

演習への準備

KEK IPNS E-sys
本多良太郎

本演習ではFPGAの中でどのように信号が流れているか実際に見てもらいます。
そのために2つのデバッグツールを利用します。

VIO

- 仮想IOをFPGAへ実装するIP。
- インタラクティブなデバッグを行うための拡張IO。
- 仮想プッシュボタン・DIP、仮想LEDを追加することが出来る。

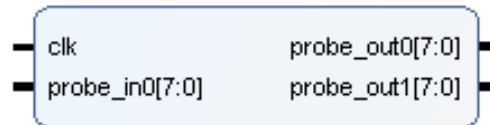
ILA (Integrated Logic Analyzer)

- FPGA内部の信号を調べるためのlogic analyzer IP。
- データキャプチャのトリガー条件を柔軟に変える事が出来、トリガー前後の信号の流れを見る事が出来る。

本講義では時間の都合からシミュレーションは行いません。

シミュレーションで論理的に正しいことを検証することは重要なステップです。
実際の設計ではシミュレーションも活用しましょう。

VIOのコンポーネント図



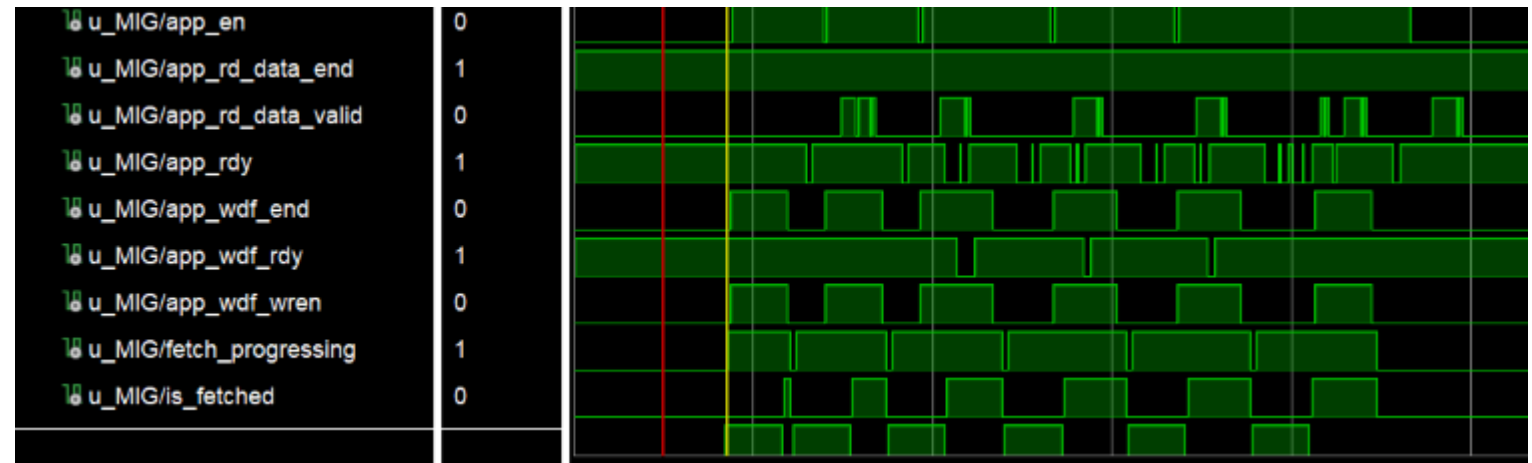
- 最大256個の入力・出力プローブを実装できる。
- (IP catalogからは64個まで実装可能)
- プローブデータ幅は1-256-bitで指定可能。
- クロック同期。

VIOの制御と監視はJTAGを介してhardware managerで行う。

- FPGAに埋め込み可能なlogic analyzer
- 現実のロジアナと同様デジタル信号の推移を観測できる
- データキャプチャする条件 (トリガー)を設定可能
 - 複雑な条件でしか起きない事象でもキャプチャできるようになる

