X2I マイグレーションガイドライン

株式会社マクニカ　アルティマカンパニー

Rev.2　2023/4

Co.Tomorrowing

MACNICA

はじめに

- Intel Agilex® 7 FPGA は 従来の FPGA に比べ、プロセスやアーキテクチャーの最適化を図ることで、消費電力を抑えつつ パフォーマンスを最大限に発揮することができる最先端の FPGA です。

- 本資料は、他社 FPGA (Xilinx® FPGA) 用に設計したデザインを Intel Agilex® 7 FPGA に置き換える手順と注意点を、実際の置き換えフローに沿ってまとめた資料です。本資料に従って置き換え作業を行うことによって、Intel® FPGA へのスムーズなデザインの移行が可能となります。

- 本資料では、新規に Quartus® Prime プロジェクトを作成し、RTL の変換、固有機能のブラックボックス化および制約ファイルの変換をステップ・バイ・ステップで行うことによって、Quartus® Prime 上でコンパイル可能なデザインを作成します。

- 置き換え先 FPGA は、Intel Agilex® 7 FPGA を想定して資料を作成していますが、他の Intel® FPGA デバイスに置き換えを行う場合も同様の方法で置き換えを行うことができます。

Agenda

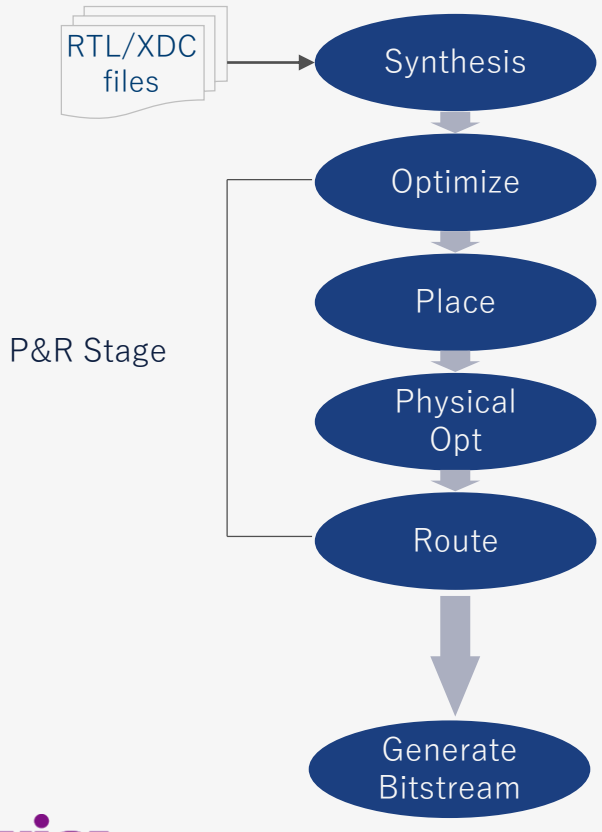
1. デザインフローについて
2. プロジェクトの置き換え
3. マクロ・IP の置き換え
4. XDC の置き換え
5. コンパイルレポートの確認
6. IP の Whitebox 化

1. デザインフローについて

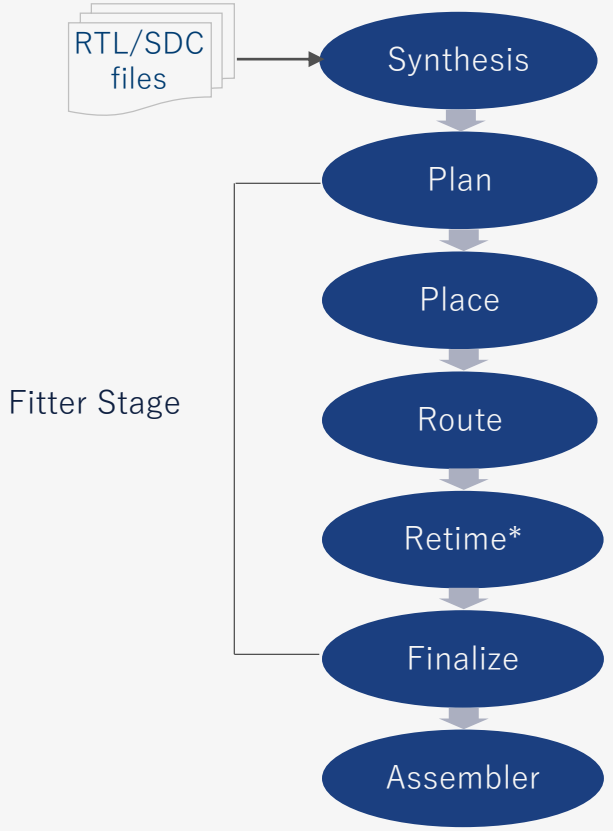
デザインフローについて

- Vivado® と Quartus® Prime のデザインフローは同等
  - Vivado®と Quartus® Prime の機能比較は Appendix を参照

Xilinx® Vivado®



Intel® Quartus® Prime

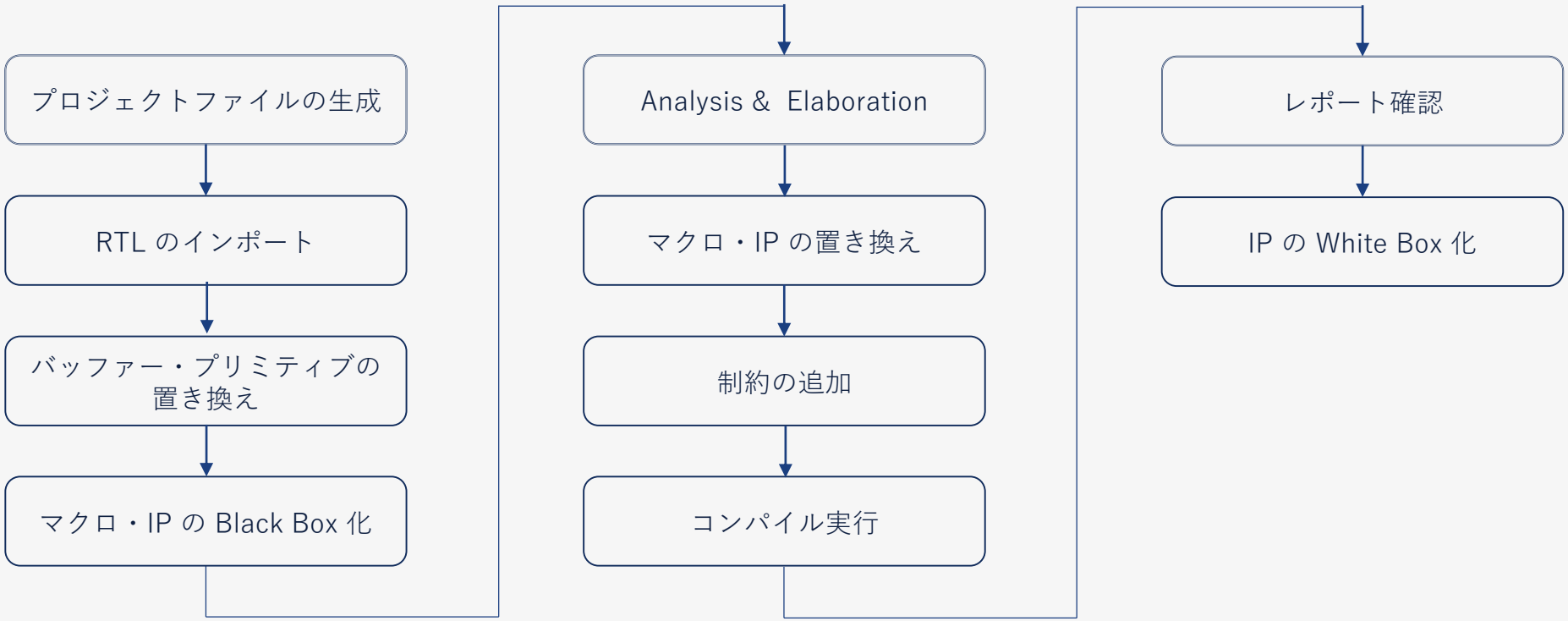


* Intel® Hyperflex™ FPGA devices

2. プロジェクトの置き換え

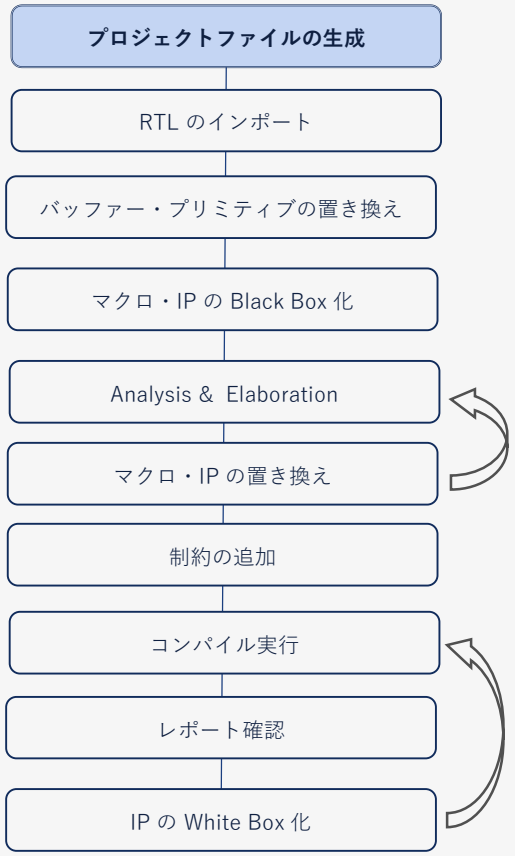
プロジェクトの置き換え手順 ～全体像～

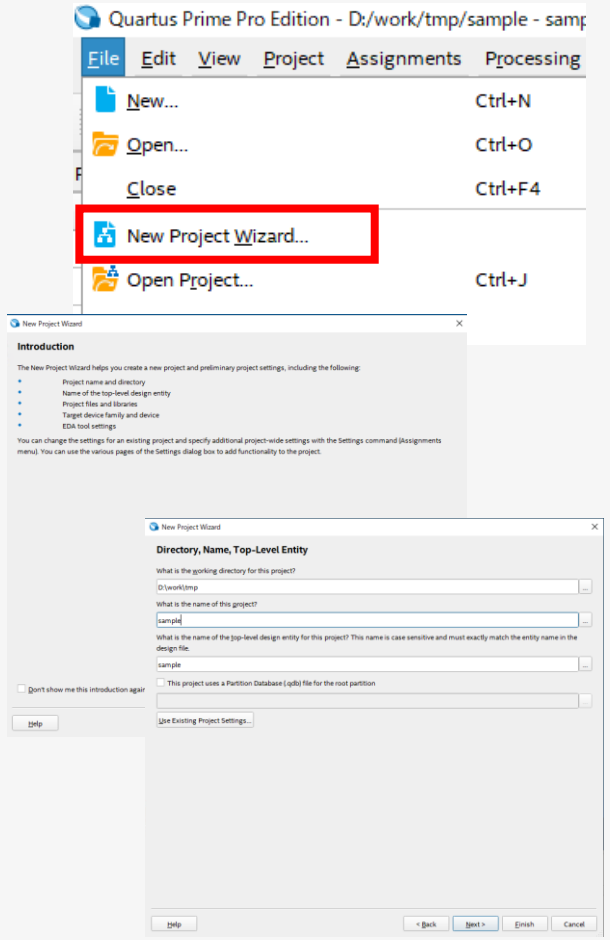
- プロジェクトの置き換えフロー
  - プロジェクトファイル形式 (**.xpr** vs **.qpf**) が異なるため、スクリプトでプロジェクトを自動で置き換えすることは困難
  - **新規にプロジェクトを作成**し、下記の手順で置き換えを行う



プロジェクトの置き換え手順１

- プロジェクトファイルの新規生成
  - ウィザード形式で簡単に作成
    - File メニュー → New Project Wizard
  - プロジェクト作成と同時に
    以下の設定も実施
    - 作業フォルダーの指定
    - デザインファイルの登録
    - デバイスの選択
    - EDA ツールの設定
  - 作業フォルダー、プロジェクト名以外は
    作成後でも変更可能

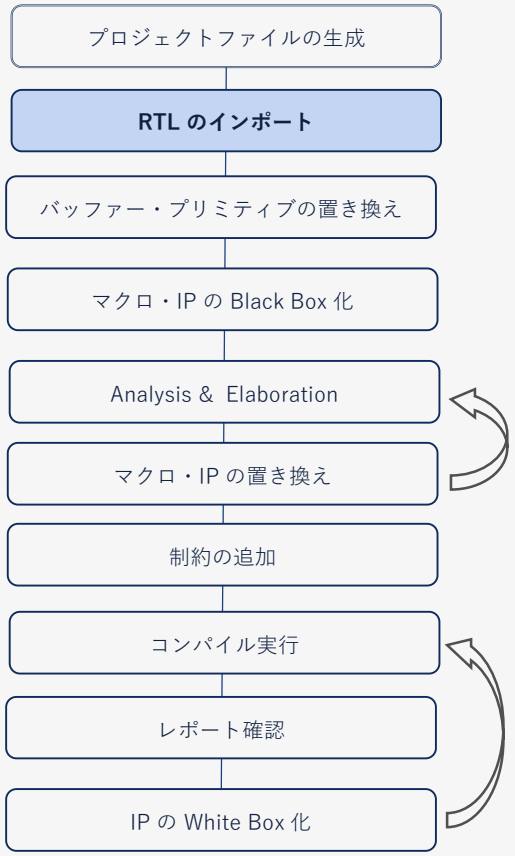


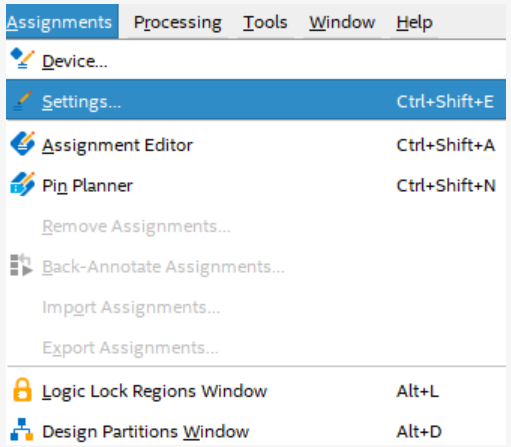


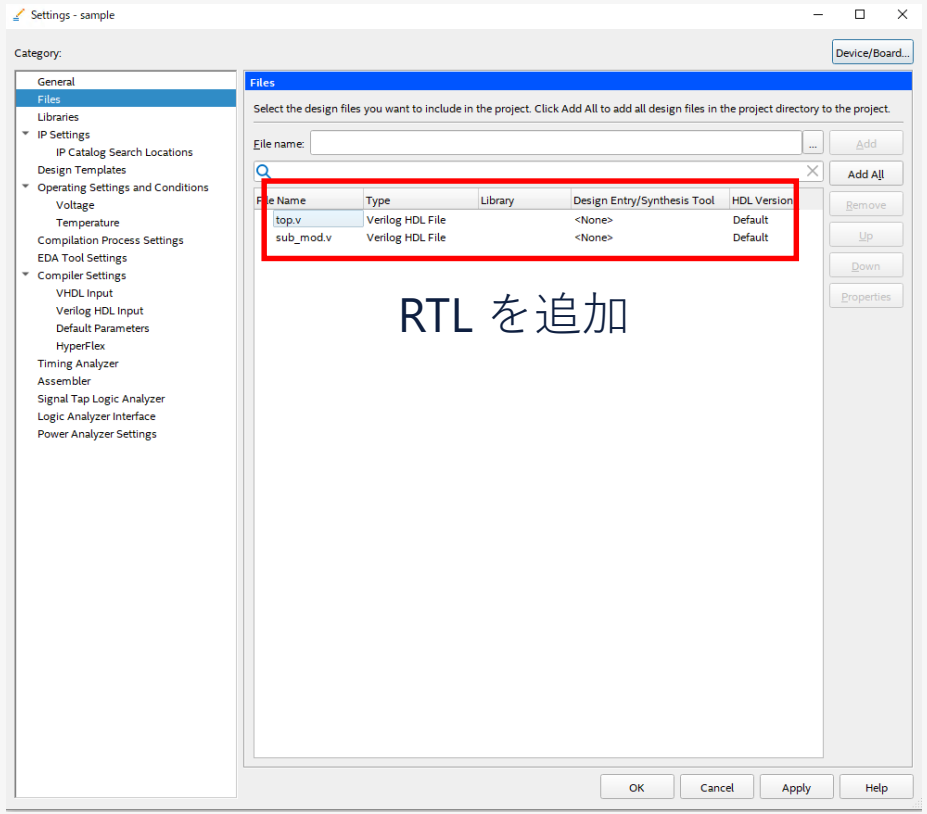
プロジェクトファイル作成詳細手順は[こちら]

プロジェクトの置き換え手順 2

- RTL のインポート
  - Assignments → Settings → Files に置き換え元の RTL を追加
    - Xilinx® 用のマクロ・IP はインポートしない





