TCP_RX_WR                  ),//in :
      .full               (                           ),//out :
      .dout               (TCP_TX_DATA[7:0]           ),//out :
      .valid              (FIFO_RD_VALID              ),//out :active hi
      .rd_en              (~TCP_TX_FULL               ),//in :
      .empty              (                           ),//out :
      .data_count         (FIFO_DATA_COUNT[11:0]      )//out:[11:0]
      );
```

3.5. EEPROM 関連モジュール

「AT93C46_IIC.v」とその他 4 種類のモジュールは、SiTCP のネットワークパラメータ保存用 EEPROM（AT93C46D が標準）のエミュレーションを行うモジュール群です。KC705 上の I2C EEPROM（M24C08）を SPI EEPROM の AT93C46D としてエミュレーションし、SiTCP から M24C08 へ SiTCP のネットワークパラメータ（SiTCP 説明書、SiTCP 内部レジスタ解説書を参照）の保存を行います。

本サンプルではこれらのモジュールをインスタンシエーションしているため、MPC 書き込みツールを使用すれば KC705 の M24C08 に SiTCP ライセンスファイルを書き込むことが出来ます。それによって、IP アドレスやその他設定値について EEPROM に保存されている値を使用することができます（後述 4.3.10 の ForceDefault モードを解除して通信することができます）。

3.6. 制約ファイル（kc705sitcp.xdc／SiTCP.xdc）

本サンプルで使用する制約ファイル（XDC ファイル）です。他のソースコード同様、Vivado のプロジェクトに add してください（設定方法は本サンプル同梱の“Notes on SiTCP.xdc.pdf“を参照してください）。

4. 本サンプルを用いた KC705 での動作確認方法

以下に、本サンプルを用いた KC705 での動作確認方法について、一連の手順を示します（Vivado は 2020.2、PC の OS は Windows 10 Pro を使用しています）。KC705 に MCS ファイルを書きこんで動作確認を行います。