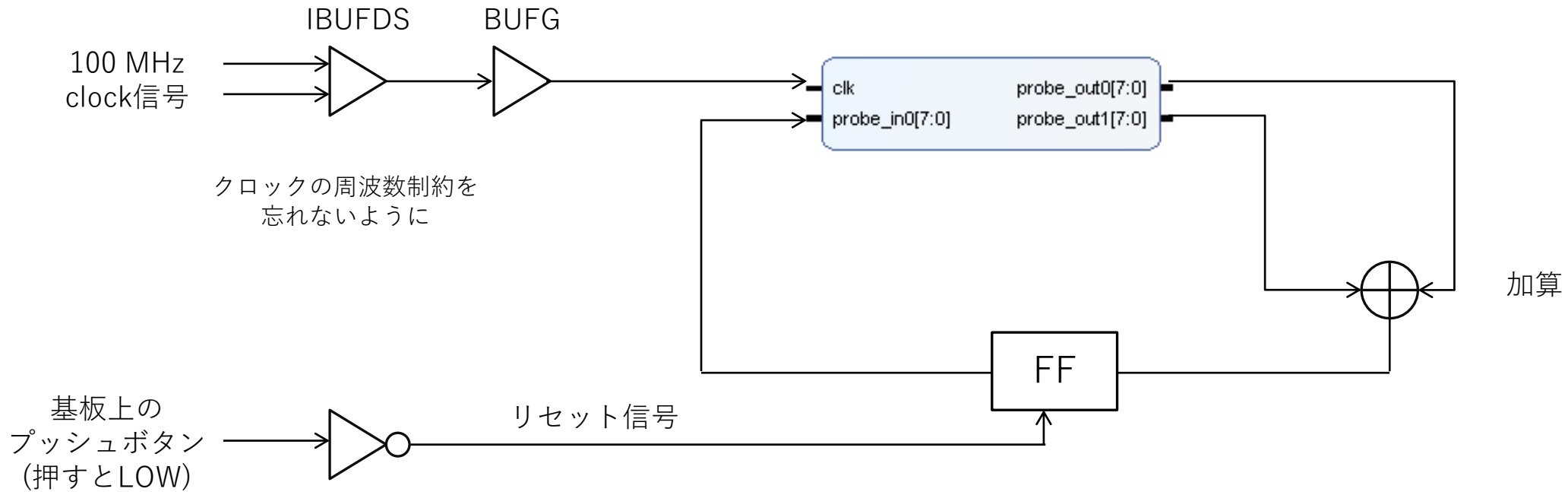
IP catalogから生成することも可能だが今回はソース内属性の指定で呼び出します。
VIOと同様操作はhardware managerで行う。

ILA確認できる波形の例

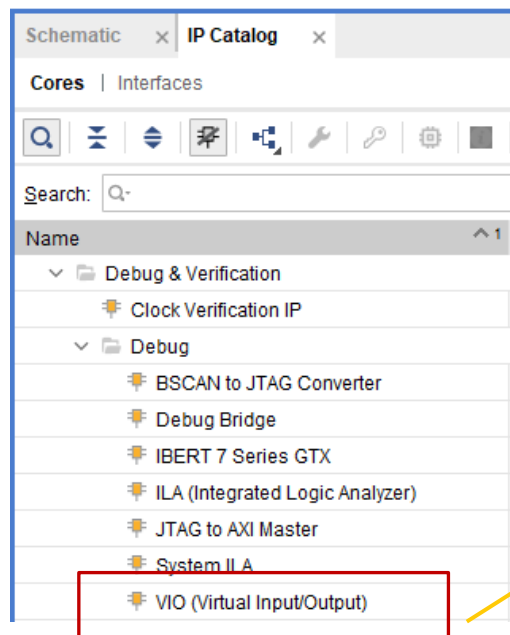


例題EX0 : VIOの実装

以下のような回路を実装しましょう。



例題：VIOの実装



Component Name: vio_1

To configure more than 64 probe ports use Vivado Tcl Console

General Options | PROBE_IN Ports(0..0) | PROBE_OUT Ports(0..1)

Input Probe Count: 1 [0 - 256]

Output Probe Count: 2 [0 - 256]

☒ Enable Input Probe Activity Detectors

To configure more than 64 probe ports use Vivado Tcl Console

General Options | PROBE_IN Ports(0..0) | PROBE_OUT Ports(0..1)