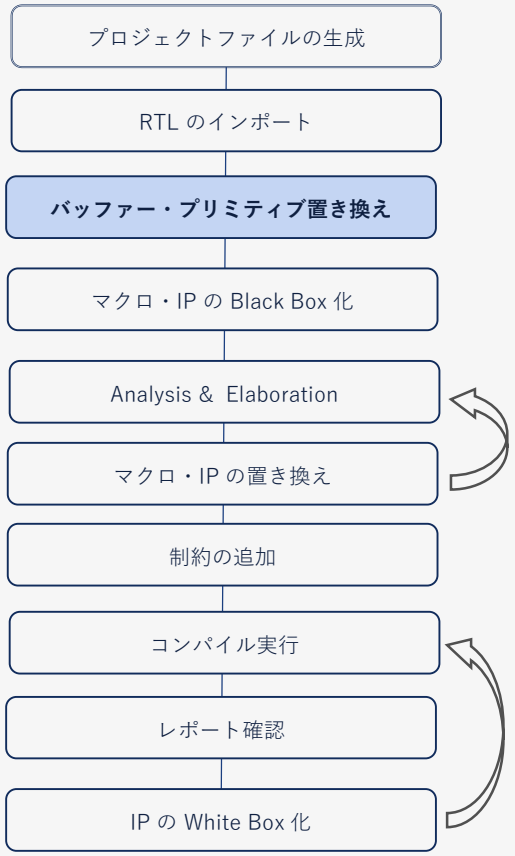


プロジェクトの置き換え手順３

- バッファー・プリミティブ の置き換え
  - Xilinx® プリミティブは、Quartus® Prime では未サポート
    - コンパイルするために、リンクのプリミティブ変換ファイルを使用する
      例：IBUFG, BUFG, BUFGCE

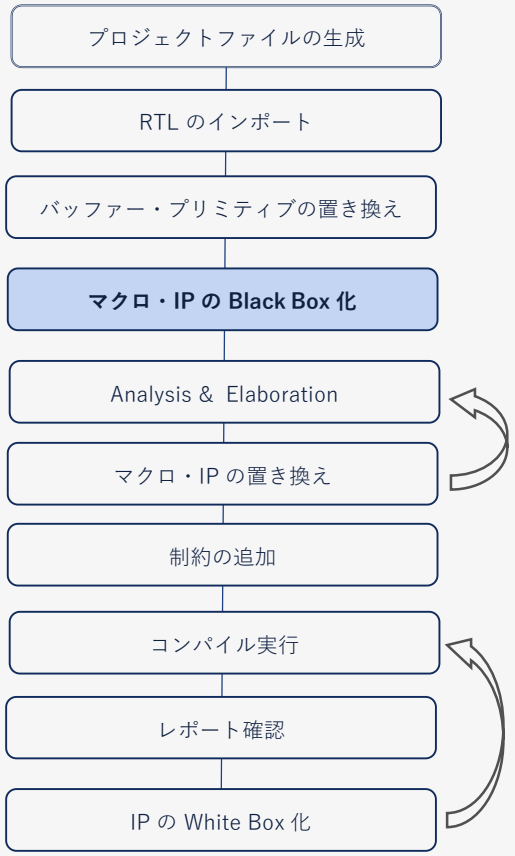
      ＊リンクにある “buf_conv” ファイルをプロジェクトに追加する
      Assignments → Settings → Files にファイルを選択し追加

    - プリミティブ変換ファイルにないプリミティブは 自作または削除する
      例：ISERDES, OSERDES　詳細は Appendix を参照



プロジェクトの置き換え手順 4

- IP・マクロの Black Box 化 (Verilog HDL 版)
  - Black Box をインスタンスするときはラッパーファイルを作成
  - 3rd ベンダーの論理合成ツールを使用するときはアトリビュートを使用
    - アトリビュート： /* synthesis syn_black_Box */;



Verilog HDL Black Box 記述例

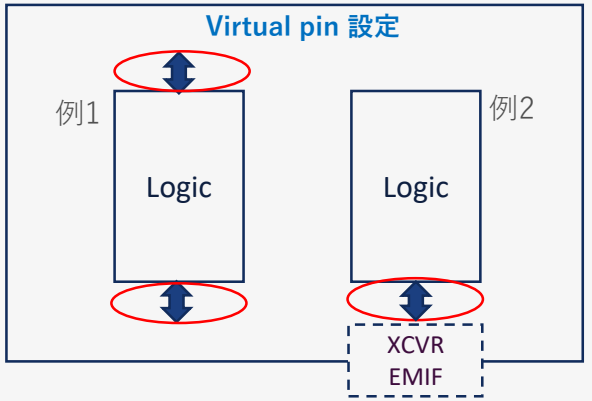
```
module top (clk, count);
        input clk;
        output[7:0] count;

        my_verilogIP verilogIP_inst (.clock (clk), .q (count));
endmodule

// Module declaration
// The following attribute is added to create a
// black Box for this module.

module my_verilogIP (clock, q)  /* synthesis syn_black_Box */;
        input clock;
        output[7:0] q;
endmodule
```

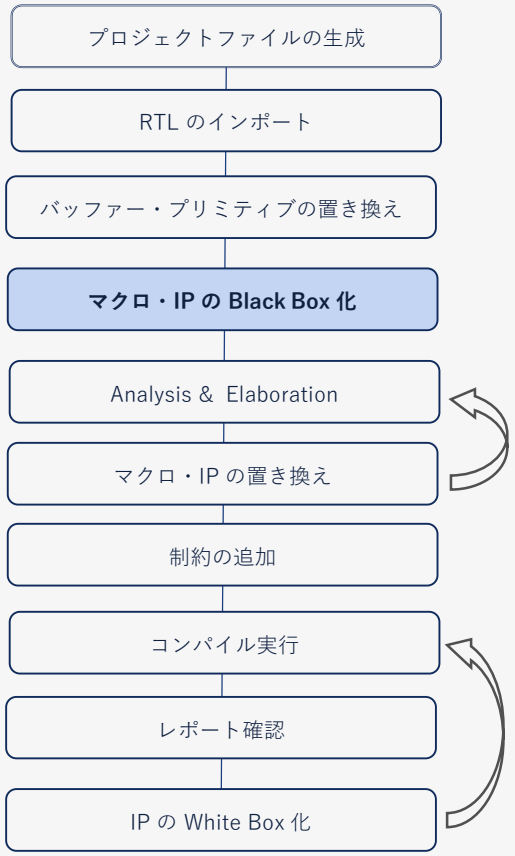
論理圧縮を防ぐために
アトリビュートを使用

※ VHDL の記述例は、Appendix を参照

プロジェクトの置き換え手順 4：未接続信号の取り扱い

- 未接続信号を Virtual Pin に設定
  - 例1：I/O ピン未接続の特定モジュール
    - 内部ロジックの I/O ポートを Virtual Pin に設定し 内部ロジックにマッピング
  - 例2：トランシーバー、EMIF IP マクロ接続の論理ポート
    - Hard IP マクロに繋がる 内部 I/O ポートを Virtual Pin として扱い 内部ロジックにマッピング





Virtual Pin とは

下位階層デザインの入力ピンと出力ピンを仮想ピンとして扱うオプション

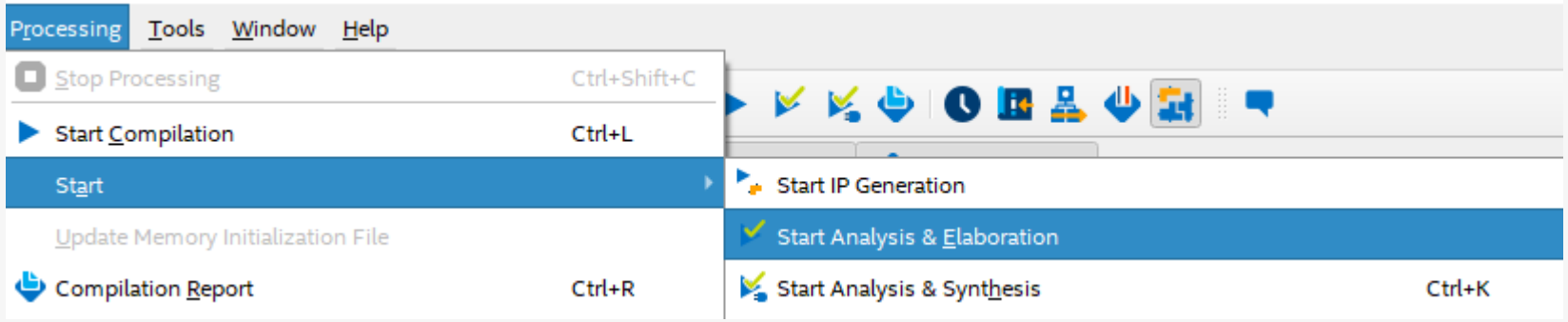
下位モジュールのポート数がターゲット・デバイスのピン数を超えてしまうような場合、コンパイル・エラーになるため下位階層の I/O ポートを仮想ピンとして指定することで、内部ロジック（LCELL）にマッピングされる
設定手順は下記を参照
https://www.macnica.co.jp/business/semiconductor/articles/intel/95581/

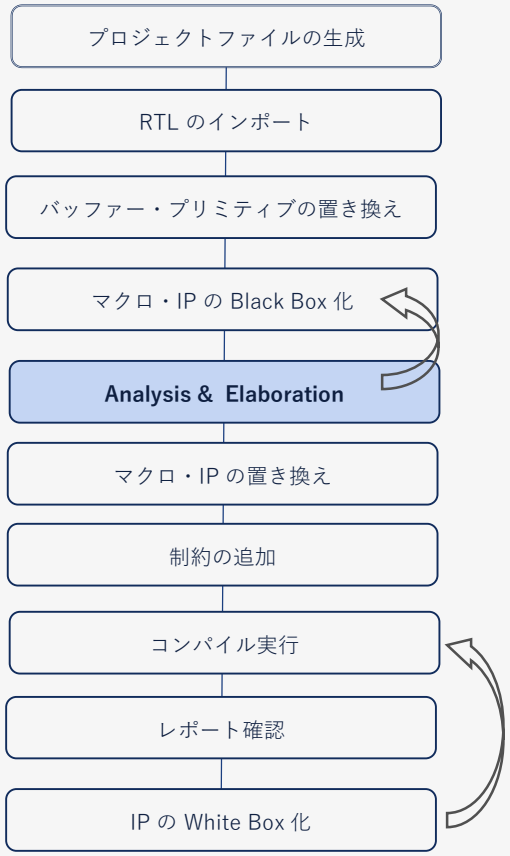
プロジェクトの置き換え手順５

- Analysis and Elaboration の実行
  - Analysis and Elaboration を 以下のメニューより実行し、**<u>エラー無く実行が完了すること</u>**を確認
  - 実行方法は2種類
    - ツールバーから実行

    - メニューから実行

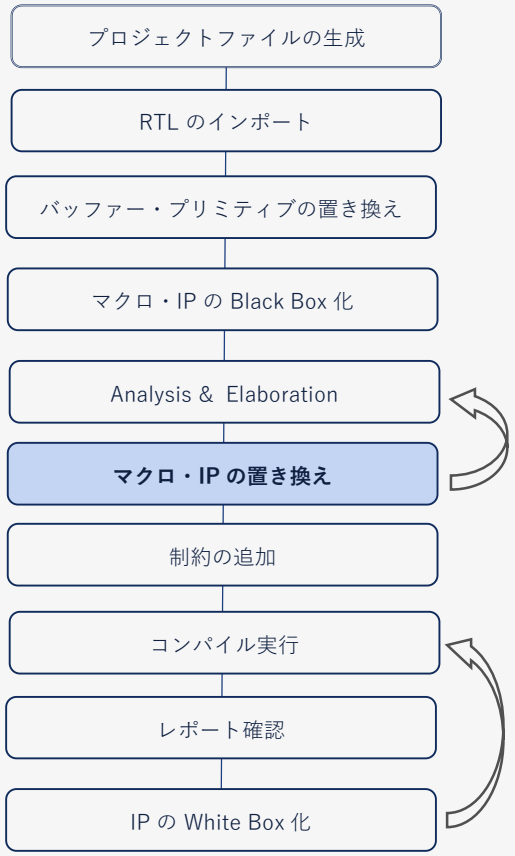
      Processing メニュー → Start → Start Analysis and Elaboration



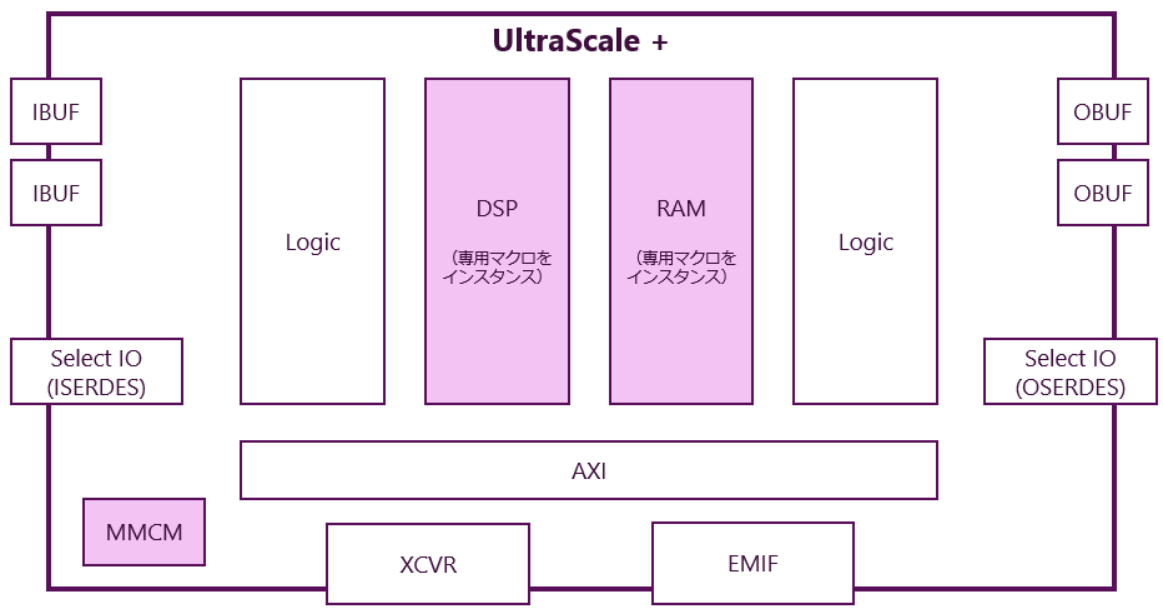


3. マクロ・IP の置き換え

マクロ・IP の置き換え

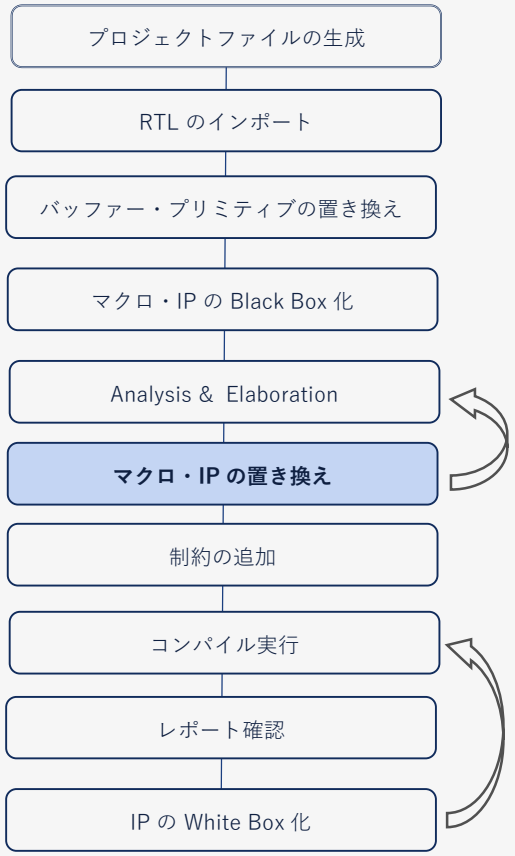


- Xilinx® Core Generator が生成した Memory、DSP、MMCM などのマクロ または IP は、Quartus® Prime の IP Catalog を使用し モジュールを生成し 置き換える必要がある
  - マクロ・IP の置き換えが終わった後、そのモジュールに対して「プロジェクトの置き換え手順 4」で設定した Black Box のアトリビュートは削除すること
  - ‘ifdef を使用し Intel®, Xilinx® の両方の記述を併記する方法もあるが、本資料では工数の少ない置き換えを採用している



マクロ・IP の置き換え：Memory Block

- Intel® FPGA と Xilinx® FPGA の 内蔵メモリーの違い



Memory Block/Block Memory

	Intel Agilex® 7 FPGA	Xilinx® Ultra Scale™ FPGA
Memory Size	20Kbit	36Kbit
Configuration	Single Port RAM	
	Simple Dual Port RAM	
	True Dual Port RAM	
	Simple Quad Port RAM	×
Clock Mode	Single	
	Input/Output	
	Read/Write	
初期化ファイル	HEX ファイル	COE ファイル