4.1. プロジェクトディレクトリ作成

予め PC 上にダウンロードした本サンプルを保存しておきます。ここでは[KC705_GMII]ディレクトリをデスクトップに作成し、その中に本サンプルを保存します。

4.2. Vivado での合成・インプリメンテーション・bit/MCS ファイル生成

4.2.1. Vivado での合成・インプリメンテーション

(1) Vivado を起動し、[Quick Start]-[Create Project]をクリックします。

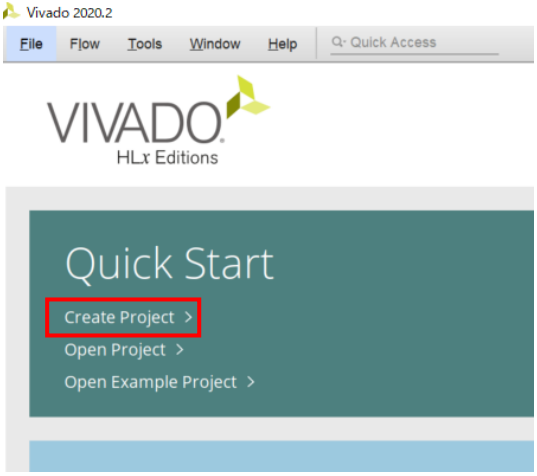

図 4-1 [Create Project]

(2) [Create a New Project]の表示が出たら[Next]をクリックします。

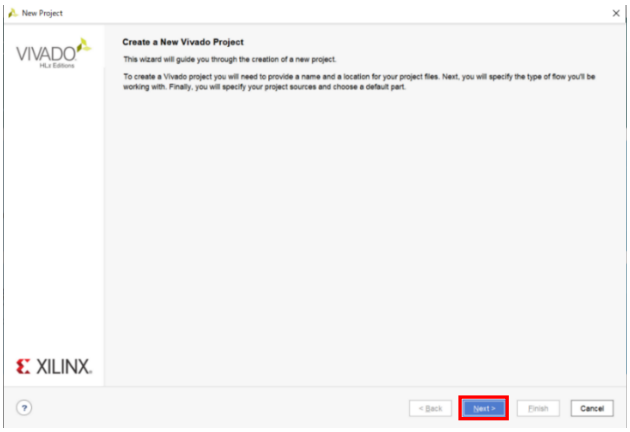

図 4-2 [Create a New Project]

(3) [Project Name]の表示が出たら[Project Name:]欄にプロジェクト名（例:project_1）を入力します。また、[Project location:]欄にプロジェクトファイルを保存する場所（作成した[KC705_GMII]ディレクトリ）を指定し、[Next]をクリックします。

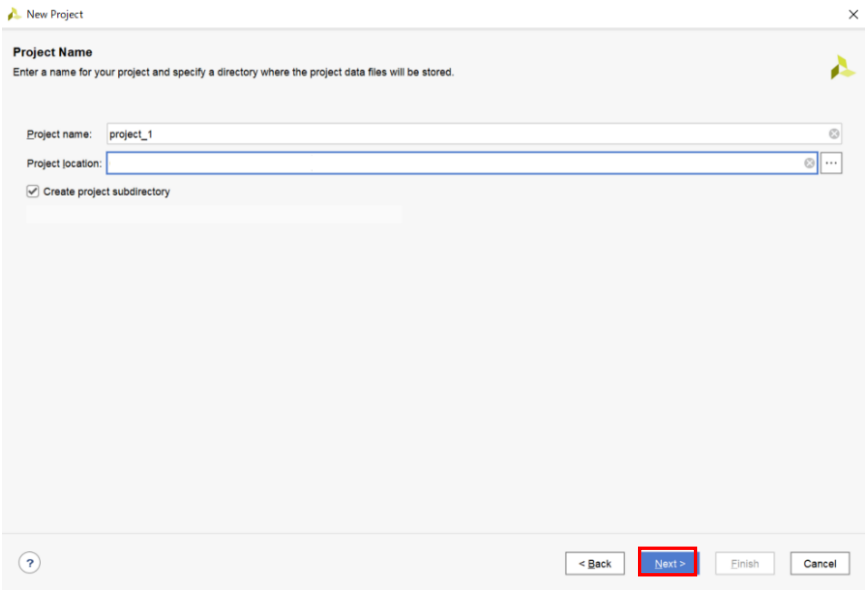

図 4-3 [Project Name]

(4) [Project Type]の表示ではデフォルト（[RTL Project]のラジオボタンが選択された状態）で[Next]をクリックします。

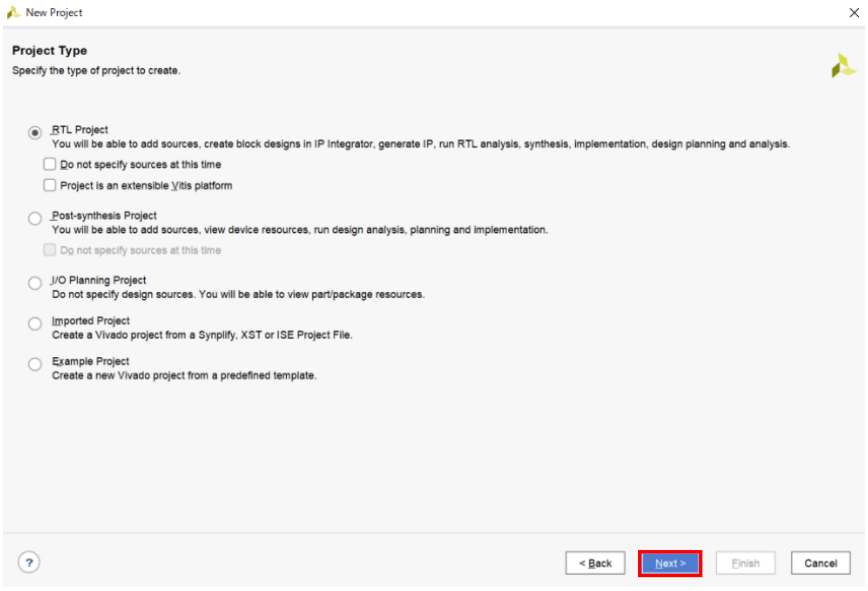

図 4-4 [Project Type]

(5) [Add Sources]で[Add Files]をクリックし、本サンプルのうち制約ファイル以外を選択して[OK]をクリック後、[Next]をクリックします。

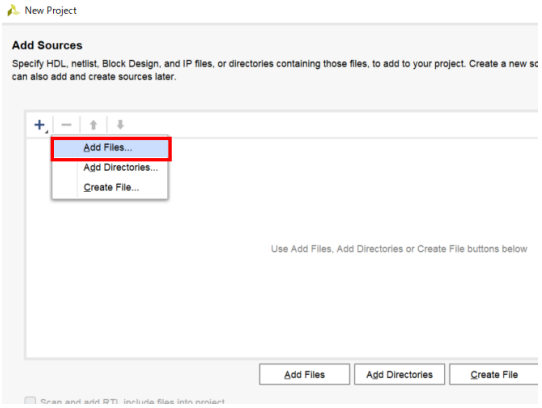

図 4-5 [Add Sources] – [Add Files]

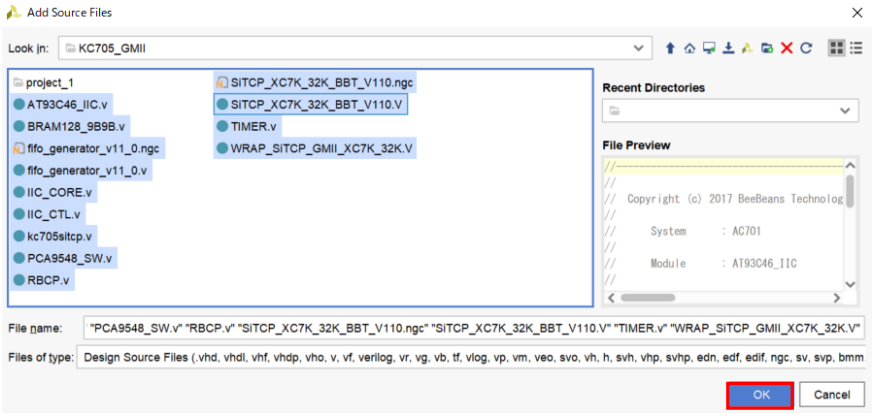

図 4-6 [Add Source Files]

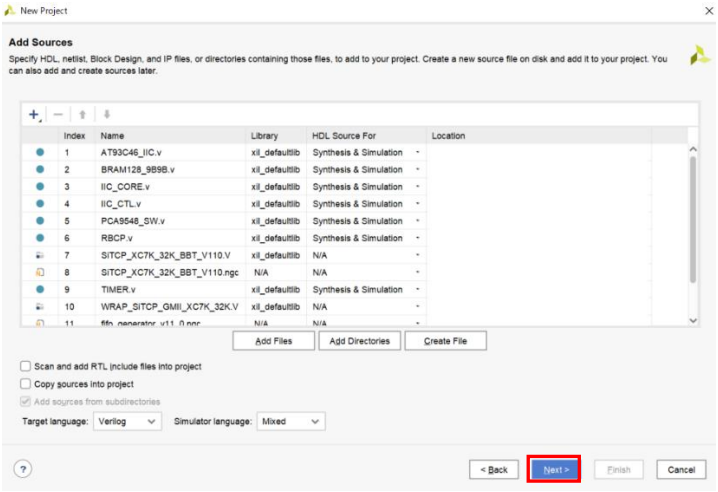

図 4-7 [Add Sources]

(6) [Add Constraints (optional)]でも同様に[Add Files]をクリックし、本サンプルの制約ファイルを選択して[OK]をクリックします。2 つの制約ファイルのうち、上から KC705 制約ファイル、SiTCP 制約ファイルの順で並ぶようにし、[Next]をクリックします。

