詳細評価報告書
MULTI-S01 H/W 実装評価

平成１３年1月１０日

暗号 H/W 実装

１．　目的

　Ｃプログラムで記述された暗号アルゴリズムを、ＦＰＧＡ（ハードウェア）に実装します。

２．　概要

２．１　暗号アルゴリズム

Ｃで作成された、「03Call6.c」のプログラムをＨ／Ｗ（FPGA）に実装します。

２．２　H／W設計方針

・H／W設計記述言語は、VerilogHDL を使用
・　ターゲットデバイスは、アルテラ社の「EP20K1000E」
・　データ入出力は，64bit 幅の同期式（１相クロック）
・　乱数発生器も他部と共通の１相クロックで動作させる。
　要求性能は 1/4 だが，非同期の同期化要素を排除するためです。
・　複合化時使用するA-1 鍵は, ソフトウェア生成としレジスタ設定としました。

３．　使用ツール

・**ModelSim VHDL/Verilog Version 5.4e (Model Technology)**
・Synplify（Synplicity Inc.）

４．　資料

４．１　Ｃプログラム

「03Call6.c」のプログラムリストを、「list-1」として添付します。

４．２　VerilogHDL ソース

ソースリスト（抜粋）を、「list-2」として添付します。

５．　結果

「ModelSim」を使用して、シミュレーションを実行し、「Ｃプログラム」の実行結果と同じデータ（暗号と復号のデータ）が得られることを確認しました。

論理合成ツールの結果（抜粋）を、「list-3」として添付します。

ＦＰＧＡによる評価結果を以下に示します。

５．１　「s01_top.v」

１）　動作クロック　：18.8 MHz

２）　実行速度　　：1．203ＧＢＰＳ（６４ｂｉｔ／クロック）

３）　回路規模　　：19811 ATOMs of 42240 (46%)

５．２　評価結果の見積もり条件

１）　実行速度の見積もりには、パラメータ・データの設定時間及びイニシャライズ時間は含みません。

２）　動作クロックと回路規模の見積もりは、「Synplify 」の　実行結果です。

６．　ＦＰＧＡ仕様

６．１　「s01_top.v」

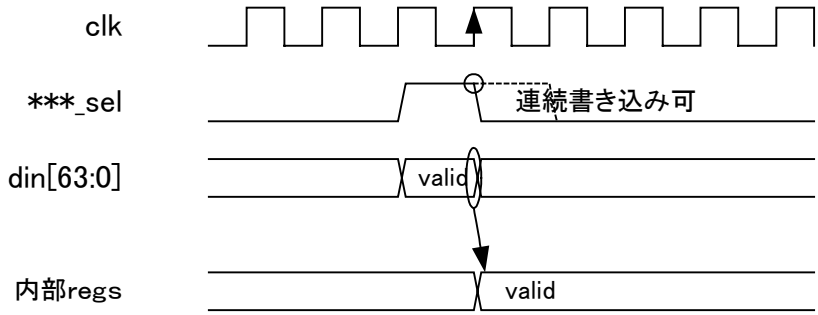
（１）　入出力信号

```
input           clk;            // clock
input           clrn;           // async clear
input           rst;            // sync clear
input           mode;           // mode sel   0:enc / 1: dec

input           len_sel;        // length area select
input           key_sel;        // key area select
input           divp_sel;       // divpara area select
input           din_sel;        // data area select
input           ainv_sel;       // A_INV area select
input           start;          // start flag
input   [63:0]  din;            // data input

output  [63:0]  dout;           // data output
output          din_req;        // data input req
output          dout_en;        // data output enable
```

