再構成型アーキテクチャ特論(5)

osana@eee.u-ryukyu.ac.jp

前回の復習: テストベンチ

- * テストベンチ
 - * `timescale, initial, always #
 - * \$ ではじまるシステムタスク

 - * インスタンス名.信号名 で深い階層の信号も参照できる



* テスト対象モジュールにテストベンチで生成した信号を入力する

前回の復習: Vivado

* FPGAの型番を指定してプロジェクトを作る

* ソースファイルは別フォルダにしておいたほうがよい

* RTL は "Design Source" で、

* テストベンチは "Simulation Source"



前回の復習: Vivado Simulator

- * Vivado Simulator の基本的な使い方
 - * Simulation settings で runtime を 0 に
 - * 設定した時間だけ進める 🕅 70 ns 🗾
 - * 表示する信号を変更したら 🔀 ソースコードを変更したら 🗔
- * ソースコードが Web にあります:

http://mux.eee.u-ryukyu.ac.jp/lecture.html.ja









今日の予定

* ゴールはボードを動かすこと * LED くるくる \rightarrow 7セグメント LED も動かしたい * RTL を書いて、シミュレーションして、論理合成と配置配線 * シミュレーションには難しいこともある: parameter の活用 * 課題が出ます



Implementation Flow

* 論理合成

* テクノロジマッピング

* 配置配線

* ビットストリーム生成

* 詳しくは演習で



後半は実習

* 論理合成・配置配線して、ビットストリームを生成後 FPGA に書くまで
 * ソースファイルは Web で配布
 * 途中のスライドに出てくるソースコードはそれに含まれています



当面の目標

* ストップウォッチを作りましょう

* 10進力ウンタ

* 今日使うソースコードをかなりの部分流用できるはず

プッシュスイッチと7セグメントLEDの制御回路



スイッチ

* チャタリング除去

* この間のでは押しっぱなしにすると定期的に信号が出ます

* テスト用の回路は後ほど(配布に含まれているけど、完全ではない)

* 解決方法は各自考えましょう





7セグメントLED

* アノードコモン

* 点灯したい桁のアノードを L、

* 点灯したい部分のカソードを L で点灯





アノードを切り替えながら残像で全桁を表示 (ダイナミック駆動)





人間と電子回路のタイムスケール

- * 回路は 100MHz で動くけど人間はムリ

 - * 6Hz なら 2²⁴=16M まで数えればよい

* LED が 100MHz で動くと見えないので、カウンタを入れて遅くする

* シミュレータの波形で 16M まで追いかけるのは…? (やっぱり無理)

* シミュレーションと論理合成でカウンタのビット数を変更したい

LEDくるくる再び

* 赤字のところを変更したい

* たとえば2ビットなら 波形見て追いかけられる

```
module led_kurukuru
    ( input wire CLK, RST,
      output reg [15:0] LED );
  reg [23:0] CNT;
  wire STROBE = &CNT;
  always @ (posedge CLK) begin
     if (RST) begin
        CNT <= 0;
        LED <= 16'b1000_0000_0000;
     end else begin
        CNT <= CNT+1;
        if (STROBE)
          LED <= {LED[0], LED[15:1]}
    end
  end
endmodule
```

Parameter を使った宣言

* やり方はふたつ

* モジュールの先頭で宣言

* ポート宣言の前に #() で宣言

* どちらも規定値が必要

* 後者はポート幅変更も可能

module led_kurukuru
 (input wire CLK, RST,
 output reg [15:0] LED);

```
parameter CounterBits = 24;
reg [(CounterBits-1):0] CNT;
(以下同文)
```

module led_kurukuru #
 (parameter CounterBits = 24)
 (input CLK, RST,
 output reg [15:0] LED);

```
reg [(CounterBits-1):0] CNT;
(以下同文)
```

外部からパラメータを変更

* 前の例は何もしないと 24

* インスタンス宣言時に変更可

* kuru2 では2ビットカウンタ

* テストベンチでは数字を変え、 論理合成は規定値で、とか便利



led_kurukuru kuru1(.CLK(CLK), .RST(RST), .LED());

led_kurukuru # (.CounterBits(2)) kuru2(.CLK(CLK), .RST(RST), .LED());





7セグメントLED

- * 0~9の数字と点灯パターン
 - * 小数点を別として7ビット
 - * これをカソードに与える
 - * 0で点灯
- * アノード(桁)の駆動はまたあとで















Individual cathodes



1桁だけ動かすことを考える

* 4ビット入力→7ビット出力









_		
10	011	00











endmodule

論理圧縮はツールがやってくれる!