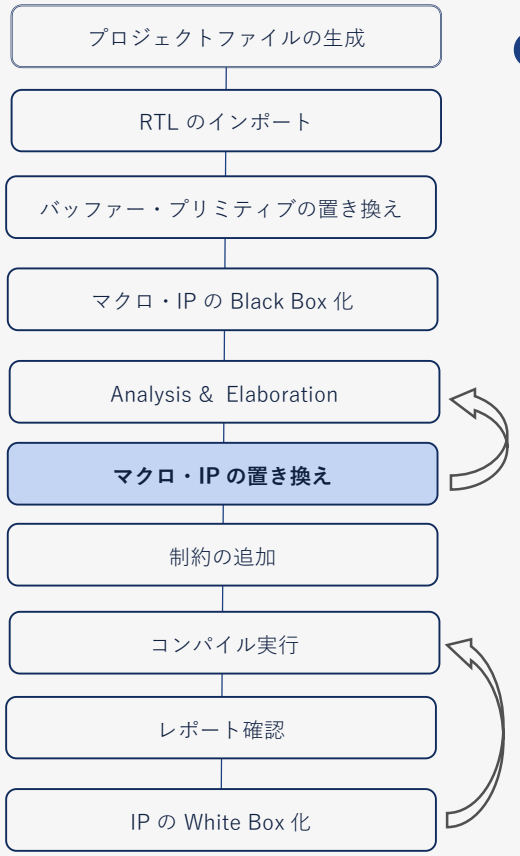
MLAB/Distributed RAM

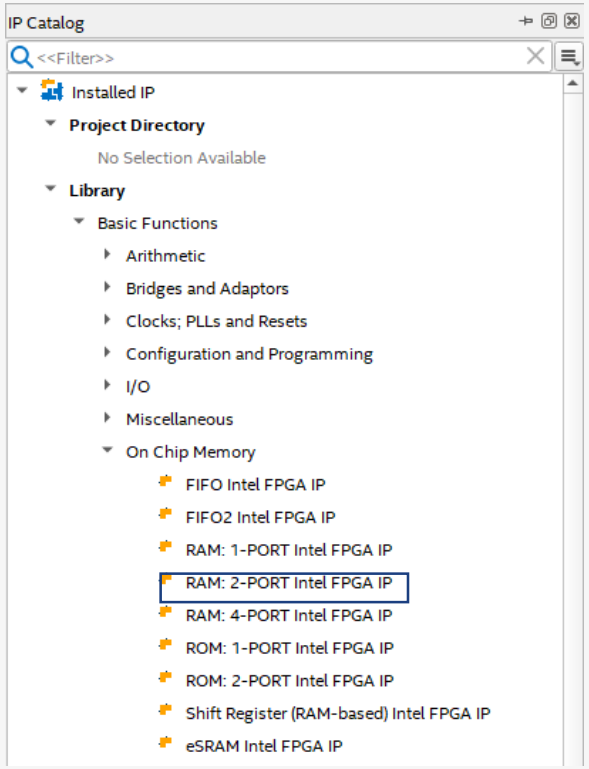
	Intel Agilex® 7 FPGA	Xilinx® Ultra Scale™ FPGA
Memory Size	640bit/LAB	512bit/CLB
Configuration	Single Port RAM	
	Simple Dual Port RAM	
	×	Dual Port 1Write 2Read
	ROM	
Clock	Sync Write	
	Sync Read	Async Read
初期化ファイル	HEXファイル	COEファイル

初期化ファイルの変換方法
https://community.intel.com/t5/FPGA-Wiki/Converting-Xilinx-RAM-initialization-coe-mif-format-to-Intel-PSG/ta-p/736050

マクロ・IP の置き換え：Memory Block

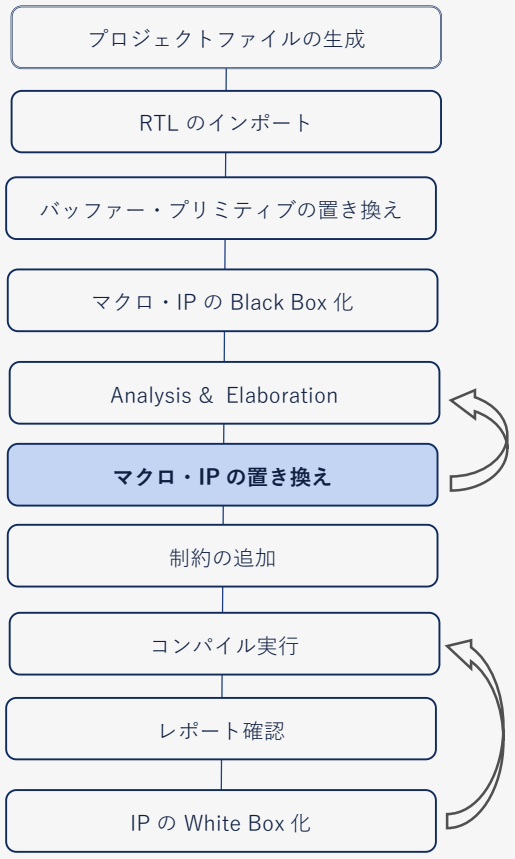
- Xilinx® Core Generator にて生成された 内蔵メモリーモジュールの置き換え
- Quartus® Prime の IP Catalog → On Chip Memory 内のメモリーマクロを活用する
  - 例：Xilinx® Simple Dual Port RAM => RAM: 2-PORT intel FPGA IP





How will you be using the dual port RAM?		
	With one read port and one write port	
Read/Write Ports		
	How wide should the 'q_a' output bus be?	16 bits
	How wide should the 'q_b' output bus be?	16 bits
What should the memory type be?		
	RAM Block Type	M20K
What clocking method do you want to use?		
	Dual clock: use separate 'read' and 'write' clock	
ECC Checking		
	Enable Error Correction Check (ECC)	On
	Enable ECC Pipeline Registers	On
Clock Enables		
	Use different clock enables for registers	On
	Use clock enable for write input registers	On
	Use clock enable for output registers	On

マクロ・IP の置き換え：Memory Block



Vivado® での Verilog HDL コード

```
module test (
    input                clka,
    input                ena,
    input       [0:0]    wea,
    input       [2:0]    addra,
    input       [15:0]   dina,
    input                clkb,
    input                enb,
    input       [2:0]    addrb,
    output      [15:0]   doubt,
    output               sbiterr,
    output               dbiterr,
    output      [2:0]    rdaddrecc
);
simple_dual port ip i1 (
   .clka                 (clka),
   .ena                  (ena),
   .wea                  (wea),
   .addra                (addra),
   .dina                 (dina),
   .clkb                 (clkb),
   .enb                  (enb),
   .addrb                (addrb),
   .doubt                (doutb),
   .sbiterr              (sbiterr),
   .dbiterr              (dbiterr),
   .rdaddrecc            (rdaddrecc)
);
endmodule
```

モジュールのポート名を書き換える

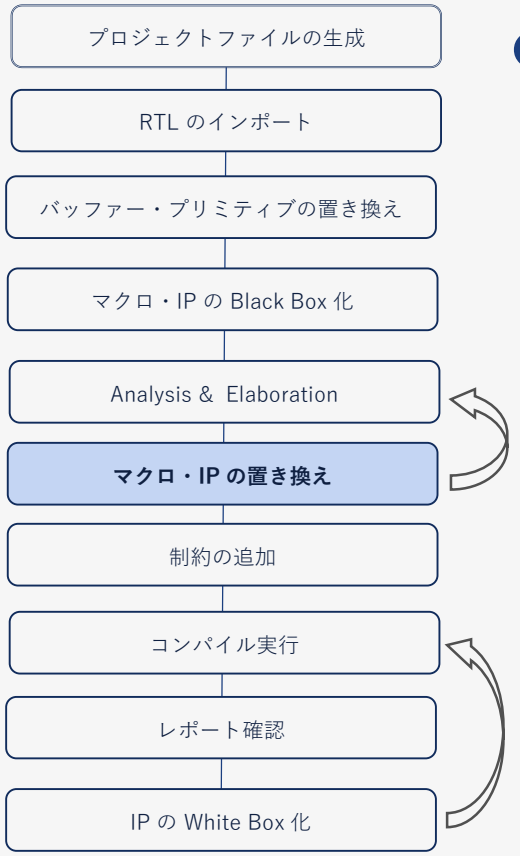
Quartus® Prime での Verilog HDL コード

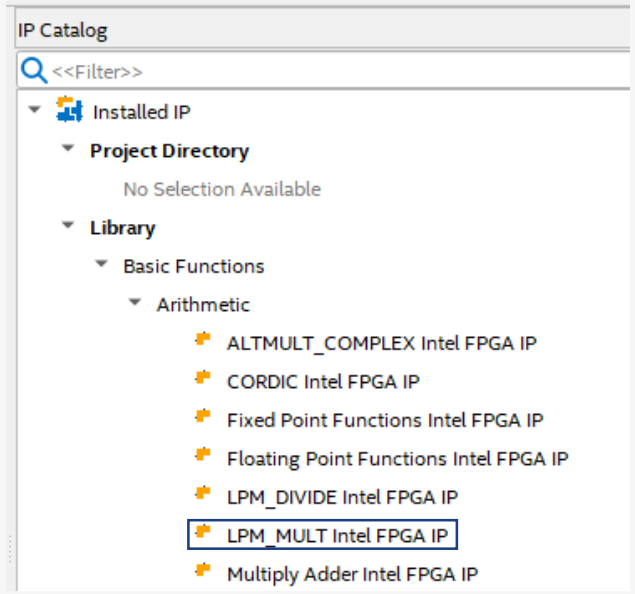
```
module test (
   input                clka,
    input               ena,
    input       [0:0]   wea,
    input       [2:0]   addra,
    input       [15:0]  dina,
    input               clkb,
    input               enb,
    input       [2:0]   addrb,
    output      [15:0]  doubt,
    output              sbiterr,
    output              dbiterr,
    output      [2:0]   rdaddrecc
);
simple_dual_port_ip i1(
   .wrclock             (clka),
   .wrclocken           (ena),
   .wren                (wea),
   .wraddress           (addra),
   .data                (dina),
   .rdclock             (clkb),
   .rdoutclocken        (regceb | enb),
   .rdaddress           (addrb),
   .q                   (doutb),
   .eccstatus           ({dbiterr, sbiterr})
);
endmodule
```

詳細は AN307 の 4.2.1.4. Memory Port Mapping を参照

マクロ・IP の置き換え：DSP

- Xilinx® Core Generator にて生成された DSP モジュールの置き換え
- Quartus® Prime の IP Catalog → Arithmetic 内の乗算器、乗算器+加算器のマクロを活用する
  - 例：Xilinx® Multiplier core => LPM_MULT intel FPGA IP

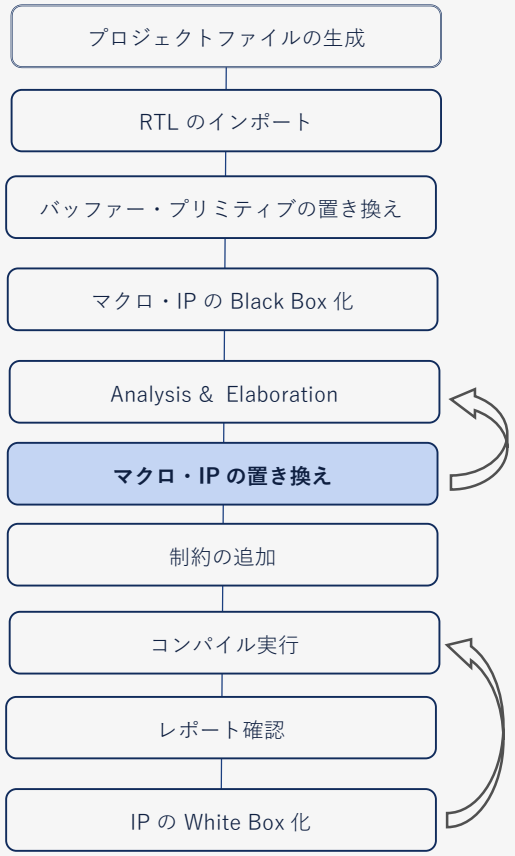




Feature	Xilinx® Multiplier Core Generator Module	Intel® FPGA LPM_MULT IP Core
Constant Coefficient	○	○
Signed and Unsigned Data	○	○
Configurable Pipeline Latency	○	○
Area versus Speed Trade-off	○	○
Asynchronous Clear	-	○
Synchronous Clear	○	○
Port A and Port B support different sign	○	- Considering to use ALTMULT_ADD

詳細は AN307 の 4.2.3. Converting Multipliers を参照

マクロ・IP の置き換え：DSP



Vivado® での Verilog HDL コード

```
module top(
    input               clk,
    input       [17:0]  a,
    input       [17:0]  b,
    input               ce,
    input               sclr,
    output      [35:0]  p
    );
mymult i1 (
    .CLK        (clk),
    .A          (a),
    .B          (b),
    .CE         (ce),
    .SCLR       (sclr),
    .P          (p)
    );
endmodule
```

モジュールのポート名を書き換える

Quartus® Prime での Verilog HDL コード

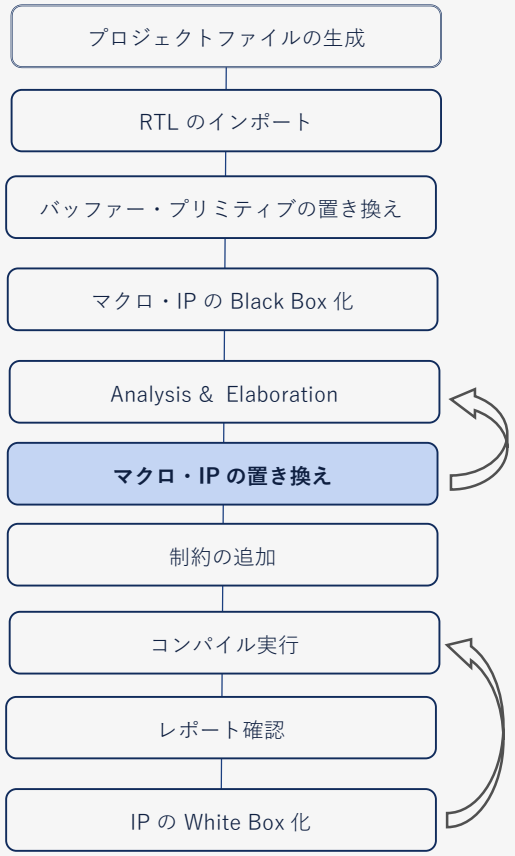
```
module test(
    input               clk,
    input       [17:0]  a,
    input       [17:0]  b,
    input               ce,
    input               sclr,
    output      [35:0]  p
    );
mymult i1 (
    .clock      (clk),
    .data       (a),
    .datab      (b),
    .clken      (ce),
    .sclr       (sclr),
    .result     (p)
    );
endmodule
```

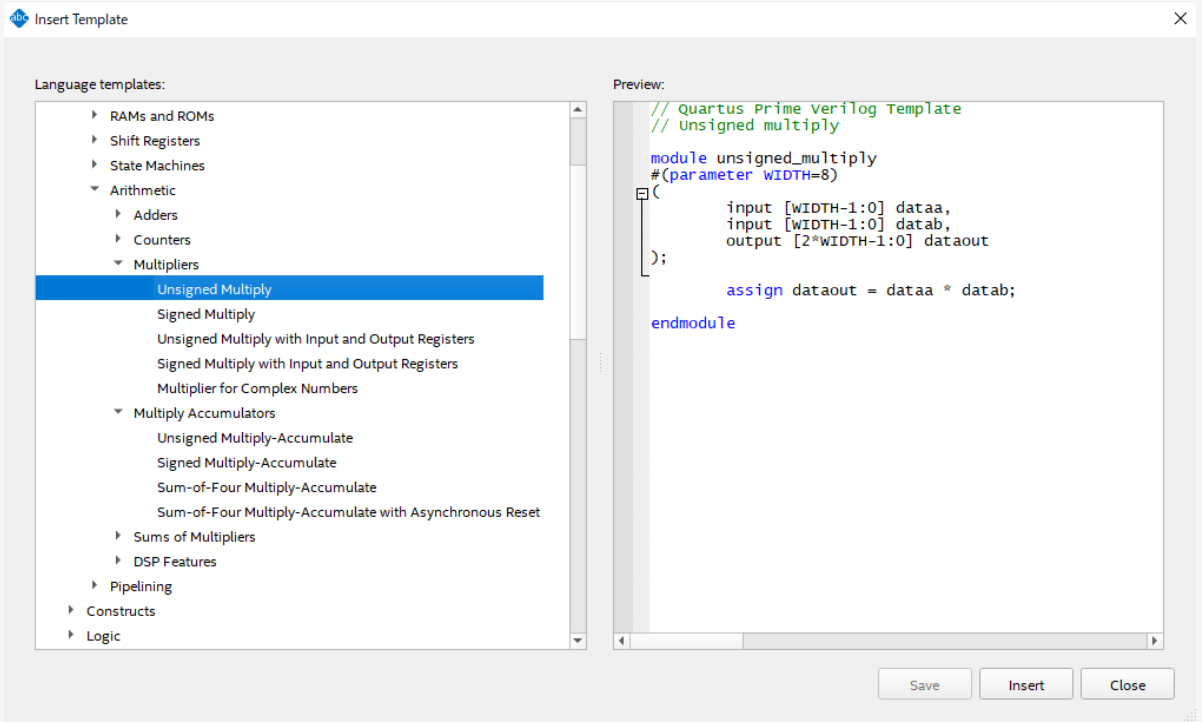
Port Mapping Between Xilinx® Multiplier Core and LPM_MULT IP Core