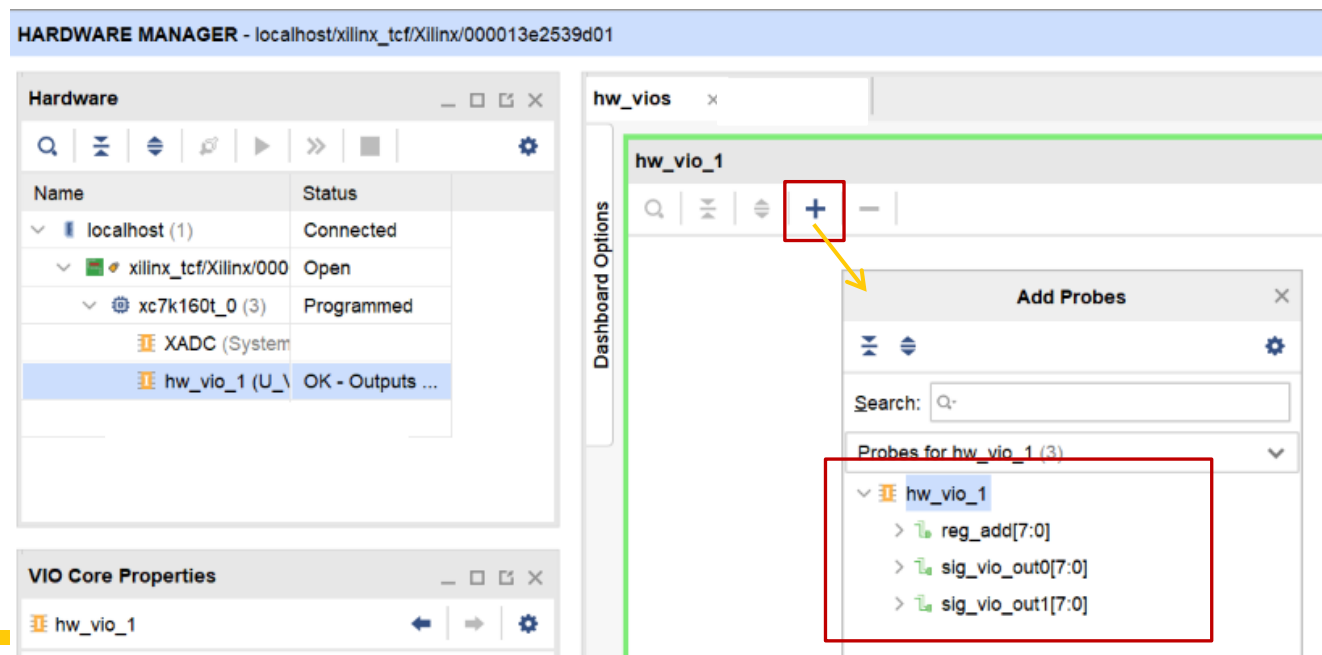
Probe Port	Probe Width [1 - 256]
PROBE_IN0	8

To configure more than 64 probe ports use Vivado Tcl Console

General Options | PROBE_IN Ports(0..0) | PROBE_OUT Ports(0..1)

Probe Port	Probe Width [1 - 256]	Initial Value (in hex)
PROBE_OUT0	8	0x0
PROBE_OUT1	8	0x0

- HDLが書けたら一気にbit stream生成まで行ってください。
 - 合成結果を見るとdbg_hubなる物が勝手に埋め込まれていることに気づくと思います。
 - VIOやILAはこのhubを介して外界と通信をします。
- Hardware managerを起動して.bitファイルと.ltxファイル両方をFPGAへ書き込みます。
 - .ltxファイルはデバッグツールの情報が埋め込まれたファイルです。
 - 忘れるとhardware managerでアクセスできないです。