図 4-8 [Add Constraint Files]

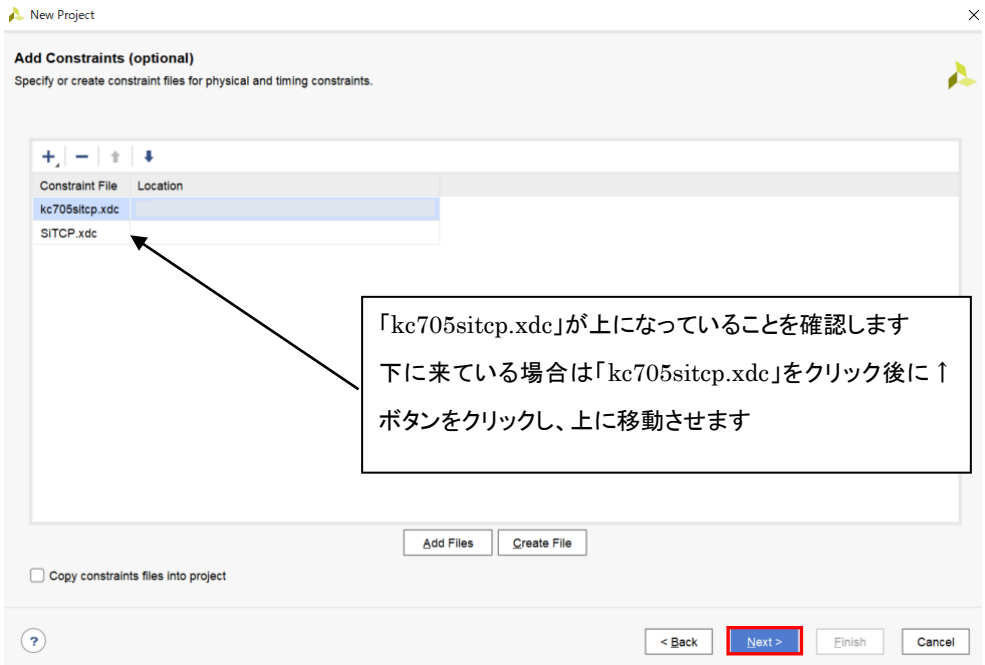

図 4-9 [Add Constraints (optional)]

(7) [Default Part]で[Boards]のタブを選択後、[Vendor]を[xilinx.com]、[Name]を[Kintex-7 KC705 Evaluation Platform]、[Board Rev.]をお持ちの KC705 の Board Revision に合わせて[Next]をクリックします。

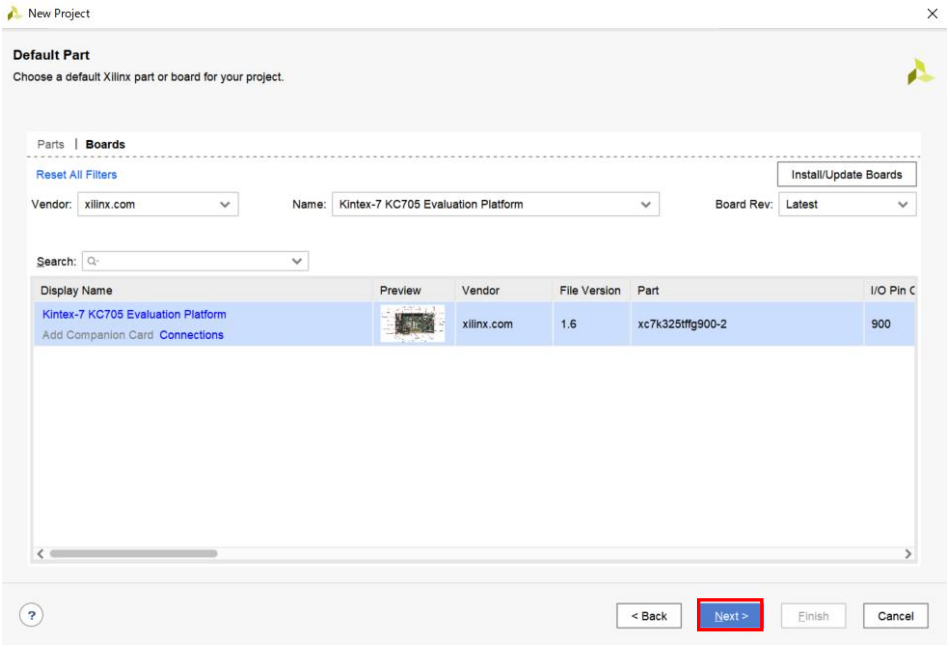

図 4-10 [Default Part]

(8) [New Project Summary]の表示が出たら[Finish]をクリックします。

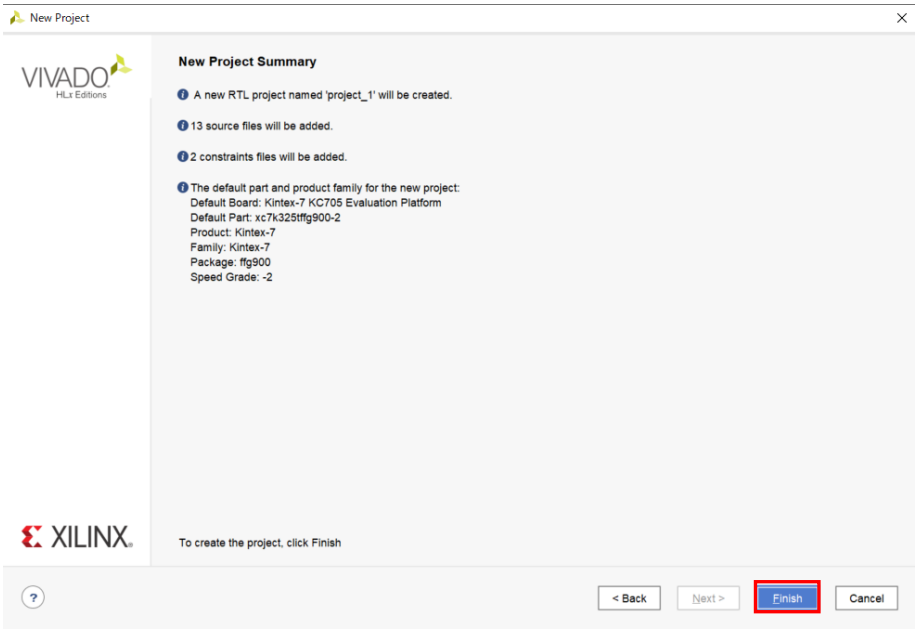

図 4-11 [New Project Summary]

(9) プロジェクトが起動したら[PROJECT MANAGER]の[Sources]ビューで、各ファイルが図 4-12 のようにツリー表示されることを確認します。

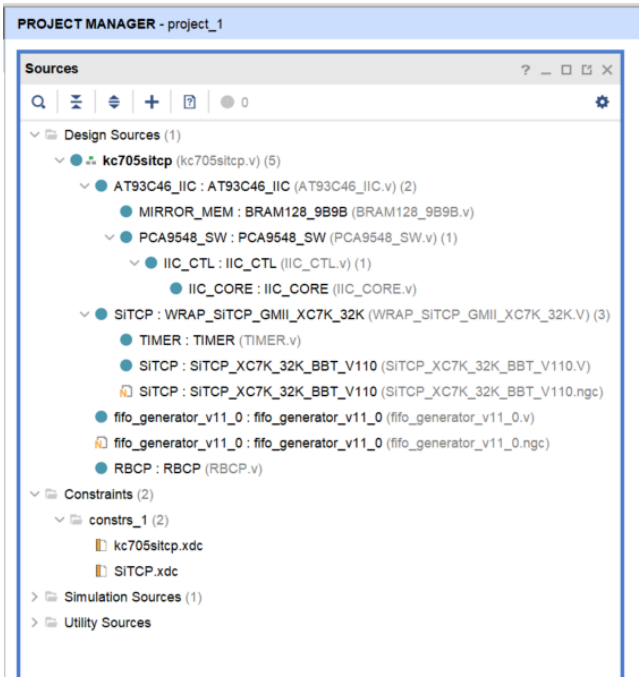

図 4-12 [PROJECT MANAGER] – [Sources]ビュー

(10) [Flow Navigator]の[PROGRAM AND DEBUG]-[Generate Bitstream]をクリックして bit ファイルを生成します（図 4-13）。図 4-14 のようなポップアップが出たら[Yes]を、[Launch Runs]（図 4-15）では[OK]をクリックすると Vivado で合成、インプリメンテーションおよび bit ファイル生成を続けて行います。

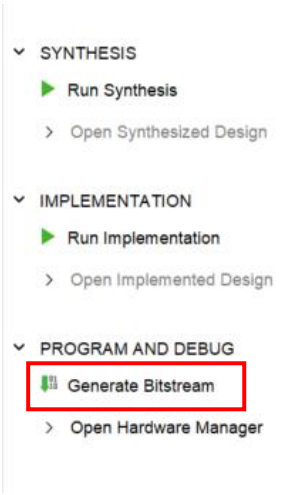

図 4-13 [Flow Navigator] – [PROGRAM AND DEBUG] – [Generate Bitstream]

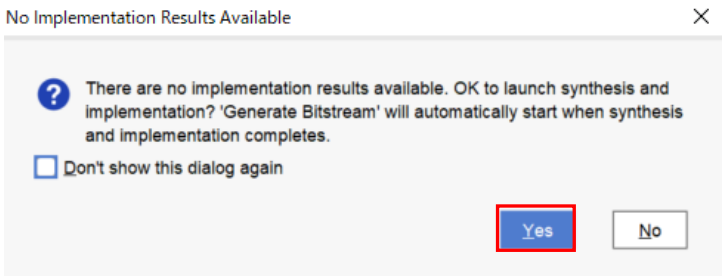

図 4-14 [No Implementation Results Available] のポップアップ

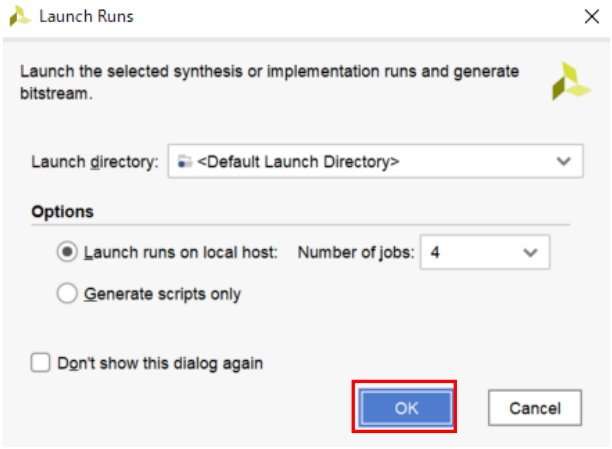

図 4-15 [Launch Runs]