Xilinx® Multiplier Core Port	Intel® FPGA LPM_MULT IP Core Port	Description
A []	dataa []	Data Input Port A
B []	datab []	Data Input Port B
CLK []	clock	Clock Port
CE	clken	Clock Enable Port
SCLR	sclr	Synchronous Clear Port
N/A	aclr	Asynchronous Clear Port
P []	result []	Multiplication Result Port

マクロ・IP の置き換え：DSP/Memory Block の推論

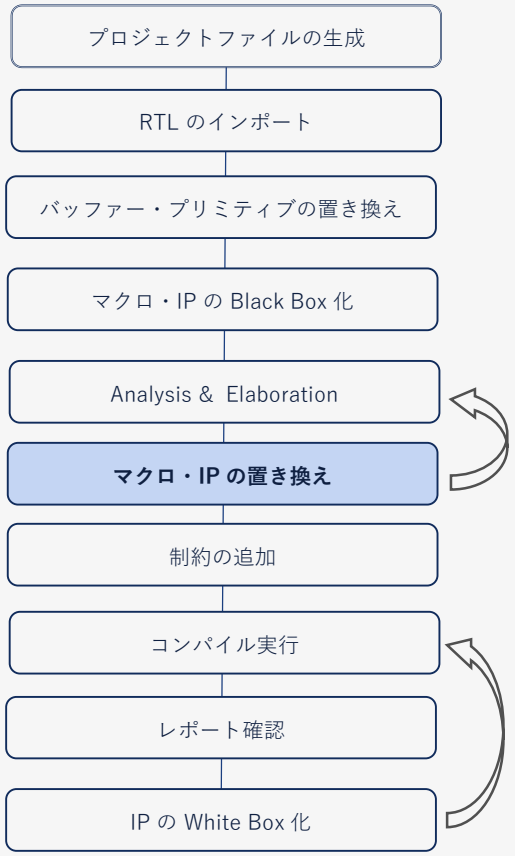
- Quartus® Prime の推論機能を使用して DPS/Memory Block を生成する場合
  - 推論結果が期待している DSP/Memory Block にアサインされているか確認
  - 期待するアサインになってない場合；
    - RTL 記述を修正
      テンプレートを使用すると便利。Edit → Insert Template
    - IP Catalog を使用

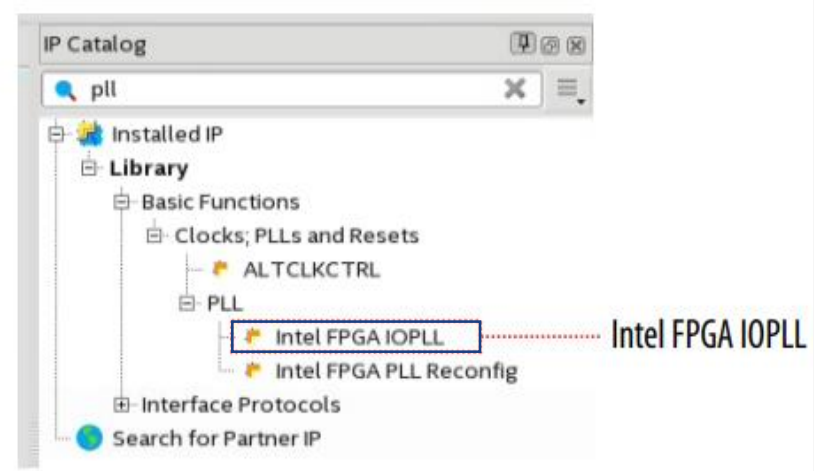




マクロ・IP の置き換え：MMCM

- Xilinx® Core Generator にて生成された MMCM モジュールの置き換え
- Quartus® Prime の IP Catalog → PLL 内の IOPLL マクロを活用する
  - 機能の差分はこちら
  - ポートの差分はこちら
  - Dynamic Phase Shift の差分はこちら
  - 例：入力クロック周波数：100MHz
    - 出力クロック周波数 1 ：Divided by 2 (50 MHz)
    - 出力クロック周波数 2 ：Multiply by 4 (400 MHz)

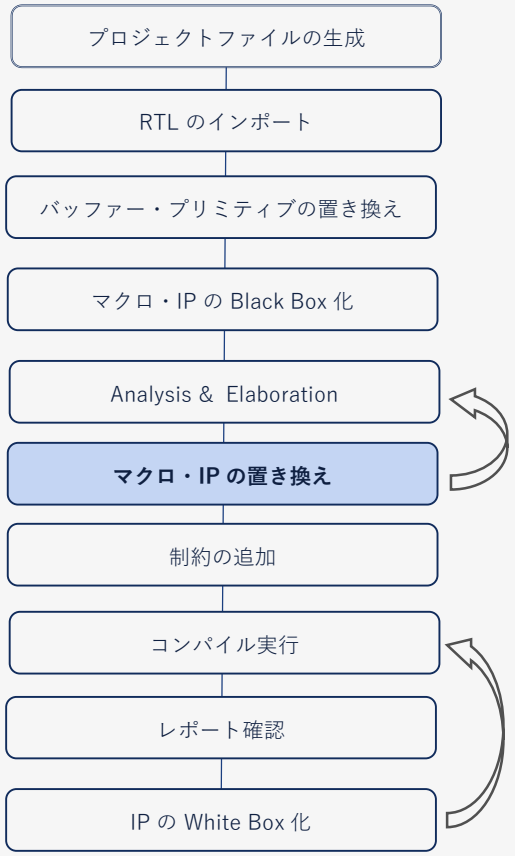




大分類	小分類	設定パラメーター
General	Reference Clock Frequency	100 MHz
output Clocks	Number of Clocks	2
outclock0	Clock Name	clk_out1
	Desired Frequency	50 MHz
outclock1	Clock Name	clk_out2
	Desired Frequency	400 MHz

マクロ・IP の置き換え：MMCM

- MMCM から Intel® PLL への置き換え (RTL 移植)



Vivado® での Verilog HDL コード

```
module top(
    input    reset,
    input    clk_in1,
    output   clk_out1,
    output   clk_out2,
    output   locked
    );
mymmcm i1(
   .reset                (reset),
   .clk_in1              (clk_in1),
   .clk_out1             (clk_out1),
   .clk_out2             (clk_out2),
   .locked               (locked)
   );
endmodule
```

モジュールのポート名を書き換える

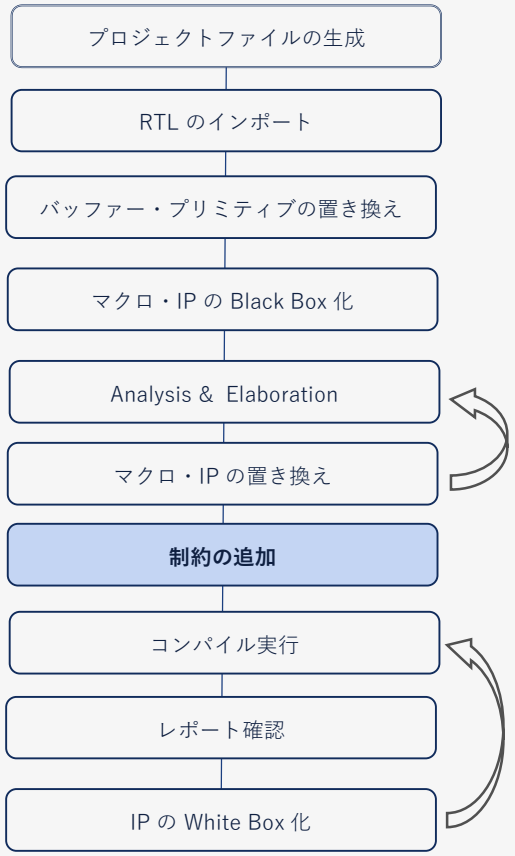
Quartus® Prime での Verilog HDL コード

```
module top(
    input    reset,
    input    clk_in1,
    output   clk_out1,
    output   clk_out2,
    output   locked
    );
mypll i1(
   .rst                  (reset),
   .refclk               (clk_in1),
   .outclk_0             (clk_out1),
   .outclk_1             (clk_out2),
   .locked               (locked)
   );
endmodule
```

4. XDC の置き換え

XDC からの置き換え

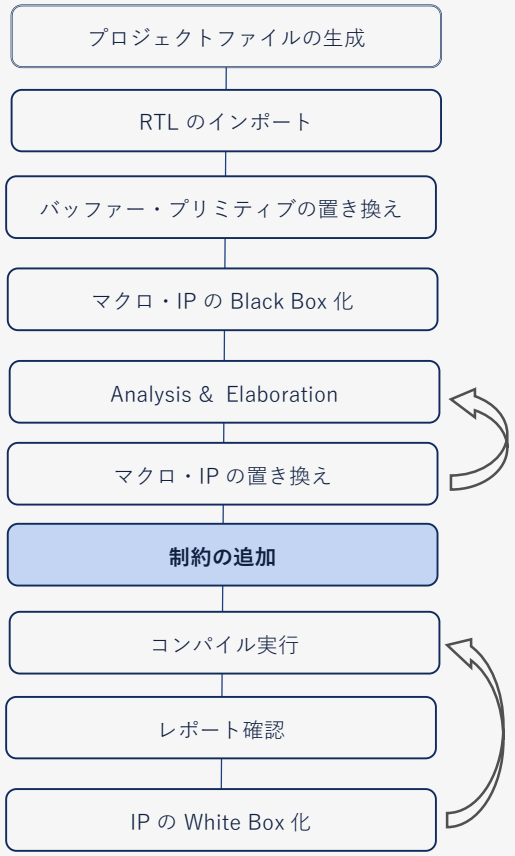
- Xilinx® Design Constraint (.xdc) には、各種アサイメント、タイミング制約を含む すべての制約が保存される



- .xdc の制約は、Intel® FPGAでは タイミング制約 (.sdc) とその他のアサイメント (.qsf) に分割して 制約をアサインする
  - .qsf にはタイミング制約以外のすべてアサイメントが保存される
    - Settings メニュー (デバイス設定)　　設定方法はこちら
    - Pin Planner (I/O ピン設定)　　設定方法はこちら
    - Assignment Editor (オプション設定)　　設定方法はこちら
  - .sdc にはタイミング制約が保存される
    - XDC タイミング制約をそのまま使用せず Quartus® Prime にて新たに設定し直すことを推奨

(参考)　SDC は Synopsys Design Constraint の略で ASIC を作成するときにも使用されている標準的なタイミング制約文法

アサイメント設定の移行



- I/O アサイメントについて ピン名以外の設定フォーマットが異なるため、すべて書き換えが必要
  - I/O Standard 記述の比較 (例)
    - 出力ピン “dout” に Differential SSTL-18 Class I を設定

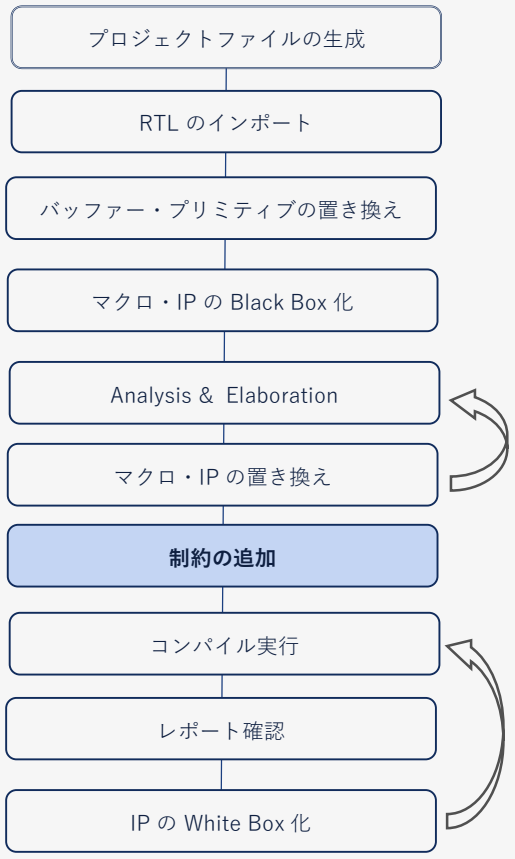
      XDC の記述
      - set_property IOSTANDARD SSTL18_I [get_ports dout];

      QSF の記述
      - set_instance_assignment -name IO_STANDARD "SSTL-18 CLASS I" -to dout

- XDC に含まれるその他の I/O アサイメント

XDC Constraint	Intel® FPGA Constraint @ QSF
DRIVE	CURRENT_STRENGTH_NEW
SLEW	SLEW_RATE
IOB	FAST_INPUT_REGISTER FAST_OUTPUT_REGISTER
IOSTANDARD	IO_STANDARD

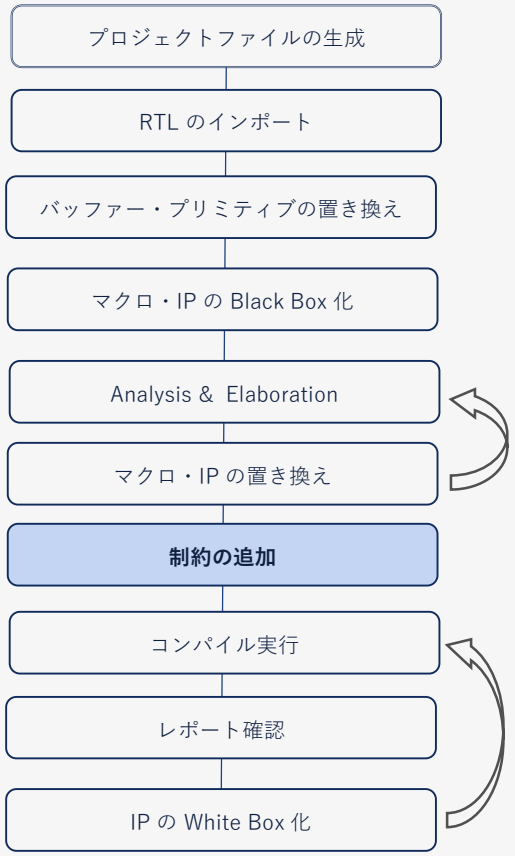
タイミング制約の移行



- 入力クロック制約
  - create_clock
    - Port の名称が同じ場合は XDC からの再利用が可能
    - 記述例) create_clock –period 50MHz [get_ports {CLK}]
- 内部生成クロック制約
  - create_generated_clock
    - create_generated_clock は pin や net の名称方法が異なるため XDC からの再利用は不可
- PLL の設定を自動生成
  - クロック名は “IOPLL Intel FPGA IP” の clock name 設定にて ユーザーが指定可能
- FPGA 内部の不確定要素 (ジッターなど) を自動生成
  - derive_clock_uncertainty
    - FPGA 内部の不確定要素を考慮し、より厳しい制約を追加
- その他のタイミング制約の記述方法や GUI での入力方法については下記の資料を参照
  - Quartus® はじめてガイド「Timing Analyzer によるタイミング制約の方法」

その他の主なタイミング制約

- XDC タイミング制約は Synopsys Design Constraint に基づいているため、基本的には XDC と SDC のタイミング制約に違いはない



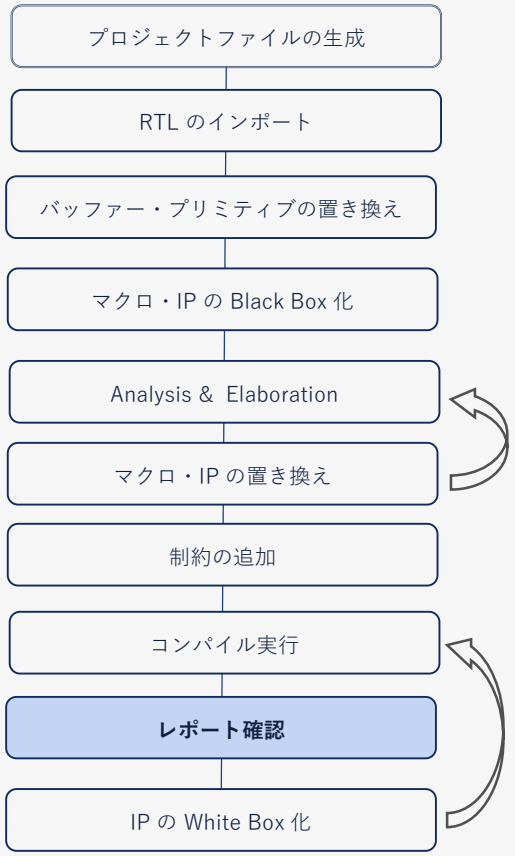
XDC	SDC	説明
create_clock create_generated_clock	create_clock create_generated_clock	クロック制約
NA	derive_pll_clocks	PLL の出力に対する制約を自動生成し設定 （インテル特有）※
NA	derive_clock_uncertainty	PLL ジッター、クロック・ツリー・ジッターなどを含む FPGA 内のクロック間の不確定要素を計算し設定（インテル特有）
set_input_delay		入力タイミング制約
set_output_delay		出力タイミング制約
set_max_delay		FPGA 内部遅延制約
set_min_delay		
set_clock_groups		クロック間解析をさせない非同期・排他クロック制約
set_false_path		配置配線とタイミグ検証においてタイミングの考慮から除外する制約