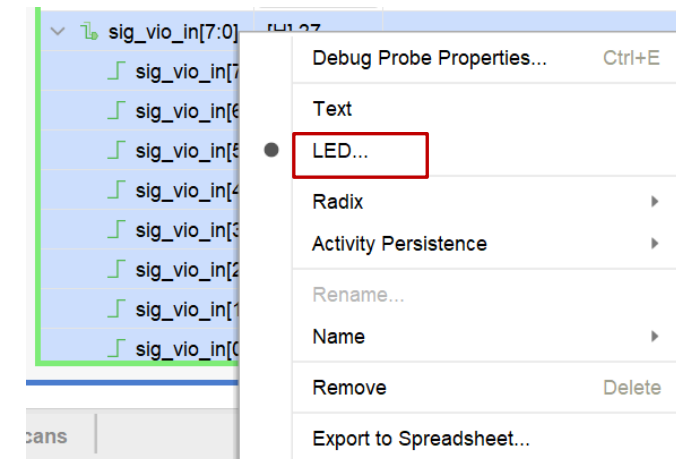
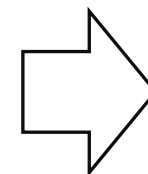
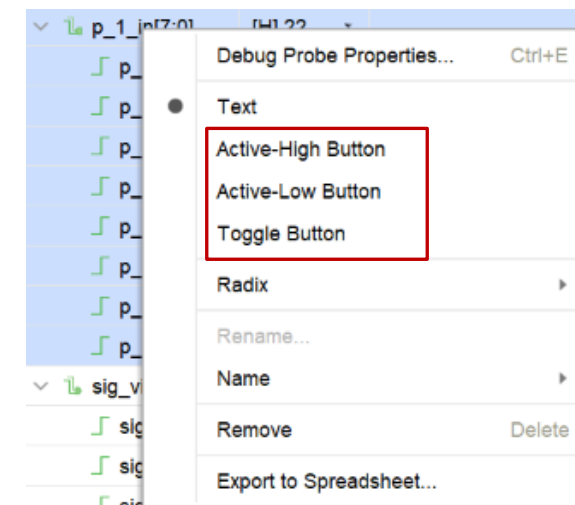
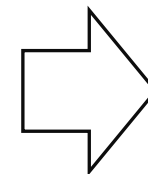