(11) [Bitstream Generation Completed]のポップアップが出たら bit ファイル生成終了です。続いて[Generate Memory Configuration File]のラジオボタンを選択し、[OK]をクリックして MCS ファイルの生成へ移ります。

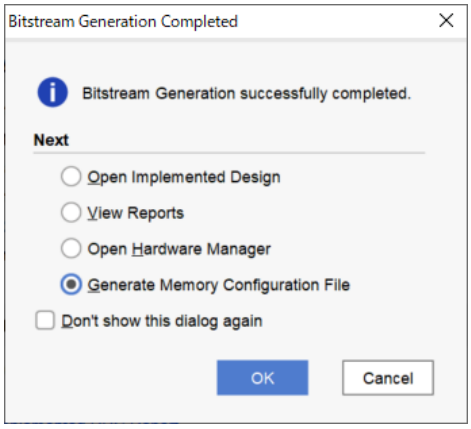

図 4-16 [Bitstream Generation Completed]

4.2.2. bit/MCS ファイル生成

(1) [Write Memory Configuration File]が表示されたら[Create a configuration file to program the device] – [Memory Part]のラジオボタンをクリック後、[…]ボタンをクリックして[Select Configuration Memory Part]を開きます。[Filter]で[Manufacturer]を Micron、[Density(Mb)]を 128、[Type]を spi とし、[Select Configuration Memory Part]で mt25ql128 を選択して[OK]をクリックします。

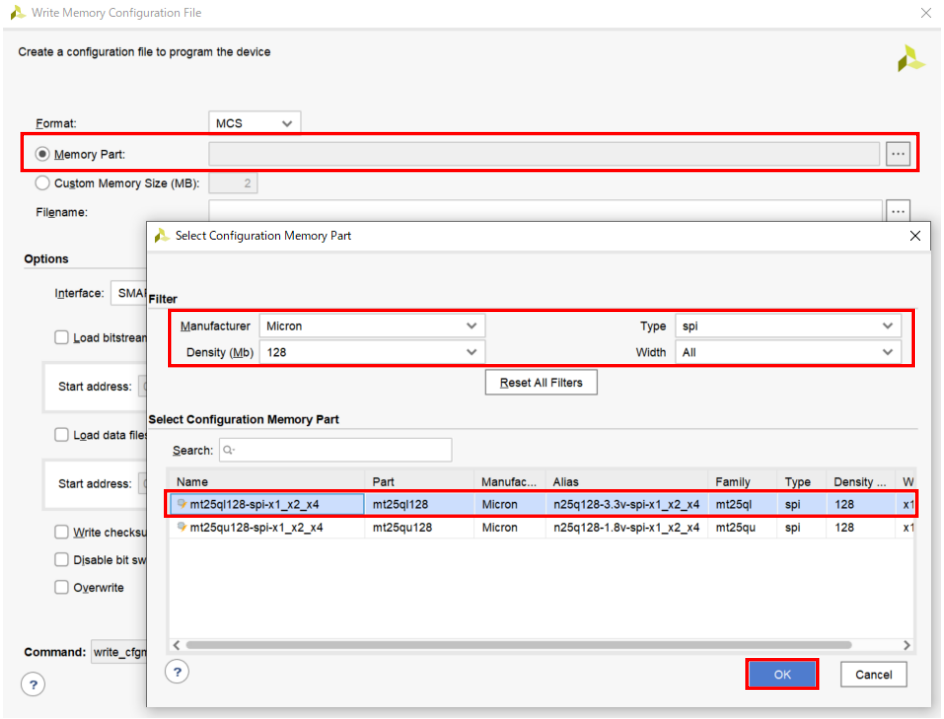

図 4-17 Memory Part の選択

(2) [Filename]の[…]ボタンをクリック後、[Specify Configuration Filename]のポップアップが出たら、MCS ファイルの保存ディレクトリを選択するとともに、[File name]の欄にファイル名を入力し(ここでは例として「kc705_sitcp」)、[Save]をクリックします。

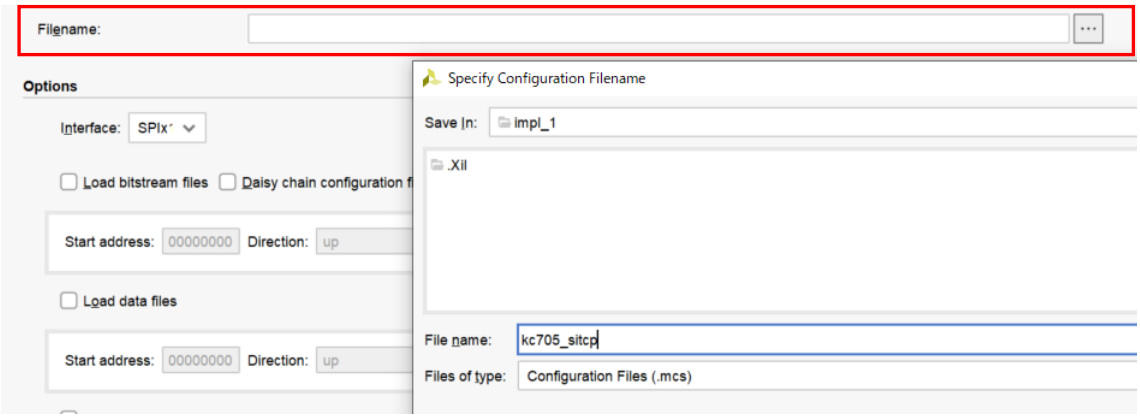

図 4-18 MCS ファイルの Filename を設定

(3) [Options]で[Interface]を[SPIx4]とし、[Load bitstream files]のチェックを入れた後、[bitfile]欄の隣の[…]ボタンをクリックします。[Specify Datafile Filename]のポップアップが出たら、生成した bit ファイル※を選択し、[OK]を押下します。

※ ここまでの設定どおりの場合、「～¥KC705_GMII¥project_1¥project_1.runs¥impl_1」のディレクトリに保存されています。

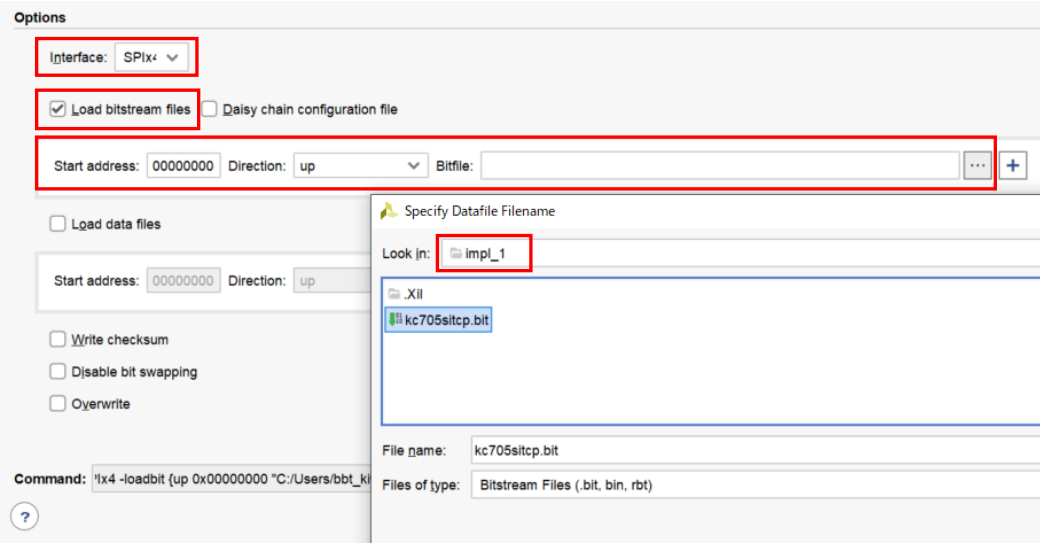

図 4-19 bit ファイルの指定