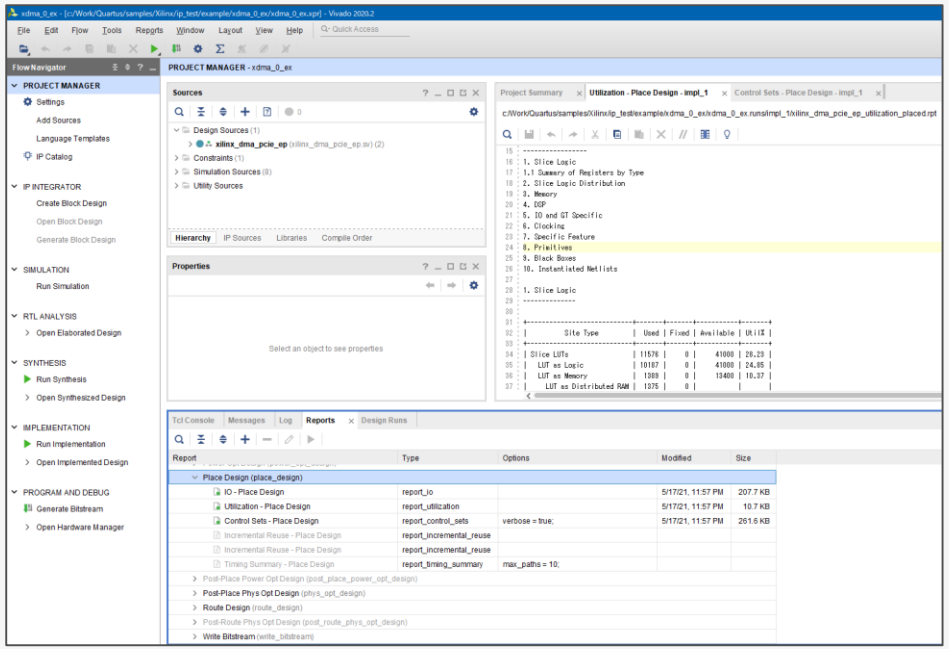
(※) Quartus® Prime Pro Edition では Arria® 10、Cyclone® 10 GX にてサポート

5. コンパイルレポートの確認

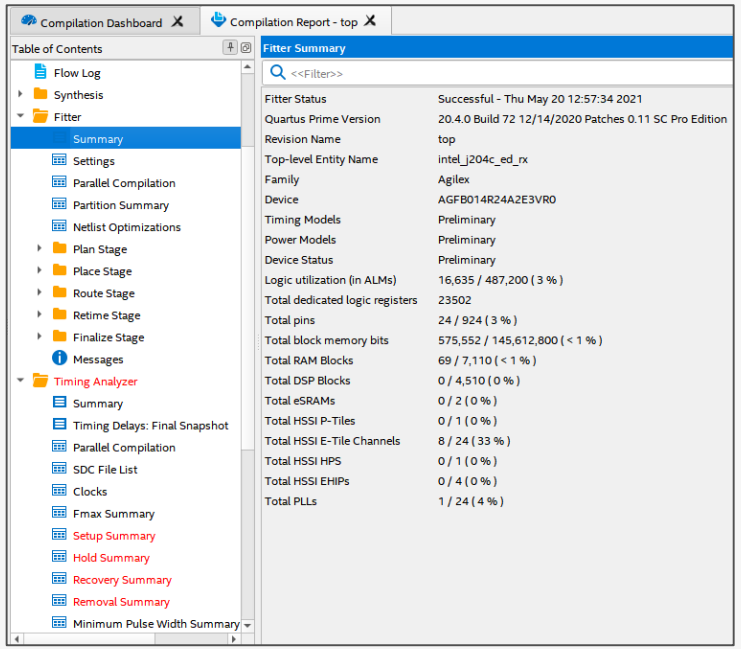
マイグレーション実現性の確認

- デバイスリソースの実現性確認
  - Vivado®　　: Implementation Report – Place Design
  - Quartus® Prime　: Fitter – Place Stages
- パフォーマンス（内部動作周波数）の実現性確認
  - Vivado®　　: Report Timing Summary
  - Quartus® Prime　: Timing Analyzer





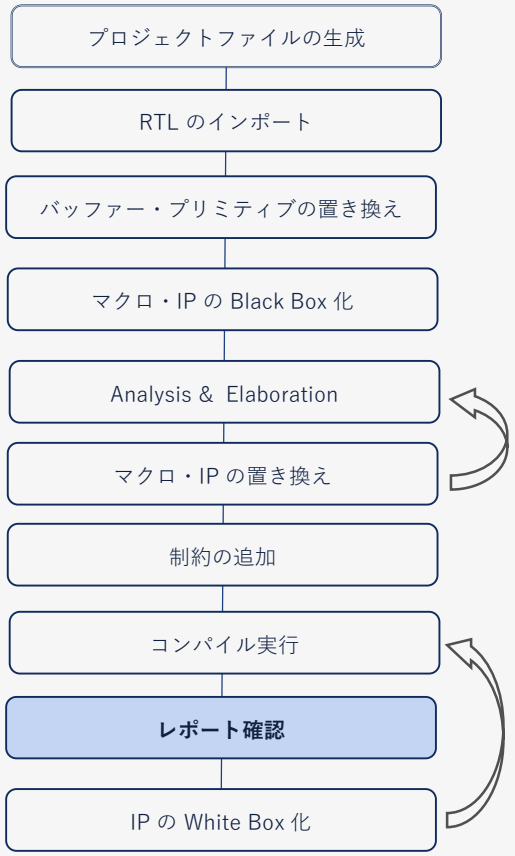
Vivado® コンパイルレポート



Quartus® Prime コンパイルレポート

リソース使用率の確認

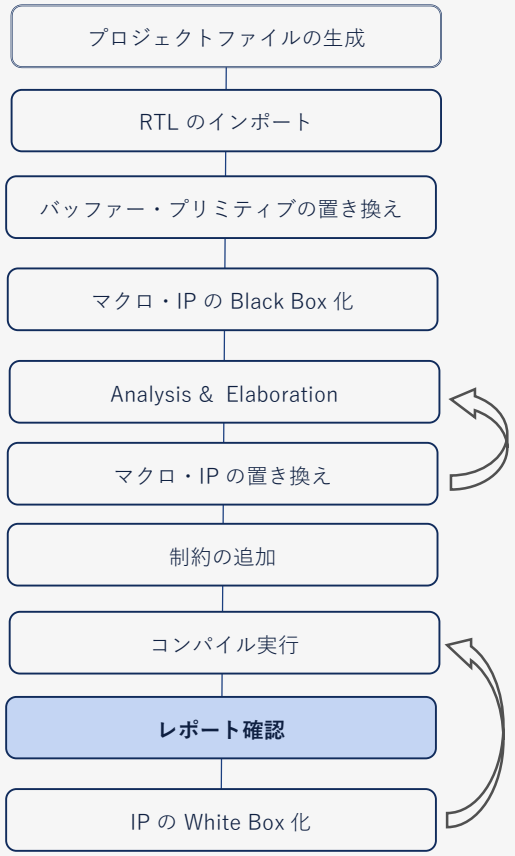
- Quartus® Prime の最適化によりリソースが削除されていないことを フィッター・レポートにて確認する
  - Quartus® Prime の Compilation Report にて確認
    - Fitter → Place Stage → Resource Usage
  - 併せて Quartus® Prime のコンパイル時の Warning も確認



Vivado® (Implementation Report Place Design)	Quartus® Prime (Fitter Place Design)
LUT as Logic	Combinational ALUT usage for logic
LUT as Memory	Total MLAB memory bits
Slice Registers	Dedicated Logic Registers
Block RAM	Total block memory bits
DSPs	DSP blocks
MMCM & PLL	IOPLLs & FPLLs

クロック制約の確認

- 制約を与えているクロックのタイミング要件をタイミング・レポートにて確認する
  - 元のクロック制約 (周波数・位相・デューティー比) が適切に移行できているか確認する
  - Quartus® Prime の Compilation Report にて確認
    - Timing Analyzer → Clocks



Vivado® : Clock Summary

Name	Waveform	Period (ns)	Frequency (MHz)
sys_clk	{0.000 5.000}	10.000	100.000
txoutclk_x0y0	{0.000 5.000}	10.000	100.000
clk_125mhz	{0.000 4.000}	8.000	125.000
clk_250mhz	{0.000 2.000}	4.000	250.000
mmcm_fb	{0.000 5.000}	10.000	100.000
userclk1	{0.000 8.000}	16.000	62.500

Quartus® Prime : Clocks

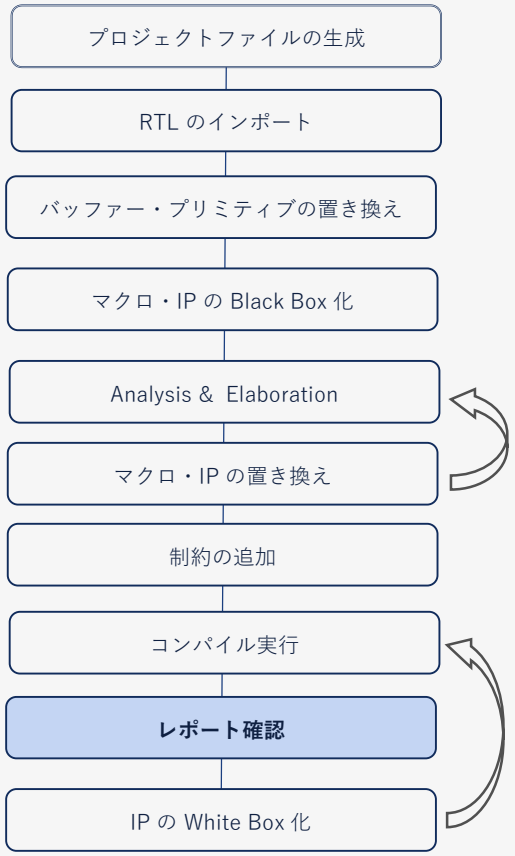
Clocks

Show: Visible | Hide | <<Filter>>

	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle
1	mgmt_clk	Base	10.000	100.0 MHz	0.000	5.000	
2	pll_ref_clk	Base	3.200	312.5 MHz	0.000	1.600	
3	seriallite_iii_streaming_ins...xcvr_native_insts[0]|avmmclk	Generated	10.000	100.0 MHz	0.000	5.000	50.00
4	seriallite_iii_streaming_inst|...r_native_insts[0]|rx_coreclkin	Generated	5.120	195.31 MHz	0.000	2.560	50.00
5	seriallite_iii_streaming_inst...vr_native_insts[0]|rx_pma_clk	Generated	5.120	195.31 MHz	0.000	2.560	50.00
6	seriallite_iii_streaming_inst|...r_native_insts[0]|tx_coreclkin	Generated	5.120	195.31 MHz	2.560	5.120	50.00
7	seriallite_iii_streaming_inst...vr_native_insts[0]|tx_pma_clk	Generated	5.120	195.31 MHz	2.560	5.120	50.00
8	seriallite_iii_streaming_ins...xcvr_native_insts[1]|avmmclk	Generated	10.000	100.0 MHz	0.000	5.000	50.00
9	seriallite_iii_streaming_inst|...r_native_insts[1]|rx_coreclkin	Generated	5.120	195.31 MHz	0.000	2.560	50.00

最大動作周波数の確認

- 最大動作周波数をタイミング・レポートにて確認する
  - Xilinx®　: Intra-Clock-Paths としてレポート
  - Intel®　: Fmax として Quartus® Prime の Compilation Report にて確認
    - Timing Analyzer → Fmax Summary



Vivado® : Intra-Clock-Paths

$$Fmax = \frac{1}{\left(Period(ns) - WNS(ns)\right)} \times 10^3$$

Intra-Clock Paths

Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	WPWS (ns)	TPWS (ns)	Failing Endpoints (TPWS)	Total Endpoints (TPWS)
sys_clk	rise - rise	8.828	0.000	0	7	rise - rise	0.172	0.000	0	7	4.220	0.000	0	13
txoutclk_x0y0											3.000	0.000	0	3
clk_125mhz	rise - rise	3.199	0.000	0	1152	rise - rise	0.062	0.000	0	1152	0.092	0.000	0	536
clk_250mhz											2.400	0.000	0	2
mmcm_fb											8.751	0.000	0	2
userclk1	rise - rise	5.656	0.000	0	41062	rise - rise	0.030	0.000	0	41062	0.549	0.000	0	15064

Quartus® Prime : Fmax Summary

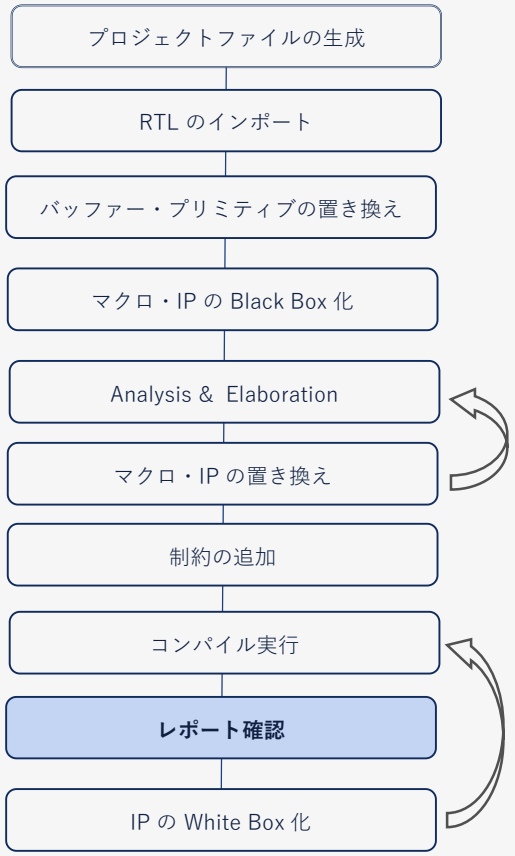
Fmax Summary

Show: Visible　Hide　<<Filter>>

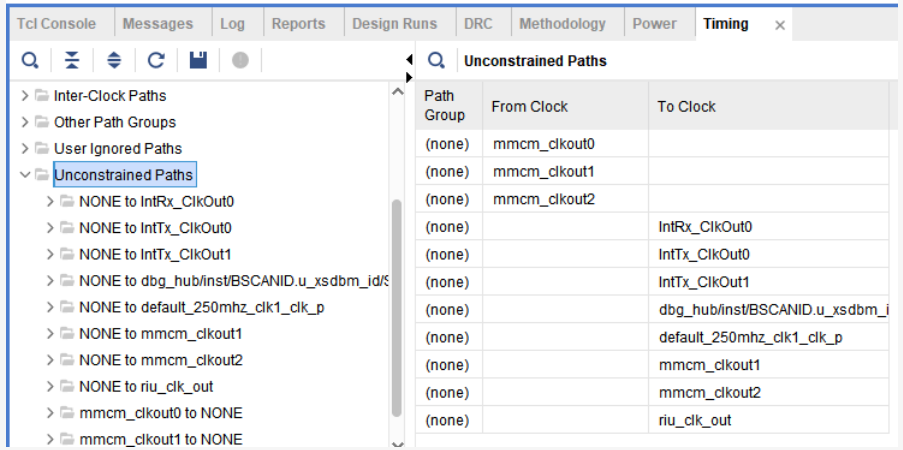
	Fmax	Restricted Fmax	Clock Name	Note	Worst-Case Operating Conditions
1	159.69 MHz	159.69 MHz	mgmt_clk		Slow 900mV 100C Model
2	211.15 MHz	211.15 MHz	seriallite_iii_streaming_inst|seri...ock_inst|altera_iopll_inst|outclk0		Slow 900mV 0C Model
3	217.77 MHz	217.77 MHz	seriallite_iii_streaming_inst|seri...ock_inst|altera_iopll_inst|outclk0		Slow 900mV 100C Model
4	297.35 MHz	297.35 MHz	seriallite_iii_streaming_inst|se..._xcvr_native_insts[0]|tx_pma_clk		Slow 900mV 100C Model
5	309.79 MHz	309.79 MHz	seriallite_iii_streaming_inst|se..._xcvr_native_insts[0]|rx_pma_clk		Slow 900mV 100C Model
6	336.36 MHz	336.36 MHz	~ALTERA_CLKUSR~		Slow 900mV 0C Model