最後はこうしたい→構成を考える

* 8桁分の値を32bitで入力

各桁4bit

* segment_driver の修正で 16進表示も可能











* ストップウォッチ作りましょう

* 1/10000 秒まで測れる

* 最大 9999.9999 秒

* 入力は RST と START-STOP のふたつ。後者はチャタリング除去する * 100MHz のクロックは無限の精度だと思って信用してよい

演習1:LEDくるくる



演習の目的

* シミュレーションと実機で違う挙動にする * ソースファイルはすべて準備してあります

* Implementation flow での Vivado の使い方を理解する

* シミュレーション時はカウンタのビット数の parameter を変更

ソースファイル

- * ソースは src1/ に入っています
 - Constraint: led_kurukuru.xdc
 - Design Source: led_kurukuru.v
 - Simulation Source: led_kurukuru_test.v

プロジェクトを作る

- * USB メモリから C:/Users/user01/ 以下にフォルダごとコピー
- * lab1/vivado としてプロジェクトを作成
 - Device: XC7A100T-1CSG324
 - * ソースファイルは先ほどの3つ



シミュレーションしてみる



Flow Navigator

- * Project Manager: プロジェクト管理 (まとめ)
- Simulation: シミュレーション
- RTL Analysis: HDL の解析
- Synthesis: 論理合成
- Implementation: テクノロジマップと配置配線
 - Program and Debug: ビットストリーム生成とFPGAへの書き込み

ここが実質的な **Implementation Flow** (上から下へ順番にやる)

(IP Integrator は今日は使いません)

	ow Navigator 👳 📥
۵	Project Manager
	🊳 Project Settings
	😚 Add Sources
	💡 Language Templates
	手 IP Catalog
⊿	IP Integrator
	🎎 Create Block Design
	💕 Open Block Design
	🧠 Generate Block Design
⊿	Simulation
	o Simulation Settings
	🔍 Run Simulation
⊿	RTL Analysis
	👂 📑 Open Elaborated Design
⊿	Synthesis
	🊳 Synthesis Settings
	📚 Run Synthesis
	👂 📑 Open Synthesized Desigr
⊿	Implementation
	🚳 Implementation Settings
	Run Implementation
	🕨 💕 Open Implemented Desig
۵	Program and Debug
	🊳 Bitstream Settings
	🚵 Generate Bitstream
	N States Handware Manager



Project Manager

* 最初の画面

* ターゲットデバイスとかそういう情報はここに

* 右下に表示されているもの:

- * Messages / Log: エラーが出たときなどに確認する
- * Design Runs: Implementation flow の進行状況を表示



RTL Analysis

- * "Open Elaborated Design" で開始
 - * 合成から先のフローを起動しても、必要なら自動で走る
 - * 完了すると Elaborated Design に関するメニューが出る
- * Schematic は RTL と対応しており、右クリックで該当箇所へ
- 特に重要な意味はないですが、自分のRTL設計を概観したいときに







Synthesis

- * Run Synthesis で起動
 - - *
 - * Project Summary にリソース使用量の予測が出ます
 - *







ings
;
Design
s Wizard
ig Constrain
bug
ming Summ:
ock Network
ock Interacti
RC
oise
ilization
wer



Implementation

* Run Implementation で起動

* I/O だけでなく回路がちゃんと配置配線された状態になる

* Project Summary のリソース使用量が確定版に

タイミングと I/O の結果が非常に重要!

Implementation Run Implementation





Implementation: Timing

* Implementation が完了すると出る

* Implemented Design を開いて Window \rightarrow Timing

Timing – Timing Summary – impl_1					
옥 🛣 🖨 ᆃ 🏓 🛃	De				
(i) This is a <u>saved report</u> ×	Set				
General Information					
—Timer Settings					
—Design Timing Summary					
Clock Summary (1)					
-Check Timing (17)					
-Intra-Clock Paths					
—Inter-Clock Paths 📃	All				
—Other Path Groups 📃 🖓					
User Ignored Paths	◀				
Timing Summary - impl_1 \times					
💭 🔚 Tcl Console 💭 Messages 🛛 🖾	Log				

* "All user specified timing constraints are met" がないとダメ

				_ D & ×				
sign Timing Summary								
up		Hold		Pulse Width				
Worst Negative Slack (WNS):	<u>6.037 ns</u>	Worst Hold Slack (WHS):	<u>0.236 ns</u>	Worst Pulse Width Slack (WPWS):				
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):				
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:				
Total Number of Endpoints:	56	Total Number of Endpoints:	56	Total Number of Endpoints:				
user specified timing constraints are met.								
				4 ▷ 🗉				
🕒 Reports 📑 Design Runs	🍯 Timing	D I/O Ports						