こういう画面が
出るはずです

VIOのプローブのうちどれを
表示するか選択します。

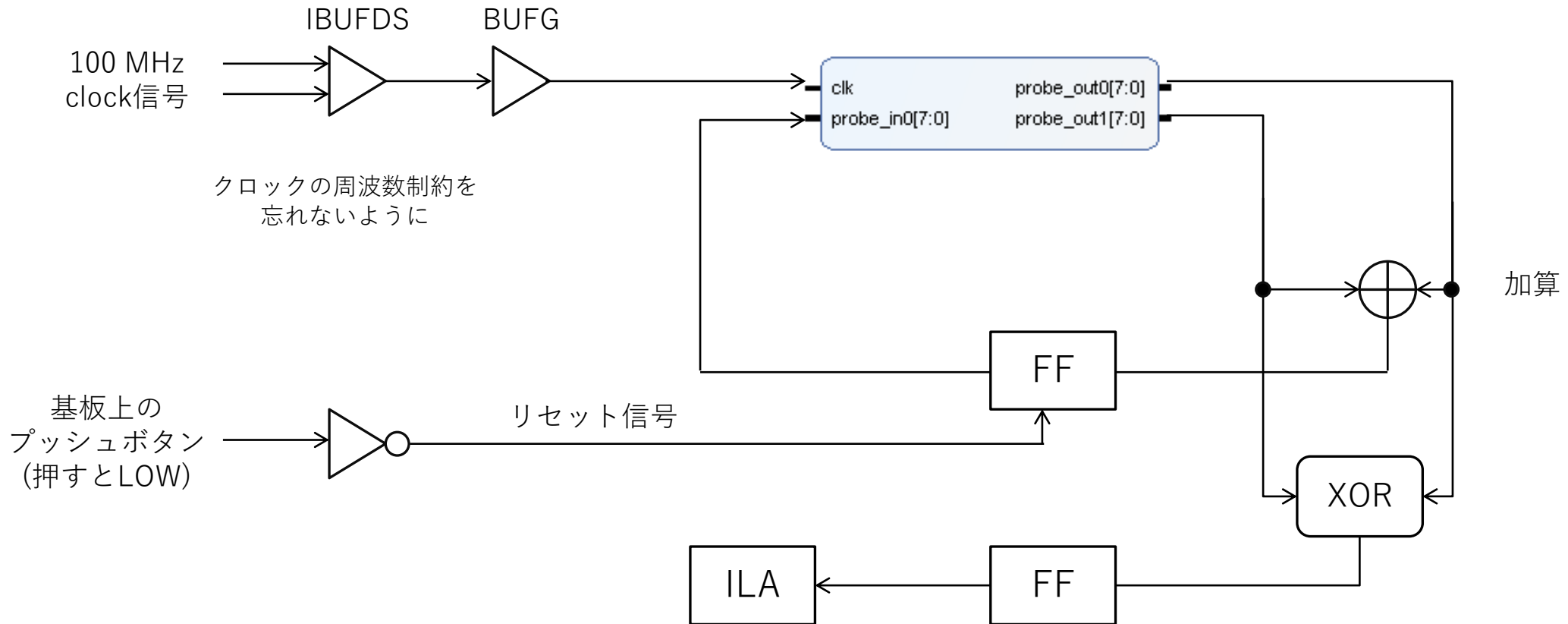
例題：VIOの実装

hw_vio_1					
Name	Value	Activity	Direct...	VIO	
> p_0_in[7:0]	[H] 05	テキストで値を変更できる 場合によってはやりづらい	Output	hw_vio_1	
▼ p_1_in[7:0]	[H] 22		Output	hw_vio_1	
└ p_1_in[7]	0		Output	hw_vio_1	
└ p_1_in[6]	0		Output	hw_vio_1	
└ p_1_in[5]	1		Output	hw_vio_1	
└ p_1_in[4]	0	モードを変えると ボタンとして操作できる	Output	hw_vio_1	
└ p_1_in[3]	0		Output	hw_vio_1	
└ p_1_in[2]	0		Output	hw_vio_1	
└ p_1_in[1]	1		Output	hw_vio_1	
└ p_1_in[0]	0		Output	hw_vio_1	
▼ sig_vio_in[7:0]	[H] 27		Input	hw_vio_1	
└ sig_vio_in[7]	●		Input	hw_vio_1	
└ sig_vio_in[6]	●		Input	hw_vio_1	
└ sig_vio_in[5]	●		Input	hw_vio_1	
└ sig_vio_in[4]	●	入力の方は LEDモードを選ぶと視覚的 にみる事が出来る	Input	hw_vio_1	
└ sig_vio_in[3]	●		Input	hw_vio_1	
└ sig_vio_in[2]	●		Input	hw_vio_1	
└ sig_vio_in[1]	●		Input	hw_vio_1	
└ sig_vio_in[0]	●		Input	hw_vio_1	



例題：ILAの実装

以下のような回路を実装しましょう。



この部分を追加してみる

HLDコード内で属性指定することによりVivadoの機能を使ってILAへ配線します。

Verilogの場合

mark_debugを付けて信号宣言する

```
(* mark_debug = "true" *) reg [7:0]counter;
```

```
always@(posedge clk_sys) begin
    if(rst_sys) begin
        counter[7:0]    <= 8'd0;
    end else begin
        counter[7:0]    <= counter[7:0] + 8'd1;
    end
end
```

VHDLの場合

宣言部 (beginの前)で属性指定する

```
signal reg_xor          : std_logic_vector(7 downto 0);

-- debug
attribute mark_debug    : string;
attribute mark_debug of reg_xor : signal is "true";
```