未制約のクロックの確認

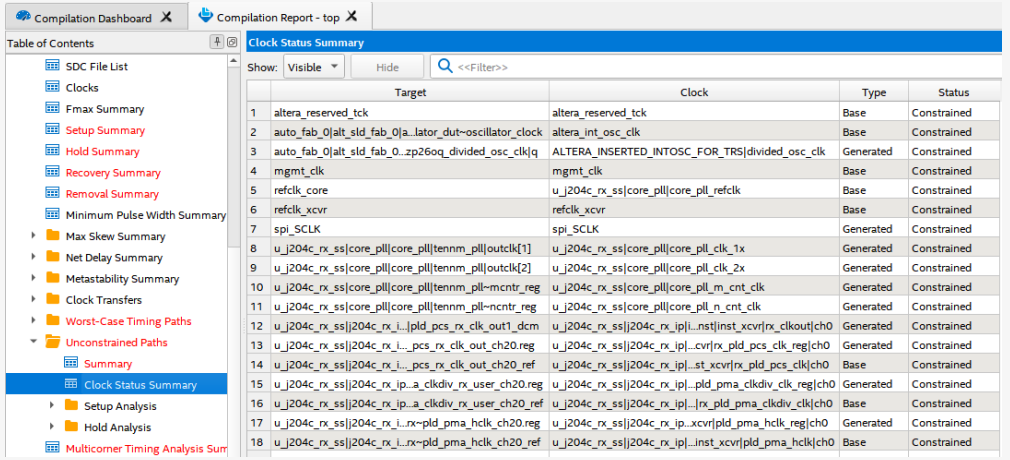
- 未制約のクロックが無い事をタイミング・レポートにて確認する
  - Quartus® Prime の Compilation Report にて確認
    - Timing Analyzer → Unconstrainted Paths



Vivado® : Unconstrainted Paths

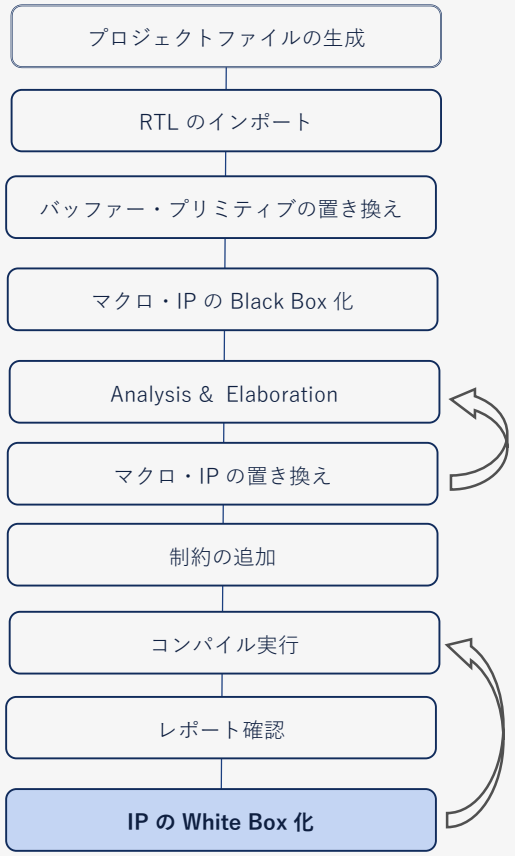


Quartus® Prime : Unconstrainted Paths



6. IP の Whitebox 化

ベンダー特有の IP の置き換え



- Xilinx® IP から Intel® IP への移行は IP の機能、IP のポートなど相違点が多いため、新規生成する必要がある
- Hard IP
  - EMIF、PCI Express
  - EMIF、PCI Express などは、デザイン & デバック・ガイドライン資料を参考にする
- Soft IP
  - IP の回路規模は 各 IP のユーザーガイドなどを参考にする
- Vivado® IP Integrator
  - Vivado® IP Integrator を使用したデザインは、Platform Designer を活用して生成を行う
    - Local Bus：AXI, Avalon-MM, Avalon-ST

最後に

- 本資料では、他社 FPGA (Xilinx® FPGA) 用に設計したデザインを Intel Agilex® 7 FPGA に置き換える手順と注意点を 具体的に紹介しました。

- 本資料を参考に 置き換え作業をステップ・バイ・ステップで行うことによって、Quartus® Prime 上でコンパイル可能なデザインを作成することができます。

- 本資料をご活用いただくことで Intel® FPGA へのスムーズなデザインの移行が可能ですので、是非お役立てください。

Co.Tomorrowing

MACNICA

Appendix

Agenda

- FPGA/CPLD 開発ツール比較
- 開発ツール機能比較
  - 開発ツール機能比較表
  - Platform Designer の紹介
  - デバックツール関連の紹介
- バッファー・プリミティブの置き換え
  - バッファー・プリミティブの置き換え例
- ISERDES / OSERDES の置き換え
  - High Speed Select IO Wizard
- IDDR / ODDR の置き換え
- プロジェクト置き換え手順 4 (VHDL 版)
- プロジェクトファイルの違い (Quaruts® Prime vs Vivado®)
- MMCM と PLL の機能比較
  - MMCM と PLL のポート比較
  - MMCM と Dynamic Phase Shift のポート比較

FPGA/CPLD 開発ツール比較

FPGA/CPLD 開発ツール比較 (Xilinx® vs Intel®)

	Xilinx®	Intel®
FPGA/CPLD 開発	Vivado® HLx Design Edition (有償) 7 Series, UltraScale™, UltraScale+™, Versal™	Intel® Quartus® Prime Pro (有償) Intel Agilex® 7, Stratix® 10, Arria® 10, Cyclone® 10 GX
	ISE Design Suite (有償) Spartan®-6, Virtex®-6, CoolRunner™,	Intel® Quartus® Prime Standard (有償) Stratix® IV/V, Arria® series, Cyclone® IV/V, Cyclone® 10 LP, MAX® series
	Vivado® HL WebPACK (無償) デバイス限定版	Intel® Quartus® Prime Lite (無償) Arria® II, Cyclone® IV/V, Cyclone® 10 LP, MAX® Series
OpenCL	SD Accel Environment	Intel® FPGA SDK for OpenCL™
HLS	Vivado® HLS	Intel® HLS Compiler
DSP	System Generator for DSP/Model Composer	DSP Builder for Intel® FPGAs
SoC	Xilinx Software Development Kit (XSDK)	Intel® SoC FPGA Embedded Development Suite
Embedded-Processor	Xilinx Software Development Kit (XSDK)	Nios® II Embedded Development System (EDS)

開発ツール機能比較　1/5

カテゴリ	Vivado® HLx Design Edition	Intel® Quartus® Prime
プロジェクト作成	New Project	New Project Wizard
Design Entry	IP Catalog	IP Catalog / IP Parameter Editor
	Vivado® IP Integrator	**Platform Designer** (System Integration Tool)
	IP Packager	**Platform Designer** (Component Editor)
	System Generator	DSP Builder
	OpenCL	OpenCL

■**Platform Designer**

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpp-platform-designer.pdf
https://www.macnica.co.jp/business/semiconductor/articles/intel/119949/

開発ツール機能比較　2/5

カテゴリ	Vivado® HLx Design Edition	Intel® Quartus® Prime
IO 制約	Device, Physical & Timing Constraints windows Xilinx Design Constraint (XDC)	Assignment Editor
Timing 制約		Synopsys Design Constraints（SDC）
Pin 設定/IO 設定		Pin Planner
		Interface Planner(Pro Edition only)
Design Processing/コンパイル	Integrated Synthesis engine	Integrated Synthesis engine
デザイン評価 & デバッグ	Schematic Windows （Elaborated）	**RTL Viewer**
	Schematic Windows （Synthesized） Schematic Windows （Implemented）	**Technology Map Viewers**（Post-Mapping） **Technology Map Viewers**（Post-Fitting）
		State Machine Viewer
		Fast Forward Viewer (Post-Fitting)*

■**RTL Viewer/ Technology Map Viewers**
https://www.intel.com/content/www/us/en/docs/programmable/683641/23-1/rtl-viewer-overview.html

開発ツール機能比較　3/5

カテゴリ	Vivado® HLx Design Edition	Intel® Quartus® Prime
消費電力解析	Xilinx Power Estimator (XPE) Report Power	Early Power Estimation (EPE) **Intel® FPGA Power and Thermal Calculator (PTC)**
シミュレーション	Vivado® Simulator Third-Party Simulation Tools	Questa® – Intel® FPGA Starter Edition Third-Party Simulation Tools
ハードウェア ベリフィケーション	Hardware Manager	System Console

■消費電力 および 熱計算 (PTC)
https://www.intel.com/content/www/us/en/docs/programmable/683445/23-1/faq.html

開発ツール機能比較　4/5

カテゴリ	Vivado® HLx Design Edition	Intel® Quartus® Prime
ハードウェア ベリフィケーション	Hardware Manager	System Console
	Integrated Logic Analyzer (ILA) and System ILA IP	**Signal Tap Logic Analyzer**
	Xilinx® Virtual Input Output (VIO)	In-System Sources and Probes
	JTAG-to-AXI Master	System Console
	IBERT IP and Serial I/O Analyzer Tool	**Transceiver Toolkit**
	Memory Calibration Debug Tool	**EMIF Debug Toolkit** EMIF Debug GUI
	Remote Debug using Xilinx Virtual Cable (XVC)	Remote Debug using existing TCP/IP connection
		Signal Probe In-System Memory Content Editor Logic Analyzer Interface (LAI)

■デバッグ ツール
https://www.intel.com/content/www/us/en/docs/programmable/683819/22-4/faq.html

開発ツール機能比較　5/5

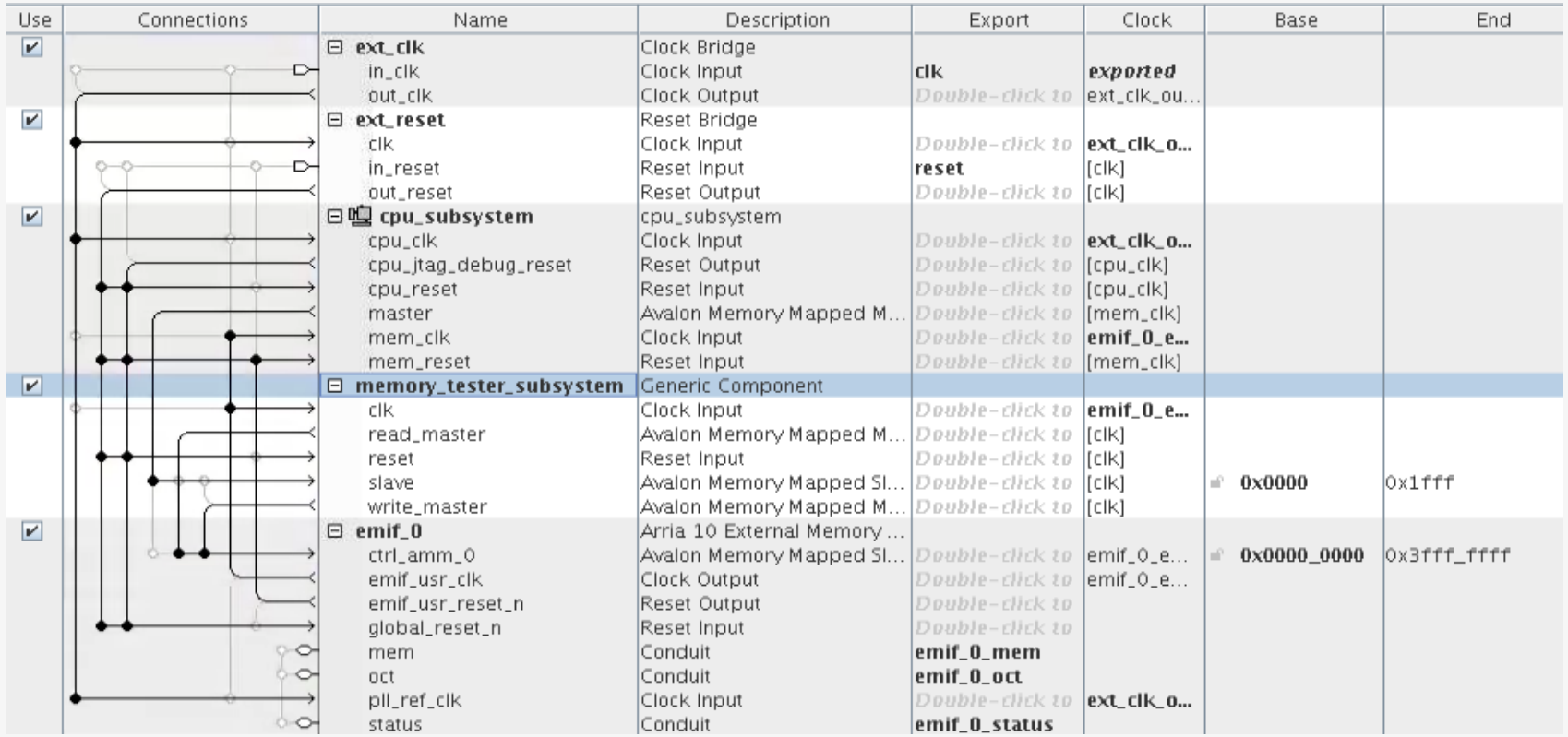
カテゴリ	Vivado® HLx Design Edition	Intel® Quartus® Prime
デザイン最適化	Physical Optimization	Physical Optimization
		Hyper-Aware Design Flow
生産性を向上させるためのテクニック	Incremental Compile	Incremental Optimization
	Hierarchical Design	Block-Based Design Flows
	Run Strategies (Strategy 変更のみ)	Design Space Explorer II (DSE) （最適化 Option & Seed 変更）

■デザインの最適化

https://www.intel.com/content/www/us/en/docs/programmable/683641/23-1/faq.html

Platform Designer の紹介

- プラットフォーム・デザイナー サポート
  - https://www.intel.com/content/www/us/en/docs/programmable/683609/23-1/faq.html

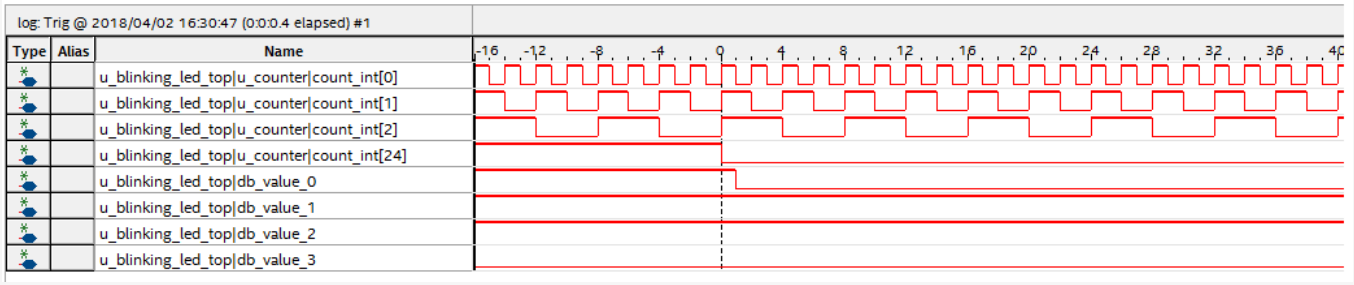


Platform Designer

GUI をベースとしたシステム統合ツールでコンポーネント間を
クリック 1つで接続できる設計ツール

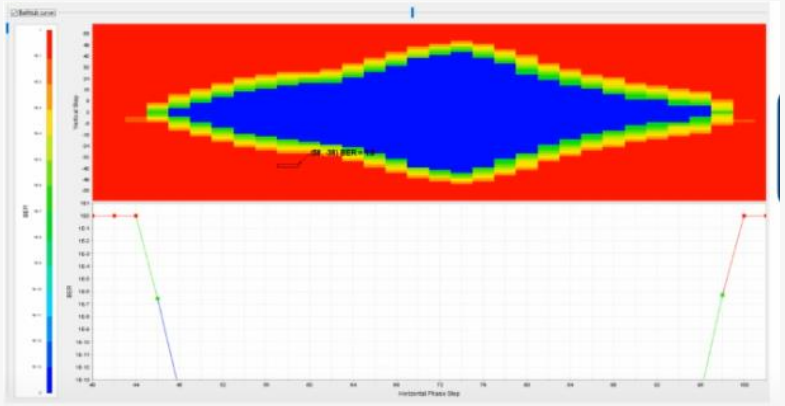
デバッグツール関連の紹介

- オンチップ･デバッグを支援するリソース
  - https://www.intel.com/content/www/us/en/docs/programmable/683819/22-4/faq.html



Signal Tap Logic Analyzer

FPGA にインプリメントしたユーザー回路の内部信号をモニタリング可能なロジアナ機能



Transceiver Toolkit (Eye Viewer)

トランシーバーチャネルのデバッグが可能

EMIF Debug Toolkit

外部メモリーインターフェイス用デバッグツール

温度・電圧変動によるタイミングマージンが確認可能

https://www.macnica.co.jp/business/semiconductor/articles/Arria10_EMIF_Implement_and_FTA_Guideline_Rev3.2.pdf