Implementation: I/O Ports

* Implemented Design を開いて Window \rightarrow I/O Ports

* 間違えると最悪の場合ボードが壊れます

1/0	I/O Ports												
0	Name	Direction	Neg Diff Pair	Site		Fixed	Bank	I/O Std		/cco	Vref Drive	Strength Slew Ty	/pe
-	P- lo All ports (18)												
	∲- 🔞 LED (16)	OUT				V	3	4 LVCMOS33*		.300	12	👻 SLOW	-
		OUT	F	P2		V	3	4 LVCMOS33*		.300	12	👻 SLOW	
⊡ _≩	ED[14]	OUT	F	R2	v	V	3	4 LVCMOS33*		.300	12	👻 SLOW	-
	LED[13]	OUT	l	J1		V	3	4 LVCMOS33*		.300	12	👻 SLOW	I
6	- 🕢 LED[12]	OUT	F	P5	Ŧ	V	3	4 LVCMOS33*		.300	12	👻 SLOW	- E
2		OUT	F	R1		V	3	4 LVCMOS33*		.300	12	👻 SLOW	- T
Ŧ		OUT	١	/1	v		3	4 LVCMOS33*		.300	12	👻 SLOW	-
	LED[9]	OUT	l	J3		V	3	4 LVCMOS33*		.300	12	👻 SLOW	
\checkmark		OUT	١	√4	v	V	3	4 LVCMOS33*	- T	.300	12	👻 SLOW	*
		OUT	l	J6		V	3	4 LVCMOS33*		.300	12	👻 SLOW	- T
		OUT	l	J7	Ŧ	V	3	4 LVCMOS33*		.300	12	👻 SLOW	- T
		OUT		Г4		V	3	4 LVCMOS33*		.300	12	👻 SLOW	- L
		OUT		Γ5			2	LIVCMOS33*		300	17	₩ SLOW	
	🔚 Tcl Console 💭 Messages 🔄 🖄 Log 🗋 Reports 📑 Design Runs 🛛 🧭 Timing 🕞 🛛 🖓 O Ports												

* 全部のポートが "Fixed" で、I/O Standard が正しいことを確認



ビットストリーム生成

- * 配置配線の結果をFPGAに書き込める形にする
 - * ハードウェアマネージャでFPGAに書き込む前にやっておく
 - は自動的に起動



* Synthesis \rightarrow Implementation \rightarrow Generate Bitstream と順番に起 動しなくても、最初から Generate Bitstream すれば必要なステップ

* これは Implementation とかでも同じなので、時間次第で使い分け

Hardware Manager

- * ボードは事前に接続、電源 on しておく
- * Open Target で FPGA へ接続
- ボード上のデバイスが認識されたことを確認
- * Program device で書き込み
 - ファイル名とかは特に指定しなくてよい



Hardware Manager - unconnected

(i) No hardware target is open. Open target

Haroware Manager – macpro-linux/xilinx_tct/Digilent/210274593047A								
There are no debug cores. Program device <u>Refresh device</u>								
Hardware	🔷 xc7a100t_0	Debug Probes _						





チェックすること

* 実機ではLED が順番に点灯すれば OK

* シミュレーションとは点灯の切り替え速度が違うことも確認しましょう

* テストベンチから Parameter を変更してみるとなおよい



演習2: 16進LEDカウンタ

プロジェクトを作る

- * \mathcal{F} Rev Project
- * ソースは lab2/src
 - * push_counter_test.v: テストベンチ, 7seg.xdc: 制約
 - * その他はデザインソース









ボタンの割り当て

* 左 (BTNL): ADV

* カウンタをふやす

* 中央 (BTNC): RST

* リセット



Parameter はふたつ

* LED の桁の切り替え (ColumnCounterBits) * 10bit カウンタ = 1/100 ms (ちょっと速すぎたかも…)

* チャタリング除去のカウンタ (ButtonFilterCount)

* $20x10^6 = 1/5sec$



試してみる

* 左ボタンを押しっぱなしにする

* 5Hz でカウンタが増えてしまう

push_counter.v で ColumnCounterBits = 24

* 実習1 と同じ間隔で LED が切り替えになる



課題ふたたび

* ストップウォッチ

* 今回の演習のモジュールを再利用して構いません

* 締切は 11/27

* ソースファイル「だけ」、つまり *.v と *.xdc をメールで提出

* どういう構成にしたかをA4で1,2枚程度で記述してPDFで提出