(4) 再び[Write Memory Configuration File]に戻ったら最下部の[OK]をクリックし、「Generate memory configuration file completed successfully」のメッセージが出れば MCS ファイル作成完了です。

4.3. KC705 への FPGA プログラム（MCS ファイル）ダウンロード手順

4.3.1. KC705 DIP スイッチ・ジャンパー設定

KC705 に電源を投入する前に、以下を設定しておきます。

- SiTCP を ForceDefault モード（※）に設定
  - SW11（DIP スイッチ）の 1 番スイッチを OFF に設定
- FPGA コンフィギュレーションモードをマスターSPI に設定
  - SW13（DIP スイッチ）の 3 番～5 番スイッチを OFF、OFF、ON に設定
- FPGA と PHY デバイス間の通信インタフェースを GMII に設定
  - J29 のピン 1-2 間をジャンパー接続
  - J30 のピン 1-2 間をジャンパー接続
  - J64 はジャンパーなし

※ SiTCP は初期状態では ForceDefault モードで通信する必要があり、本サンプルでは SW11 の 1 番スイッチで設定を行います。MAC アドレス、IP アドレス、ポート番号は表 4-1 に示す値となります。MAC アドレスを除きこれらの値はレジスタの初期値であり、ポートから任意の値を設定できます。なお、SiTCP ライセンスファイルを書きこむことでグローバル MAC アドレスが付与され、同一ネットワーク内で 2 台以上の通信が可能となります。

表 4-1 ForceDefault モードでのネットワークパラメータ

項目	設定値
MAC アドレス	02:00:C0:A8:00:10
IP アドレス※	192.168.10.16
TCP ポート番号※	24
RBCP ポート番号※	4660

4.3.2. KC705 へ MCS ファイルをダウンロード

(1) PC と KC705 を USB ケーブル（A-to-micro-B）で接続し、KC705 の電源を ON にします。

(2) Vivado の [Flow Navigator] で [PROGRAM AND DEBUG]-[Open Hardware Manager]-[Open Target]-[Auto Connect]をクリックします。

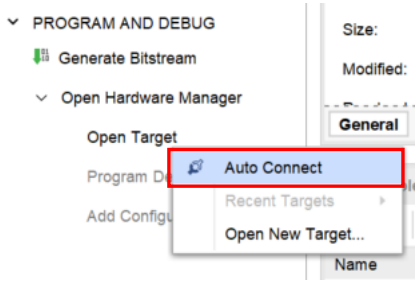

図 4-20 [Auto Connect]を選択

(3) Hardware Manager が起動したら、FPGA のアイコン(xc7k325t_0)を右クリックし、[Add Configuration Memory Device…]をクリックします。