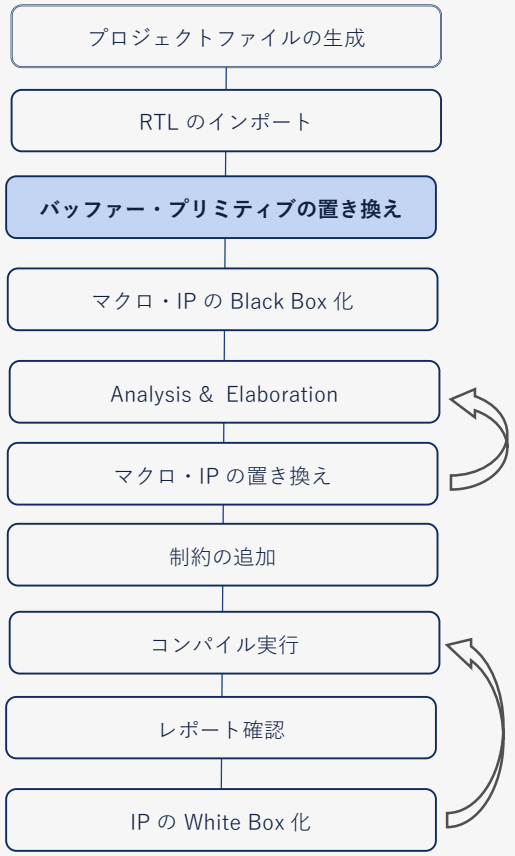
バッファー・プリミティブの置き換え

バッファー・プリミティブの置き換え

Xilinx® プリミティブ	説明	Intel® FPGA 等価	変換方法
IBUF	シングル入力バッファー	wire / signal を割り当て	HDL
OBUF	シングル出力バッファー	wire / signal を割り当て	
BUFG	グローバルクロック・バッファー	wire / signal とグローバル信号を割り当て	HDL Assignment Editor
IBUFG_<I/O 規格>	各 I/O 規格の入力グローバル・バッファー	wire / signal と I/O 規格とグローバル信号を割り当て	
IBUF_<I/O 規格>	各 I/O 規格の入力バッファー	wire / signal と I/O 規格を割り当て	
IOBUF_<I/O 規格>	各 I/O 規格の双方向バッファー	wire / signal と I/O 規格を割り当て	
OBUFG_<I/O 規格>	各 I/O 規格の出力グローバル・バッファー	wire / signal と I/O 規格を割り当て	
OBUF_<I/O 規格>	各 I/O 規格の出力バッファー	wire / signal と I/O 規格を割り当て	
IBUFDS, OBUFDS	差動 I/O バッファー	wire / signal と I/O 規格を割り当て	

バッファー・プリミティブの置き換え

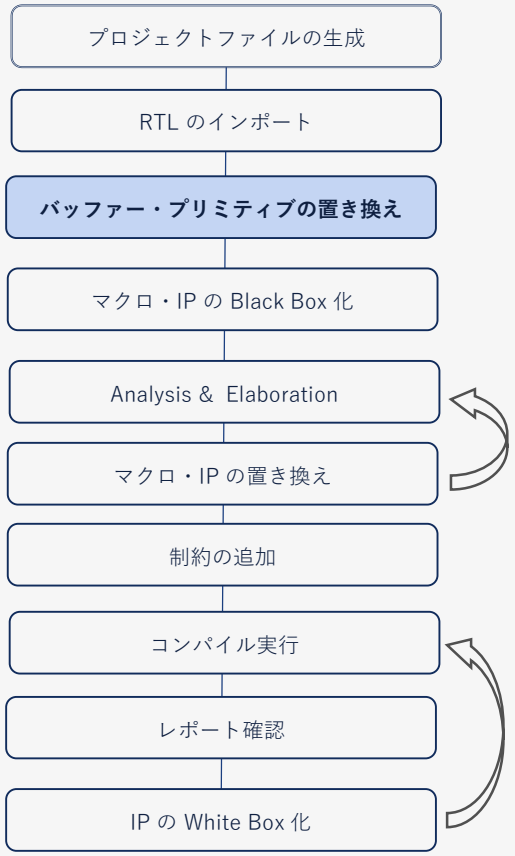
- Quartus® Prime は、入力・出力・双方向のバッファーを自動挿入
- 変換方法
  1. Xilinx® デザインの HDL コードから 全てのバッファー・プリミティブを削除
  2. HDL コード内のプリミティブを wire (Verilog HDL) か signal (VHDL) 宣言で置き換え
  3. バッファーの種類によっては Assignment Editor で設定をする



バッファー種類	使用する設定
各 I/O 規格のバッファー	I/O Standard
グローバル・バッファー	Global Signal
各 I/O 規格のグローバル・バッファー	I/O Standard

バッファー・プリミティブの置き換え

- Assignment Editor を使用して、グローバル信号と I/O 規格の割り当てを行う
  - 入力 a, b, clk はグローバル信号として割り当てられる
  - 入力ポート a と b は特定に I/O 規格が割り当てられ、その他のポートは自動でデフォルトの I/O 規格が割り当てられる
    - 制約の追加の項を参照



Assignment Editor

<<new>> | Filter on node names: * | Category: All

		From	To	Assignment Name	Value	Enabled
1	✔		a	I/O Standard	SSTL-12	Yes
2	✔		b	I/O Standard	SSTL-18 Class I	Yes
3	✔		c	I/O Standard	1.8 V	Yes
4	✔		clk	I/O Standard	1.8 V	Yes
5	✔		a	Location	PIN_AU25	Yes
6	✔		b	Location	PIN_F14	Yes
7	✔		c	Location	PIN_AW20	Yes
8	✔		clk	Location	PIN_AW26	Yes
9	✔		a	Global Signal	Global Clock	Yes
10	✔		b	Global Signal	Global Clock	Yes
11	✔		clk	Global Signal	Global Clock	Yes

バッファー・プリミティブの置き換え例 (Verilog HDL)

Vivado® 元の Verilog HDL コード

```
module Top (a,b,c,clk);

input a, b, clk;
output c;

reg c_buf;
wire a_buf, b_buf, clk_buf;

BUFG inst1 (.O (clk_buf), .I (clk));
IBUFG #("FALSE", "SSTL12") inst2 (.O (a_buf), .I (a));
IBUFG #("FALSE", "SSTL18_I") inst3 (.O (b_buf), .I (b));
OBUF inst4 (.O (c), .I (c_buf));

always @ (posedge clk_buf)
            c_buf <= a_buf & b_buf;

endmodule
```

Quartus® Prime 用に変換した Verilog HDL コード

```
module Top (a, b, c, clk);

input a, b, clk;
output c;

reg c_buf;
wire a_buf, b_buf, clk_buf;

assign clk_buf = clk;
assign a_buf = a;
assign b_buf = b;
assign c = c_buf;

always @ (posedge clk_buf)
            c_buf <= a_buf & b_buf;

endmodule
```

バッファー・プリミティブの置き換え例 (VHDL)

Vivado® 元の VHDL コード

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY buf_top IS PORT(
   a, b : IN STD_ULOGIC; clk : IN STD_ULOGIC; c : OUT
STD_ULOGIC);
END buf_top;
ARCHITECTURE Behave OF buf_top IS
   SIGNAL a_buf, b_buf, c_buf, clk_buf : STD_ULOGIC;
COMPONENT BUFG
   PORT (O : OUT STD_ULOGIC; I : IN STD_ULOGIC); END
COMPONENT;
   COMPONENT IBUFG
      generic(IBUF_LOW_PWR : boolean := FALSE;
            IOSTANDARD : String := "DEFAULT");
   PORT (O : OUT STD_ULOGIC; I : IN STD_ULOGIC); END
COMPONENT;
   COMPONENT OBUF
   PORT (O : OUT STD_ULOGIC; I : IN STD_ULOGIC); END
COMPONENT;
   BEGIN
      inst1 : BUFG
      PORT MAP (O => clk_buf, I => clk);
      inst2 : IBUFG
      generic map(
         IBUF_LOW_PWR => FALSE,
         IOSTANDARD => "SSTL12")
      PORT MAP (O => a_buf,I => a);
      inst3 : IBUFG
      generic map(
         IBUF_LOW_PWR => FALSE,
         IOSTANDARD => "SSTL18_I")
      PORT MAP (O => b_buf, I => b);
      inst4 : OBUF
      PORT MAP (O => c, I => c_buf);