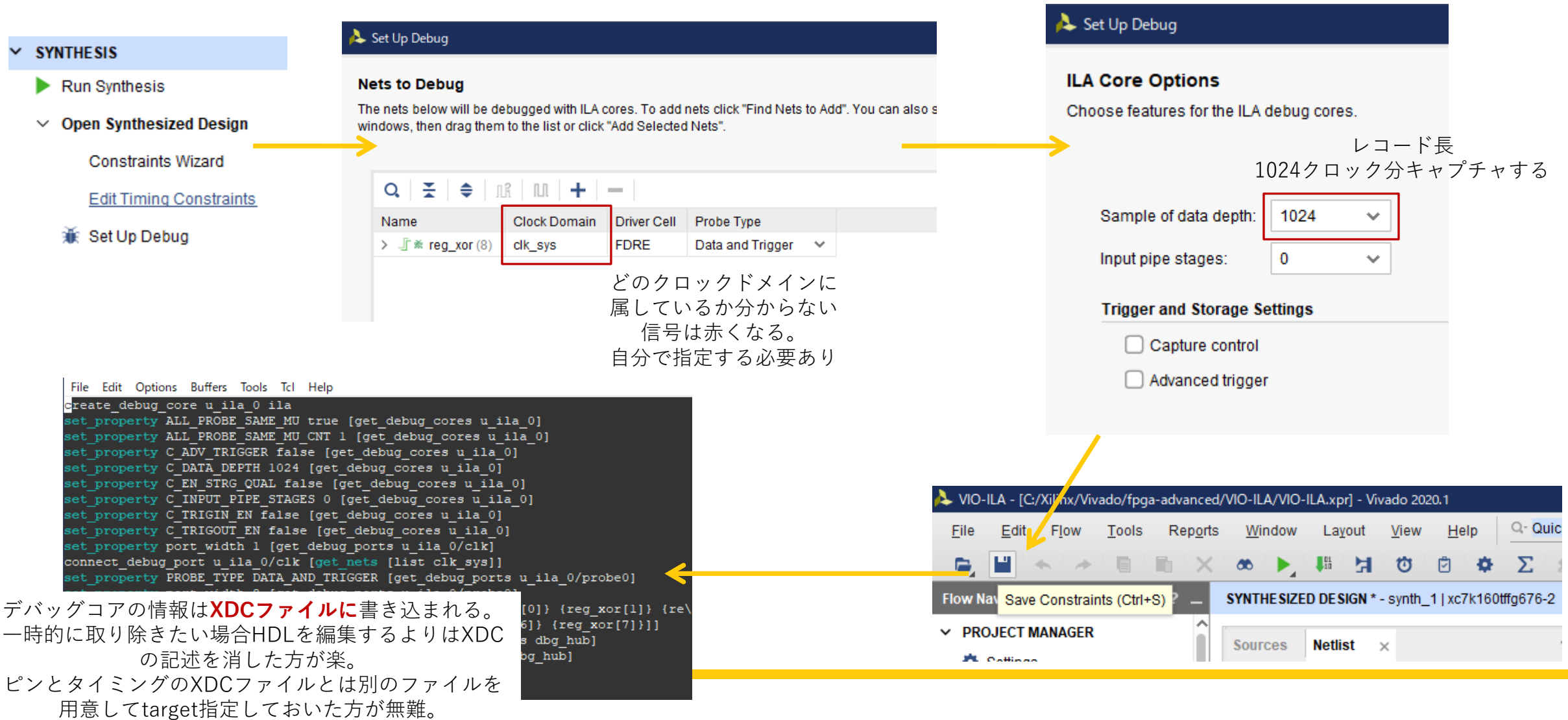
```
U_BUF : process(rst_sys, clk_sys)
begin
    if(rst_sys = '1') then
        reg_add    <= (others => '0');
    elsif(clk_sys'event and clk_sys = '1') then
        reg_add    <= std_logic_vector(signed(sig_vio_out0)
                                         + signed(sig_vio_out1));

        reg_xor    <= sig_vio_out0 xor sig_vio_out1;
    end if;
end process;
```

実際にreg_xorへ代入を記述している場所
(ここには特に何も書かない)

例題：ILAの実装

そのまま合成して、合成済みデザインにデバッグコアを埋め込みます。



Set Up Debug

Nets to Debug

The nets below will be debugged with ILA cores. To add nets click "Find Nets to Add". You can also select nets from the design windows, then drag them to the list or click "Add Selected Nets".

Name	Clock Domain	Driver Cell	Probe Type
> reg_xor (8)	clk_sys	FDRE	Data and Trigger

どのクロックドメインに属しているか分からない信号は赤くなる。
自分で指定する必要あり

ILA Core Options

Choose features for the ILA debug cores.

Sample of data depth: 1024

Input pipe stages: 0

Trigger and Storage Settings

☐ Capture control

☐ Advanced trigger

レコード長
1024クロック分キャプチャする

```
File Edit Options Buffers Tools Tcl Help
create_debug_core u_ila_0 ila
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
set_property port_width 1 [get_debug_ports u_ila_0/clk]
connect_debug_port u_ila_0/clk [get_nets [list clk_sys]]
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
```

デバッグコアの情報は**XDCファイル**に書き込まれる。
一時的に取り除きたい場合HDLを編集するよりはXDCの記述を消した方が楽。
ピンとタイミングのXDCファイルとは別のファイルを用意してtarget指定しておいた方が無難。