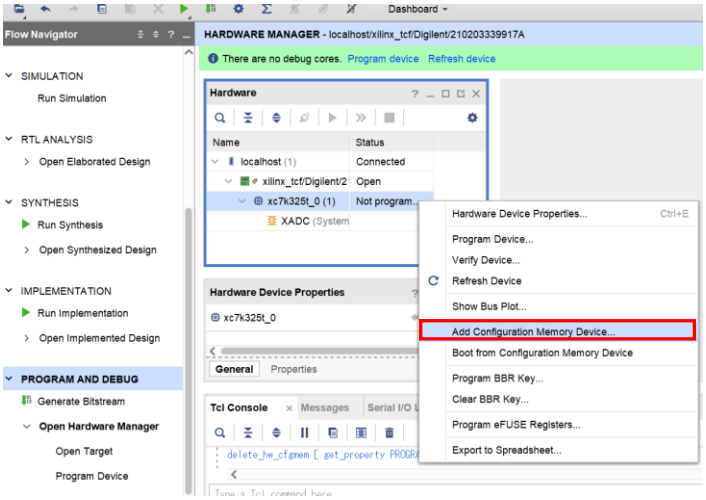

図 4-21 [Add Configuration Memory Device…]での選択

(4) [Choose a configuration memory part]では、図 4-17 同様に mt25ql128 を選択し、[OK]をクリックします。[Add Configuration Memory Device Completed]のポップアップが出るので、ここでも[OK]をクリックします。

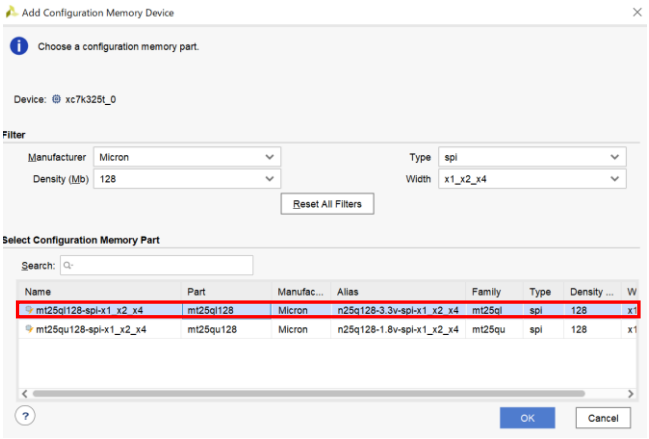

図 4-22 [Choose a configuration memory part]での選択

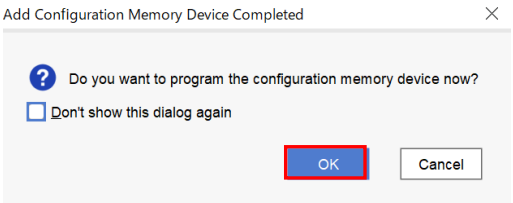

図 4-23 [Add Configuration Memory Device completed]

(5) [Program Configuration Memory Device]の[Configuration file:]欄で[…]ボタンからMCS ファイルを選択します。また、[Program Operations]の[Erase][Program][Verify]にチェックが入っていることを確認後[OK]をクリックすると MCS ファイルのダウンロードが始まります。

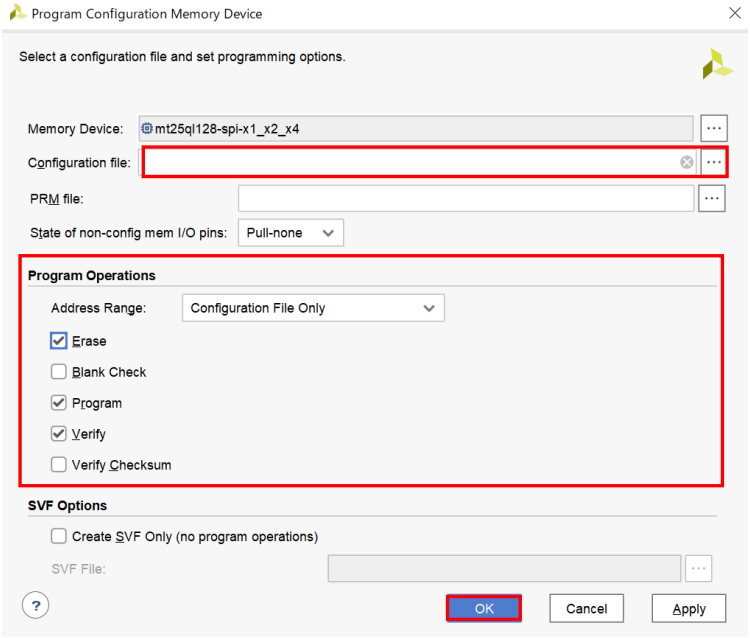

図 4-24 [Program Configuration Memory Device]

(6) [Flash programming completed successfully]の表示が出たら MCS ファイルのダウンロードは完了です。

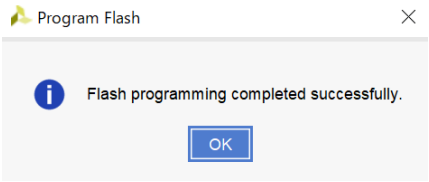

図 4-25 [Flash programming completed successfully]

4.4. KC705 との通信の確認

MCS ファイルを KC705 へダウンロード後、電源を切断・再投入します。KC705 と IP アドレスを設定した PC(※1)を LAN ケーブルで接続し、SiTCP Utility(※2)を用いて以下のように SiTCP との通信を確認します。

PC 上で SiTCP Utility を起動すると図 4-26 の画面([一般]タブ)が表示されます。制御対象の欄に KC705 の SiTCP へ設定した IP アドレスおよびポート番号(※3)を入力し、「読み込み」ボタンを押します。読み込み結果に MAC アドレス等が表示されれば PC と KC705 が正しく通信できています。結果が表示されない場合は IP アドレス等の設定を確認してください。