端子概要

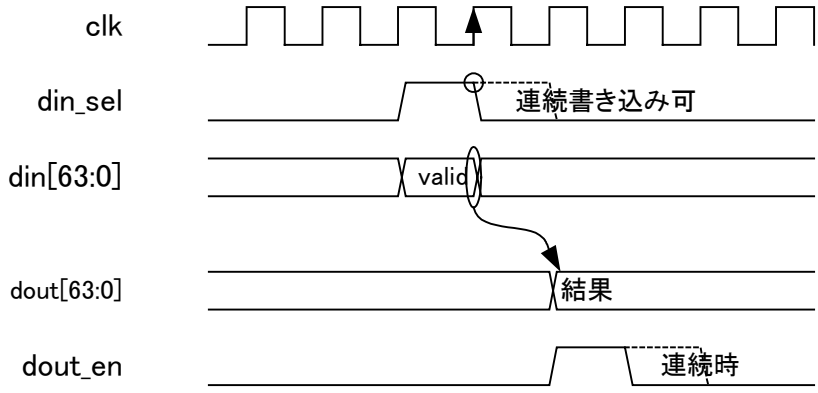
mode	：暗号化処理時 LOW、複合時 HIGH とする。
len_sel	：処理するデータ数（64bit 単位）設定に使用
key_sel	：KEY パラメータ設定に使用
divp_sel	：divparam パラメータ設定に使用
din_sel	：暗号/複合データ入力に使用
ainv_sel	：複合時、A-1 の設定に使用
start	：パラメータ設定完了の通知（初期化の起動）
din	：64bit（8BYTE）のデータ入力
dout	：64bit（8BYTE）のデータ出力
din_req	：初期化完了し、データ入力可能の通知
dout_en	：dout に有効なデータが存在する通知