VIO-ILA - [C:/Xilinx/Vivado/fpga-advanced/VIO-ILA/VIO-ILA.xpr] - Vivado 2020.1

File Edit Flow Tools Reports Window Layout View Help

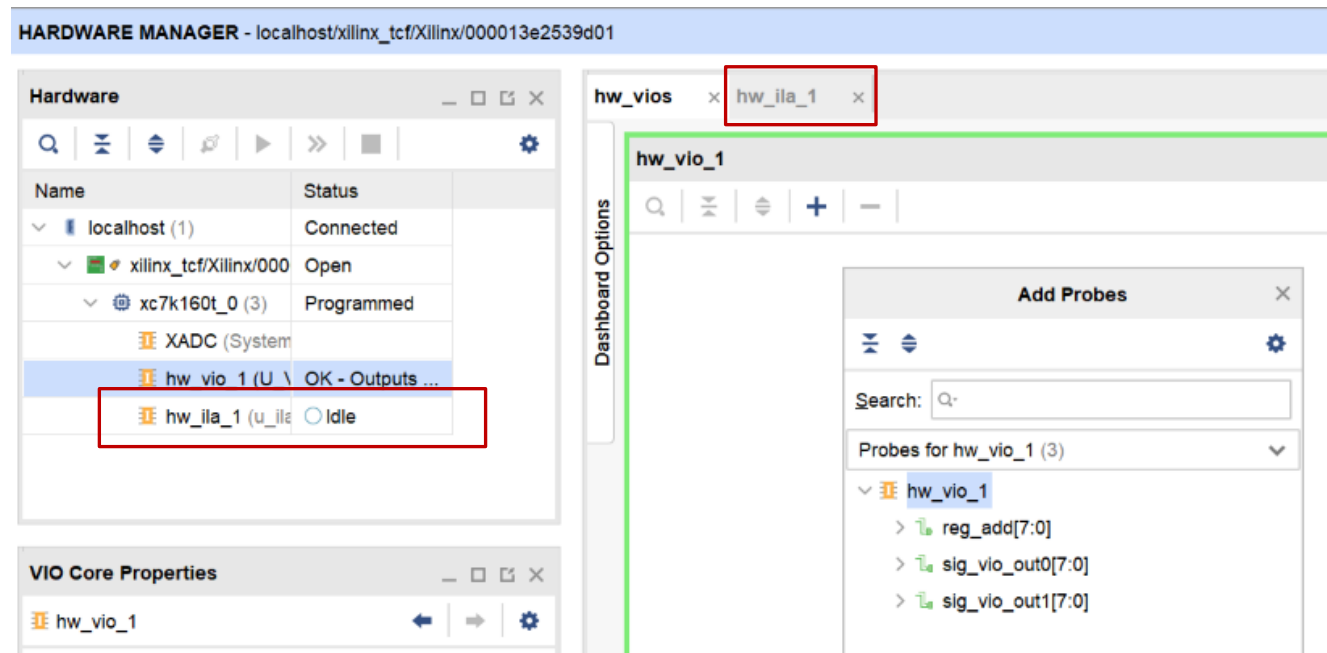
Flow Nav Save Constraints (Ctrl+S) SYNTHESIZED DESIGN * - synth_1 | xc7k160tffg676-2

PROJECT MANAGER

Sources Netlist

例題：ILAの実装

- XDCが書き変わったが、制約を読み込むのは合成の後なので配置配線に進んでよいです。
- 同じくhardware managerを起動して.bitファイルと.ltxファイル両方をFPGAへ書き込みます。



例題：ILAの実装

キャプチャを
開始する



Settings - hw_ila_1

Status - hw_ila_1

Core status: Waiting for Trigger

Capture status - Window 1 of 1

Window sample 512 of 1024

50%

トリガー発生を
待っている

Trigger Setup - hw_ila_1

Name	Operator	Radix	Value	Port	Comparator Usage
reg_xor[7:0]	==	[B]	0001_0000	probe0[7:0]	1 of 1

reg_xorが0x10になったらトリガーがかかる

トリガー条件が未設定だと即座に波形が現れる

Settings - hw_ila_1

Status - hw_ila_1

Trigger Mode Settings

Trigger mode: BASIC_ONLY

Capture Mode Settings

Capture mode: ALWAYS

Number of windows: 1 [1 - 1024]

Window data depth: 1024 [1 - 1024]

Trigger position in window: 512 [0 - 1023]

General Settings

Refresh rate: 500 ms

トリガーポジションをどこにするか
トリガーの前の情報が欲しいのか
後ろの情報が欲しいのか