図 4-26 SiTCP Utility 起動画面

※ 1: PC の IP アドレス設定例:192.168.10.100(サブネットマスク 255.255.255.0)

※ 2: SiTCP Utility は BBT のホームページ-[サポート]-[ダウンロード]-[SiTCP 関連](https://www.bbtech.co.jp/download-files/sitcp/index.htm)からダウンロードできるソフトウェアです。

※ 3: [一般]タブでの通信に必要となるのは UDP ポート番号のみですが、TCP ポート番号もこの時点で設定してください。

4.5. RBCP 通信の確認

SiTCP Utility を使用して、サンプルコードで記述した RBCP レジスタ領域について通信の確認を行います。SiTCP Utility を起動し、[制御(UDP)]タブを開くと図 4-27 の画面が表示されます。書き込みは、レジスタアドレス欄に書き込みを行う領域の先頭アドレス、データ長欄に Byte 数を入力し、データ欄に書き込みデータ(16 進数)を入力して書き込みボタンを押すことで実行できます。読み込みは、レジスタアドレス欄に読み込みを行う領域の先頭アドレス、データ長欄に Byte 数を入力し、読み込みボタンを押すことで実行できます。

具体的には以下の手順で本サンプルの動作を確認できます(書き込むデータは 00~FF の任意)。

- [レジスタアドレス]を 0、[データ長]を 1 として[読み込み]ボタンをクリックすると、DIP スイッチで設定した値が表示される(※)
- [レジスタアドレス]を 1、データ長を 3 として[読み込み]ボタンをクリックすると、初期値(00 00 00)が表示される、その後データを書き換えて[書き込み]ボタンをクリック後、再度[読み込み]ボタンをクリックすると書き換えた値が表示される
- [レジスタアドレス]を 4、[データ長]を 1 として[読み込み]ボタンまたは[書き込み]ボタンをクリックするとバスエラーが発生する(SiTCP Utility のログ欄にタイムアウト表示が出る)

※ 「RBCP.v」では、アドレス 0(0x0000_0000)のレジスタに DIP スイッチの信号が接続されているため、常に DIP スイッチの値が読み出されます。アドレス 1～3 は書き込んだ値を保持する記述になっているため、書き込んだ値が読み出されます。アドレス 4(0x0000_0004)はレジスタを記述していないため、バスエラーが発生します。

図 4-27 UDP 制御タブ

4.6. TCP 通信の確認