      PROCESS(clk_buf) BEGIN
         IF (clk_buf'event and clk_buf = '1')
                  THEN c_buf <= a_buf AND b_buf;
         END IF;
      END PROCESS;
END Behave;
```

バッファー・プリミティブの置き換え例 (VHDL)

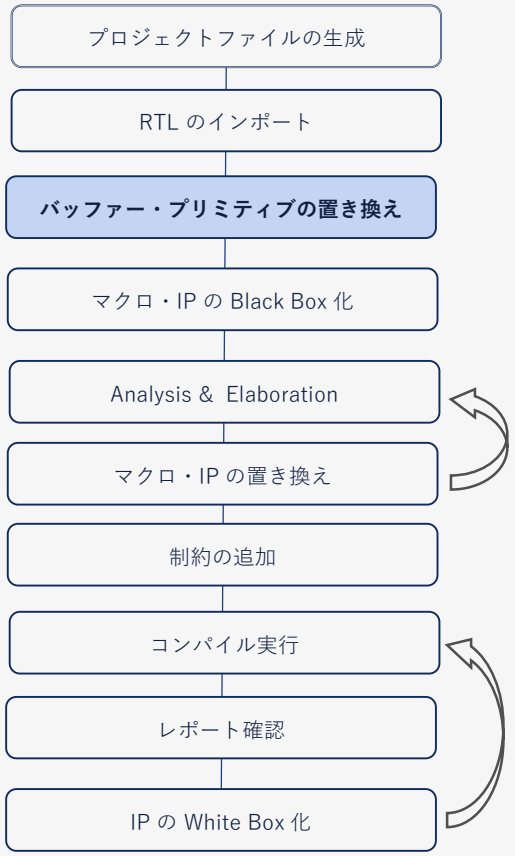
Quartus® Prime 用に変換した VHDL コード

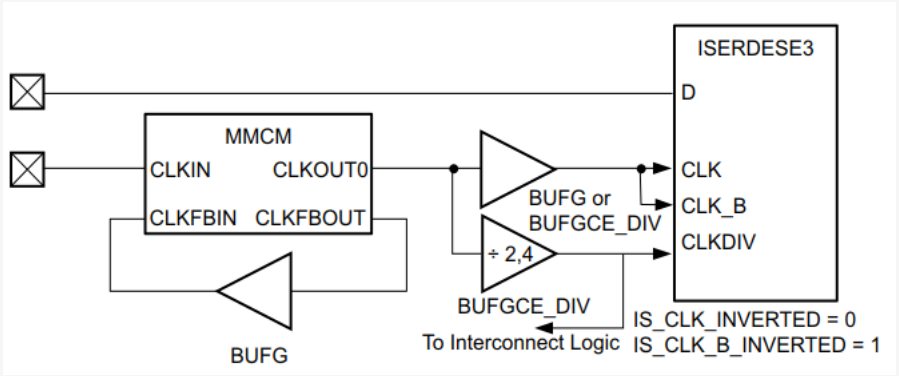
```
LIBRARY ieee;
USE ieee.std_logic
ENTITY Top IS
PORT(
   a, b: IN STD_ULOGIC;
   clk: IN STD_ULOGIC;
   c: OUT STD_ULOGIC);
END Top;
ARCHITECTURE Behave OF Top IS
   SIGNAL a_buf, b_buf, c_buf, clk_buf: STD_ULOGIC;
   BEGIN
   PROCESS (a, b, c_buf, clk)
     BEGIN
     clk_buf <= clk;
     a_buf <= a;
     b_buf <= b;
     c <= c_buf;
   END PROCESS;
   PROCESS(clk_buf)
     BEGIN
     IF (clk_buf'event and clk_buf = '1') THEN
        c_buf <= a_buf AND b_buf;
     END IF;
   END PROCESS;
END Behave;
```

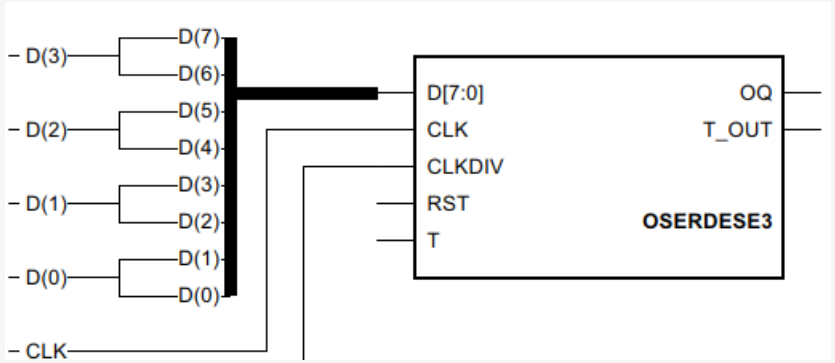
ISERDES / OSERDES の置き換え

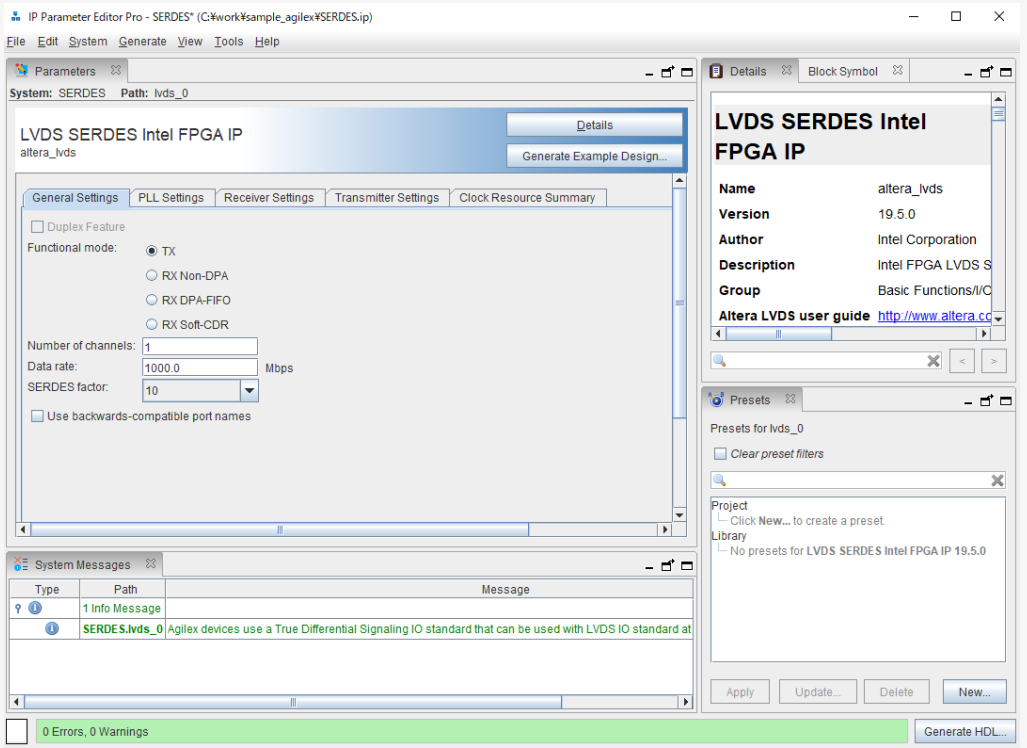
ISERDES / OSERDES の置き換え

- ISERDES / OSERDES + MMCM は LVDS SERDES で置き換える









High Speed Select IO Wizard

- ISERDES / OSERDES + MMCM は LVDS SERDES で置き換える

プロジェクトファイルの生成

RTL のインポート

バッファー・プリミティブの置き換え

マクロ・IP の Black Box 化

Analysis & Elaboration

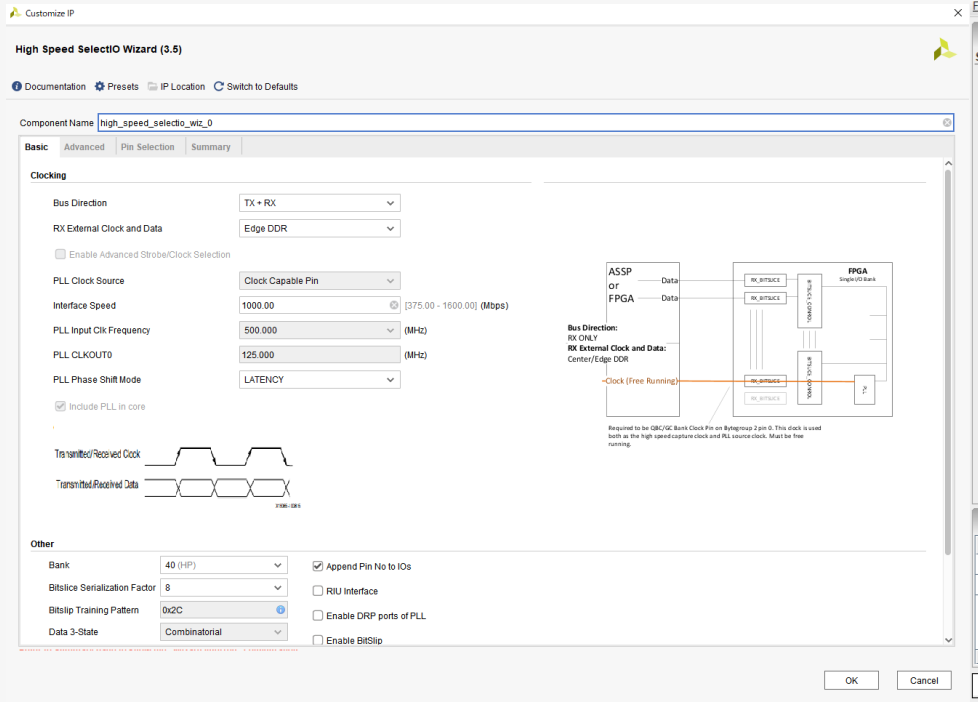
マクロ・IP の置き換え

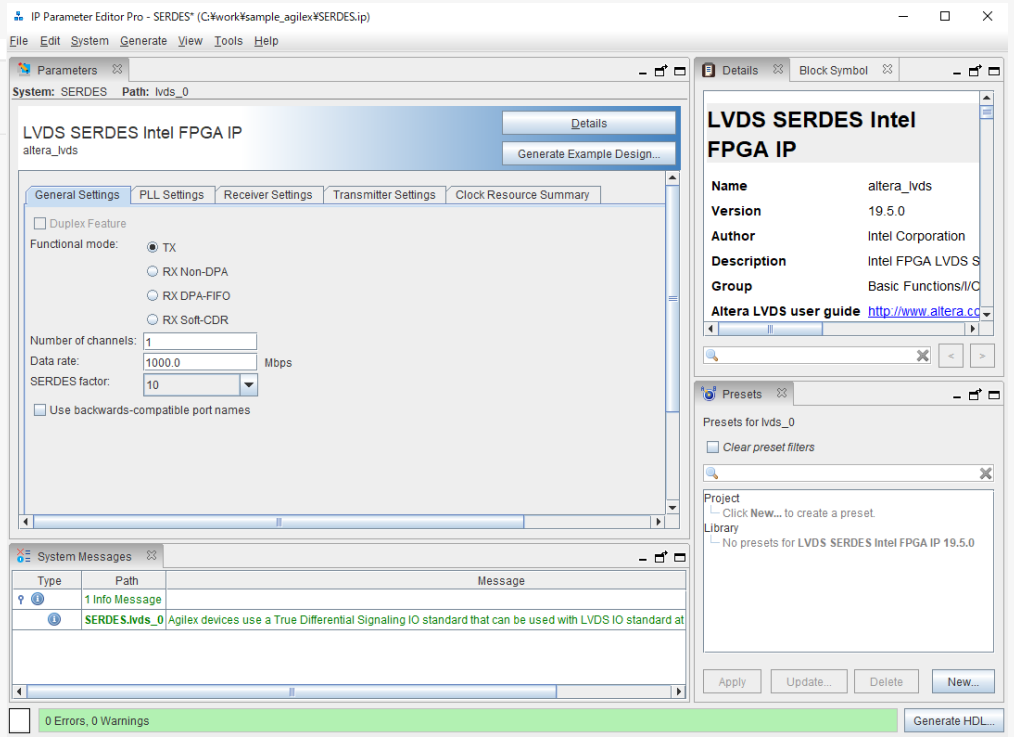
制約の追加

コンパイル実行

レポート確認

IP の White Box 化

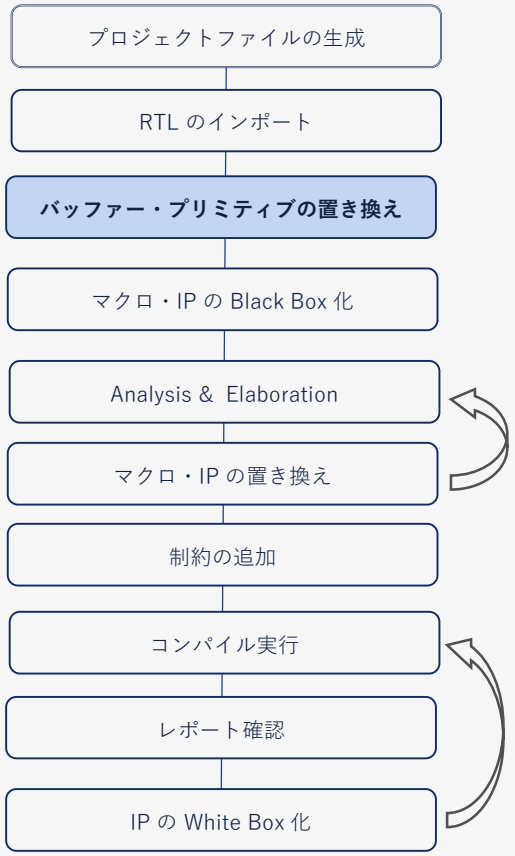


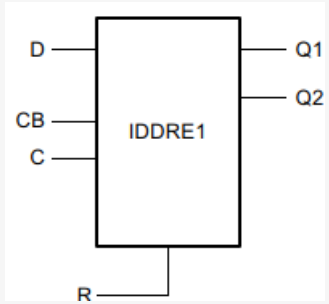


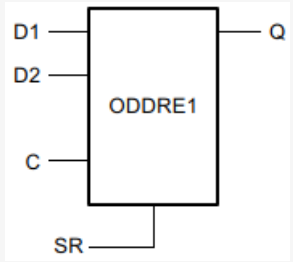
IDDR / ODDR の置き換え

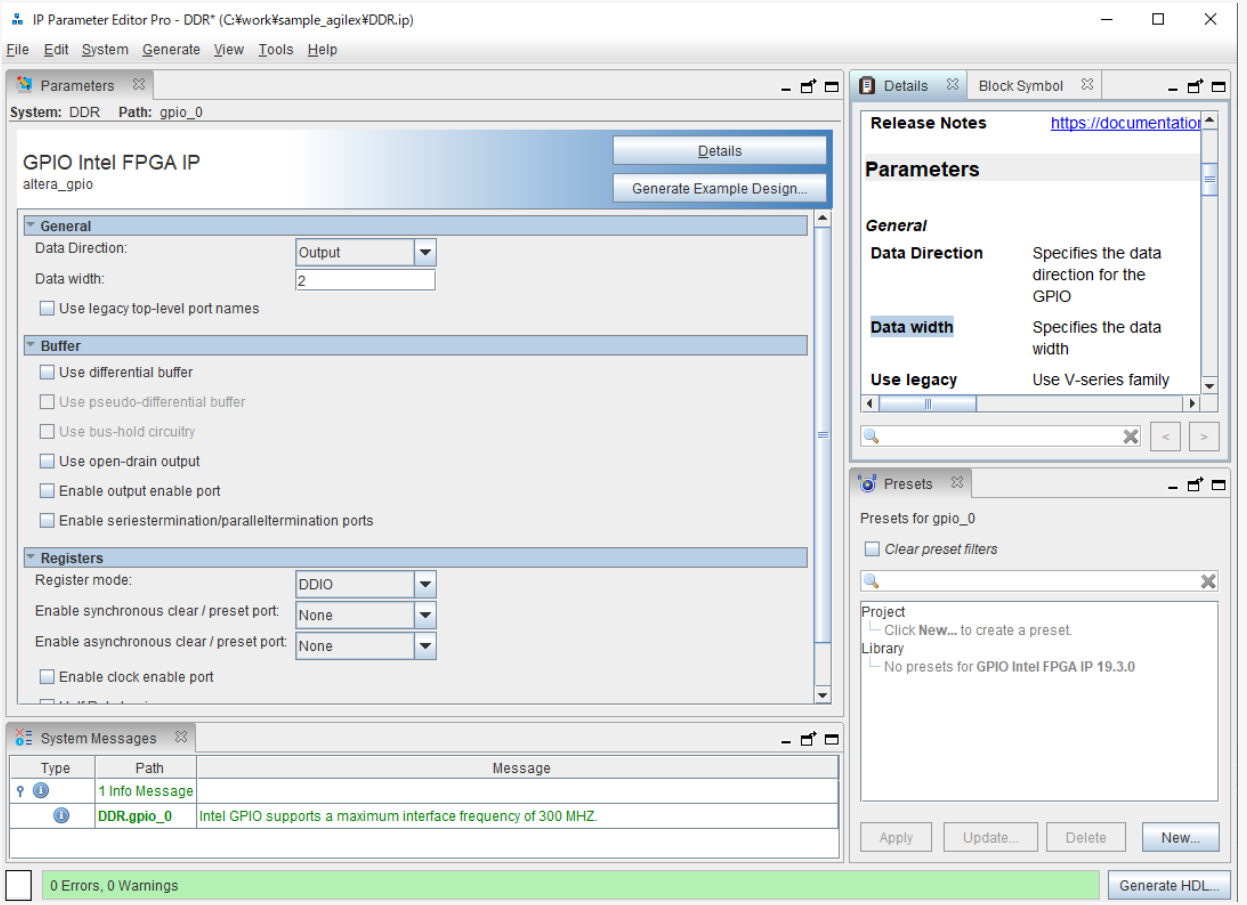
IDDR / ODDR の置き換え

- IDDR / ODDR は GPIO (DDIO) で置き換える









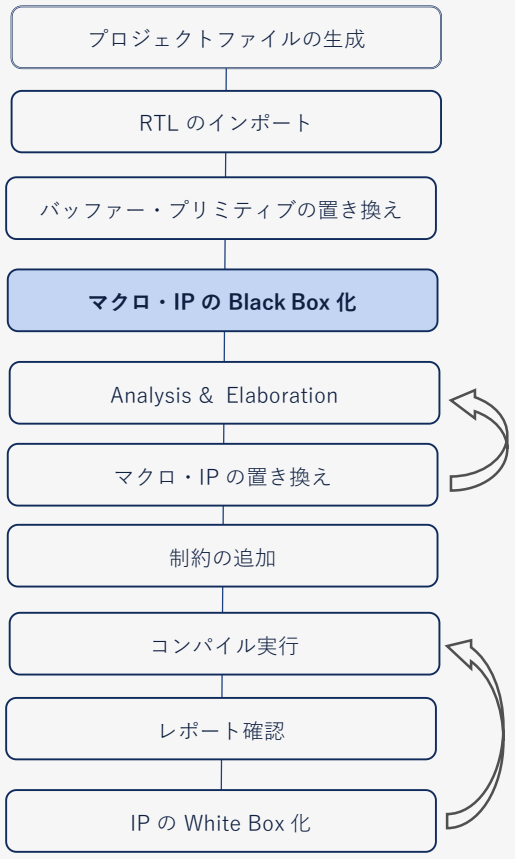
プロジェクト置き換え手順 4
(VHDL 版)

プロジェクトの置き換え手順 4 (VHDL 版)

- IP・マクロの Black Box 化 (VHDL版)
  - Black Box をインスタンスするときはラッパーファイルを作成
  - 3rd ベンダーの論理合成ツールを使用するときはアトリビュートを使用
    - アトリビュート：

      attribute syn_black_Box : boolean;

      attribute syn_black_Box of my_vhdlIP: component is true;



VHDL Black Box 記述例

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY top IS
   PORT (
      clk: IN STD_LOGIC ;
      count: OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
   );
END top;

ARCHITECTURE rtl OF top IS
   COMPONENT my_vhdlIP
   PORT (
      clock: IN STD_LOGIC ;
      q: OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
   );
   end COMPONENT;
   attribute syn_black_Box : boolean;
   attribute syn_black_Box of my_vhdlIP: component is true;
   BEGIN
      vhdlIP_inst : my_vhdlIP PORT MAP (
         clock => clk,
         q => count
   );
END rtl;
```

論理圧縮を防ぐために
アトリビュートを使用

プロジェクトファイルの違い (Quaruts® Prime vs Vivado®)

プロジェクトファイルの違い (Quaruts® Prime vs Vivado®)

- プロジェクト・ファイルの形式
  - Vivado® のプロジェクトファイルでは 拡張子.xpr の形式
  - Quartus® Prime のプロジェクトファイルでは .qpf .qsf ファイルを使用
    - .qpf (Quartus Prime プロジェクト・ファイル)
      - プロジェクト名とプロジェクトのすべてのリビジョン名
    - .qsf (Quartus Prime 設定ファイル)
      - デザインに適用されるタイミング制約以外のすべてのアサインメント
      - ファイル・リスト、デバイス、合成・配置配線制約、ピンなどの レイアウト制約

.qpf <sample.qpf>の例

```
# -------------------------------------------------------
#
# Quartus Prime
# Version 20.4.0 Build 72 12/14/2020 SC Pro Edition
# Date created = 13:02:33  February 10, 2021
#
# -------------------------------------------------------

QUARTUS_VERSION = "20.4"
DATE = "13:02:33  February 10, 2021"

# Revisions

PROJECT_REVISION = "sample"
```

.qsf <sample.qsf>の例

```
set_global_assignment -name TOP_LEVEL_ENTITY sample
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 20.4.0
set_global_assignment -name PROJECT_CREATION_TIME_DATE "13:02:33  FEBR
set_global_assignment -name LAST_QUARTUS_VERSION "20.4.0 Pro Edition"
set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 100
set_global_assignment -name DEVICE 10AX115S2F45E1SG
set_global_assignment -name FAMILY "Arria 10"
set_global_assignment -name DEVICE_FILTER_PACKAGE FBGA
set_global_assignment -name DEVICE_FILTER_PIN_COUNT 1932
set_global_assignment -name DEVICE_FILTER_SPEED_GRADE 1
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
```

MMCM と PLL の機能比較

MMCM と PLL の機能比較（参考）

機能		MMCM	Intel® FPGA IOPLL IP
周波数生成	クロックの乗算と除算	○	○
	位相シフト	○	○
	クロック・デューティ・サイクル	○	○
MMCM デスキュー・アジャスト	内部フィードバック	○	
	スペクトラム拡散	○	○
	システム同期 ノーマルモード	○	○
	ソース同期	○	○
	ゼロ遅延バッファ	○	○
	補償なし	○ (CMT から CMT への接続)	○ (ダイレクト補償)
	外部フィードバック	○	○
その他	入力クロックスイッチオーバー	○	○
	ダイナミック・リコンフィグ	○	○
	シングルクロック または差動クロック入力	○	○

Xilinx® UltraScale+™ vs. Intel Agilex® 7

MMCM と PLL のポート比較 (参考)

Xilinx® MMCM コア ポート	Inte® FPGA IOPLL IP コアポート	説明
clk_in1	refclk	１番目のクロック入力
clk_in2	refclk1	２番目のクロック入力
clkfb_in	fbclk	外部フィードバッククロック
clkfbout	fboutclk	フィードバックポートに供給
-	activeclk	I/O PLL が使用するリファレンスクロックソースを示す出力信号
clk_in_sel	extswitch	入力クロックポートの切り替え
reset	rst	非同期リセットポート
clk_out1, clk_out2, clk_outX	outclk_[]	クロック周波数出力ポート
clkinstopped	clkbad[1..0]	クロック入力信号がスイッチングを停止したかを示す
clkfb_stopped	-	フィードバッククロックが停止したかどうか指定
Locked	locked	PLL がロックされているかを示す
その他	adjpllin	アップストリーム I/O PLL から供給される入力信号
-	cascade_out	ダウンストリーム I/O PLL へ供給される入力信号
-	zdbfbclk	ミミック回路に接続する双方向ポート

MMCM と Dynamic Phase Shift のポート比較 (参考)

Xilinx® MMCM コア ポート	Intel® FPGA IOPLL の Dynamic Phase Shift ポート	説明
psclk	scanclk	ダイナミック・フェーズシフト・オペレーションのためのクロックを指定
psen	phase_en	ダイナミック・フェーズシフトの開始を指示
psincdec	updn	フェーズシフトの方向を指示
-	cntsel	ダイナミック・フェーズシフト・オペレーションのためののカウンタを指定
-	num_phase_shift	ダイナミック・フェーズシフト・オペレーションごとのフェーズシフト量を指定
psdone	phase_done	ダイナミック・フェーズシフト・オペレーションの完了
power_down	-	ユーザー選択用の power_down 入力ポートをイネーブル

参考

- AN 307: Intel® FPGA Design Flow for Xilinx* Users
  - https://www.intel.com/content/www/us/en/docs/programmable/683562/21-3/introduction-to-fpga-design-flow-for-users.html

Co.Tomorrowing

MACNICA

改版履歴

Revision	年月	概要
1	2021年10月	初版作成
2	2023年5月	テンプレート変更 ブランド名変更 インテル® Agilex™ -> Intel Agilex® 7 p10, p62 Black Box をインスタンスするときはラッパーファイルを作成。3rd ベンダーの論理合成ツールを使用する場合はアトリビュートを使用　に変更 p17, p18, p43-p48, 69 資料リンクの更新 p50 OBUFG_<I/O 規格> 説明変更 p57 誤記修正 OSERDE -> OSERDES