（２）　パラメータの設定シーケンス

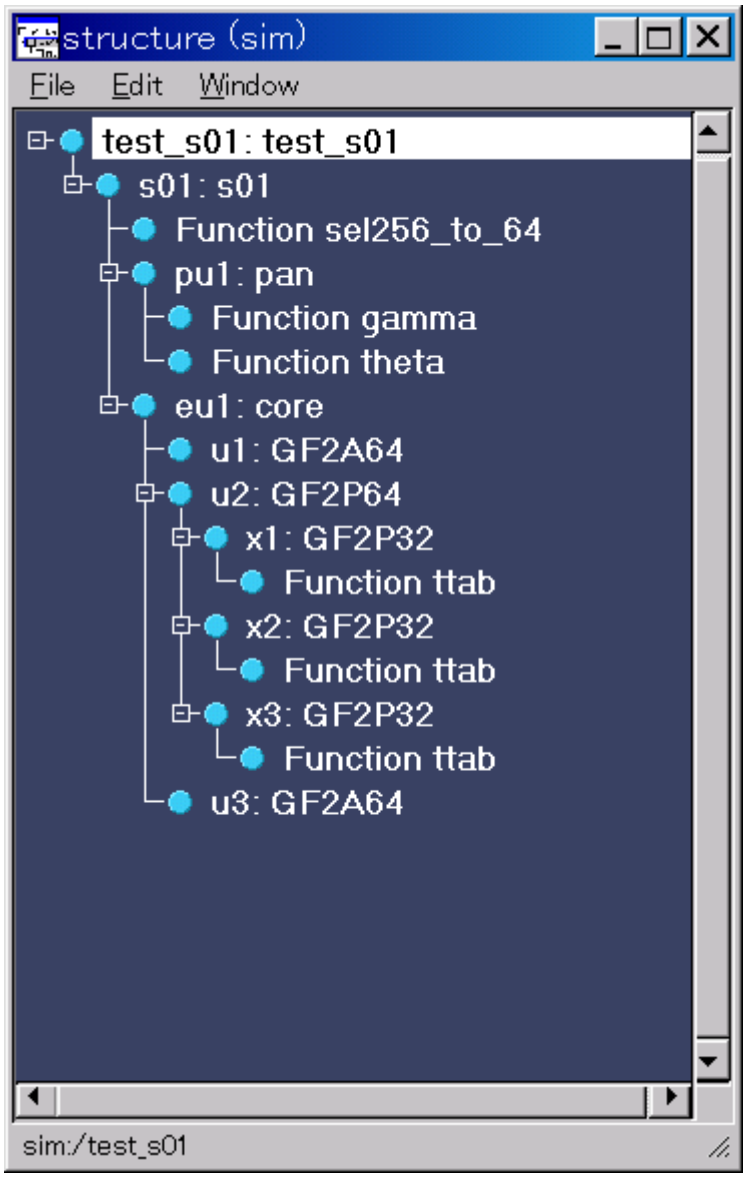


***_selは、(len_sel　key_sel　divp_sel　ainv_sel)

（３）　暗号/複合処理のデータ入出力シーケンス



（４）　構成



S01：S01　　デバイス最上位階層（入出力制御と、パラメータ保持）

pu1:pan　　乱数発生器

eu1:core　　暗号化/複合化器

A．　添付資料

「list-1」　：　論理合成の対象部分のみを、「03Call6.c」から抜粋して記載します。

―――　転載の許可を得ていないので、省略します　―――

「list-2」　：　「s01_top.v」, 「pan.v」, 「core.v」から抜粋して記載します。

```
module s01 ( clk, clrn, rst,
                mode,
                len_sel, key_sel, divp_sel, din_sel, ainv_sel, start,
                din, dout,
                din_req, dout_en);

        input                   clk;                    // clock
        input                   clrn;                   // async clear
        input                   rst;                    // sync clear
        input                   mode;                     // mode sel   0:enc / 1: dec
        input                   len_sel;                // length area select
        input                   key_sel;                // key area select
        input                   divp_sel;               // divpara area select
        input                   din_sel;                // data area select
        input                   ainv_sel;               // A_INV area select
        input                   start;                  // start
        input   [63:0]          din;                    // data input

        output  [63:0]          dout;                   // data output
        output                    din_req;              // data input req
        output                    dout_en;               // data output enable
＜中略＞
          /******************/
          /* PANAMA engine */
          /******************/
    pan       pu1( clk, clrn, pan_cken, pan_rst,
                    pan_mode, pan_in_null, pan_in, pan_out) ;
＜中略＞
    assign core_1st = din_reg;
          /******************/
          /* enc/dec core   */
          /******************/
    core      eu1( mode, core_1st, core_2nd, core_feed, core_a, core_b, core_fd);

/* param. feed update */
    always @(posedge clk or negedge clrn)
    begin
        if (clrn == 1'b0)
                    core_feed <= 64'b0;
        else if (pan_rst == 1'b1)
                    core_feed <= 64'b0;
            else if ( l1din_sel == 1'b1)
                    core_feed <= core_fd;
    end
―――　以下省略　―――
```

```
module core ( mode, core_1st, core_2nd, feed, core_a, core_b, core_fd);

        input                   mode;           // 0:enc       / 1:dec
        input   [63:0]          core_1st;       // plain       / cipher
        output  [63:0]          core_2nd;       // cipher      / plain
        input   [63:0]          feed;
        input   [63:0]          core_a;         // A           / A_INV
        input   [63:0]          core_b;
        output  [63:0]          core_fd;        // next feed data

        wire    [63:0]          x,y;

        wire    [63:0]          u1_2nd;
        wire    [63:0]          u3_2nd;

    assign u1_2nd   = ( mode == 1'b0 ) ? core_b : feed;
    assign u3_2nd   = ( mode == 1'b0 ) ? feed   : core_b;

    GF2A64  u1( core_1st, u1_2nd, x);
    GF2P64  u2( x, core_a, y);
    GF2A64  u3( y, u3_2nd, core_2nd);

    assign core_fd = ( mode == 1'b0 ) ? x       : y;

endmodule
```

```
module pan ( clk, clrn, cken, rst,
                mode, in_null, in,    out );

        input                   clk;                    // clock
        input                   clrn;                   // async clear
        input                   cken;                    // clock enable
        input                   rst;                    // sync clear

        input                   mode;                     // pan mode 0:push / 1:pull
        input                   in_null;                // input controle   0:valid / 1:NULL
        input   [255:0]       in;

        output  [255:0]       out;
```

＜中略＞

```
/*****************************************/
/*    GAMMA                                */
/*****************************************/
    function      [31:0]   gamma;
        input    [31:0]   in1,in2,in3;
        begin
                  gamma     = in1 ^ (in2 | ~in3);
        end
    endfunction


    assign   gamma_i0 = gamma(state0, state1, state2);
    assign   gamma_i1 = gamma(state1, state2, state3);
```

＜中略＞

```
    assign   gamma_i15 = gamma(state15, state16, state0);
    assign   gamma_i16 = gamma(state16, state0, state1);


/*****************************************/
/*    PI                                 */
/*****************************************/

    assign   pi_i0 = gamma_i0;
    assign   pi_i1 = {gamma_i7[30:0], gamma_i7[31]};
    assign   pi_i2 = {gamma_i14[28:0], gamma_i14[31:29]};
```