Telnet のエコーバック機能を用いて TCP 通信の確認を行います。PC から送信した文字列をサーバー(SiTCP)が受け取り、受け取った文字列を PC へ送り返す動作を確認します。

4.6.1. Telnet クライアントのインストール

Windows10 では初期設定で Telnet クライアントが無効になっているため、本サンプルでの動作確認時に有効化してください。[コントロールパネル]または[アプリと機能]から[プログラムと機能] – [windows の機能の有効化または無効化]をクリックします。表示される機能の項目から[Telnet クライアント]のチェックを入れることで Telnet クライアントを有効にすることができます(動作確認が終わったら再度無効に設定することを推奨します)。

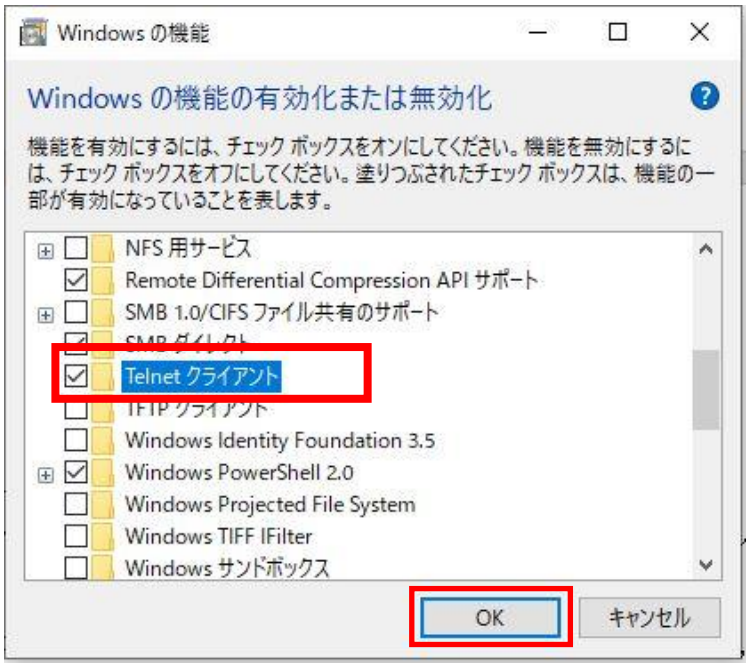

図 4-28 Telnet クライアントの有効化

4.6.2. Telnet による TCP 通信の確認

KC705 が通信可能な状態で、PC 上でコマンドプロンプトを起動し、”telnet 192.168.10.1624”を入力します（KC705 に設定した IP アドレスと TCP ポート番号）。キーボードで任意の文字キーを入力すると、入力した文字と TCP 通信のエコーバックにより返ってきた文字の合計 2 文字が表示されます。例えば「abcd」と入力すると、「aabbccdd」と表示されます。数十文字程度を連続で入力しエコーバックが確認できれば TCP 通信が正常に行われています。確認後、「 ctrl +] 」で終了します（以下に telnet コマンドの一覧を示します）。

Wireshark 等のソフトウェアで TCP パケットを確認することでより詳細な確認を行うことができます。

telnet コマンド一覧		
c	- close	現在の接続を終了
d	- display	パラメータを表示
o	- open	ホスト名 [ポート番号] ホスト名に接続(既定のポート番号は 23)
q	- quit	telnet を終了
sen	- send	サーバーに文字を送信
set	- set	オプションを設定('set ?' で一覧表示)
st	- status	状態を表示
u	- unset	オプションを解除('unset ?' で一覧表示)
?/h	- help	ヘルプを表示
Ctrl+]		エスケープ

5. 参考文献

SiTCP 説明書	内田智久／BBT
SiTCP ライブラリ	内田智久／BBT
Kintex-7 FPGA 用 KC705 評価ボード ユーザー ガイド(ug810)	Xilinx

6. 関連ドキュメント

SiTCP 内部レジスタ解説書	BBT
SiTCP 入出力ポート解説書	BBT