＜中略＞

```
    assign   pi_i16 = {gamma_i10[23:0], gamma_i10[31:24]};


/*****************************************/
/*    THETA                              */
/*****************************************/
    function      [31:0]   theta;
        input    [31:0]   in1,in2,in3;
        begin
                  theta     = in1 ^ in2 ^ in3;
        end
```

```
    endfunction

    assign   theta_i0 = theta(pi_i0, pi_i1, pi_i4);
    assign   theta_i1 = theta(pi_i1, pi_i2, pi_i5);
＜中略＞
    assign   theta_i16 = theta(pi_i16, pi_i0, pi_i3);

/*****************************************/
/*   L                                   */
/*****************************************/
    assign L = (mode == 1'b0) ? ( in & in_mask) : stage4;

/*****************************************/
/*   b                                   */
/*****************************************/
    assign b = stage16;
/*****************************************/
/*    SIGMA                               */
/*****************************************/

    assign   sigma_i0 = theta_i0 ^ 32'h00000001;
    assign   sigma_i1 = theta_i1 ^ L[255:224];
＜中略＞
    assign   sigma_i16 = theta_i16 ^ b[31:0];

/***************************/
/*    MAIN a REGS          */
/***************************/
    always @(posedge clk or negedge clrn)
    begin
        if (clrn == 1'b0) begin
                    state0 <= 32'b0;
                    state1 <= 32'b0;
＜中略＞
                    state16 <= 32'b0;
            end
            else if( cken == 1'b1) begin
                    state0 <= sigma_i0;
                    state1 <= sigma_i1;
＜中略＞
                    state16 <= sigma_i16;
                end
    end



/***************************/
/*    MAIn b REGS          */
/***************************/

    assign stage25_in = stage24 ^ { stage31[255-64:0],stage31[255:255-63]};
    assign stage0_in = stage31 ^ ( (mode == 1'b0) ? L: {state1, state2, state3, state4,
                                                        state5, state6, state7, state8});

    always @(posedge clk or negedge clrn)
    begin
```

```
        if (clrn == 1'b0) begin
                        stage0 <= 256'b0; stage1 <= 256'b0; stage2 <= 256'b0; stage3 <= 256'b0;
                        stage4 <= 256'b0; stage5 <= 256'b0; stage6 <= 256'b0; stage7 <= 256'b0;
＜中略＞
                        stage28<= 256'b0; stage29<= 256'b0; stage30<= 256'b0; stage31<= 256'b0;
        end
        else if (rst == 1'b1) begin
                        stage0 <= 256'b0; stage1 <= 256'b0; stage2 <= 256'b0; stage3 <= 256'b0;
                        stage4 <= 256'b0; stage5 <= 256'b0; stage6 <= 256'b0; stage7 <= 256'b0;
＜中略＞
                        stage28<= 256'b0; stage29<= 256'b0; stage30<= 256'b0; stage31<= 256'b0;
                end
                else if (cken == 1'b1) begin
                        stage31 <= stage30; stage30 <= stage29; stage29 <= stage28; stage28 <= stage27;
                        stage27 <= stage26; stage26 <= stage25; stage25 <= stage25_in; stage24 <= stage23;
＜中略＞
                        stage3   <= stage2; stage2 <= stage1; stage1 <= stage0; stage0 <= stage0_in;
                    end
    end


    always @(posedge clk or negedge clrn)
    begin
        if (clrn == 1'b0)
                        out <= 256'b0;
        else if (rst == 1'b1)
                        out <= 256'b0;
            else if (cken == 1'b1)
                        out <= ( in & in_mask) ^ { state9, state10, state11, state12,
                                                   state13, state14, state15, state16};
    end

endmodule
```

———　ここまで　———

「List-3」　：　「Synplify：s01_top.srr」から抜粋して記載します。

———　ここから始まり　———

Performance Summary
****

Clock	Requested Frequency	Estimated Frequency	Requested Period	Estimated Period	Slack
clk	1.0 MHz	18.8 MHz	1000.0	53.1	946.9

Resource Usage Report

Synplify is performing all technology mapping.

Design view:work.s01(verilog)
Selecting part ep20k1000ebc652-1

I/O ATOMs:　　140

Logic resources:　19811 ATOMs of 42240 (46%)
ATOM count by mode:
　normal:　　19250
　arithmetic:　537
　counter:　　22
　qfbk_counter: 2

ESBs:　　0　(0% of 160)
ATOMs using regout pin: 9873
　also using enable pin: 9802
　also using combout pin: 64
　with no input combinational logic: 208 (uses cell for routing)

ATOMs using combout pin: 6854
ATOMs using cascin pin: 2757

Number of Inputs on ATOMs: 91125
Number of Nets:　23273

———　ここまで　———