組み込みソフトウェア実践プロジェクト演習講座

（本演習は EDK8.2i , ISE8.2i を対象としています）

Lab1: BSB を使った PowerPC システムの構築

《この Lab で習得する事》

BSB(BaseSystemBuileder)を使ったツール基本操作と FPGA へのダウンロード手順を習得します。

１）下記要件を満たす PowerPC システムを BSB(BaseSystemBuilder)で作成してください。

その後 BitFile 生成まで行い FPGA ボードにダウンロードし動作確認まで行ってください。

- system.xmp は lab1 フォルダに作成してください
- repository は使用しません
- new design を作成します
- Board Vendor:Xilinx, Board Name:Virtex4 ML403 Evaluation Platform, Board revision:1 にします
- Processor は PowerPC を使います
- Reference clock:100MHz, Processor clock:100MHz, Bus clock:50MHz にします
- Debug I/F:FPGA JTAG, OCM なし, Cache なしです
- 使用する Peripheral は以下です
  - LEDs_4Bit (Interrupt 使用)　←チェック忘れないで下さい！
  - LEDs_Positions
  - Push_Buttons_Position

- RAM 構成は以下のようにしてください
  - plb_bram_if_cntlr_1 64KB
  - opb_bram_if_cntlr_1 4KB
  - opb_bram_if_cntlr_2 4KB

- Sample application は Peripheral selftest だけ選択してください
- Instruction, Data, Stack/Heap はすべて plb_bram_if_cntlr_1 に格納してください

確認画面で設定を確認の後、Generate してください。

２）“Start using Platform Studio”を選択し、
メニューの Hardware -> Generate Bitstream を実行してください（多少時間がかかります）

３）メニューの Software -> Generate Libraries and BSPs を実行してください。

４）メニューの Device Configuration -> Update Bitstream を実行してください。

５）メニューの Device Configuration -> Download Bitstream を実行してください。

FPGA ボードの LED 点灯を確認してください。
“CPU Reset”のプッシュボタンで再動作させられます。

６)Applications タブの”Sources”にある以下の C ソースを確認してください。
LED を点灯させる記述になっています。

- TestApp_Peripheral.c
- xgpio_tapp_example.c

（xintc_tapp_example.c は今は関係ないので確認する必要はありません）

時間に余裕がある人は C ソースを変更しても構いません。変更後は４)、５）を再実行してください。

７）メニューの File -> Close Project でプロジェクトを終了してください。

Lab2: ループ文を使ったLEDブリンク

《このLabで習得する事》

Cの基本的な記述スタイルとPowerPC及びPeripheralの初期化方法を習得します。

１）lab1フォルダを同じ階層にコピーし、lab2にリネームしてください。

２）メニューのFile -> Open Project でlab2の下のsystem.xmpを選択してください。　（※lab1ではないので注意してください）

３）Applicationsタブの”Sources”と”Headers”のリストをすべてremoveしてください。
（removeはファイルを選択し、右クリック）

４）lab2¥TestApp_Peripheral¥src　の TestApp_Peripheral_LinkScr.ld 以外の*.cと*.hを削除します。

５）source_files¥forlab2¥TestAppLab2.c をlab2¥TestApp_Peripheral¥srcにコピーしてください。

６）TestAppLab2.c はループ文で LED ブリンクする記述です。未完成ですのでincompleteの箇所に記述を追加し完成させてください。

７）Applicationsタブの”Sources”にTestAppLab2.cを追加してください。
（ファイル追加は、右クリックでAdd Existing Files）

８）Update Bitstream, Download Bitstreamを実行してください。
エラーがあれば、TestAppLab2.cを修正してください。
（※ソースコード以外のエラーは講師に聞いてください）

時間に余裕がある人はDS3,11,12,13,14のLEDもブリンクさせてみてください。

９）メニューのFile -> Close Projectでプロジェクトを終了してください。

Lab3: 割り込みを使ったLEDブリンク

《このLabで習得する事》

MHSファイルの編集と割り込みの記述スタイルを習得します。

１）lab2フォルダを同じ階層にコピーし、lab3にリネームしてください。

２）メニューのFile -> Open Project でlab3の下のsystem.xmpを選択してください。

３）Applicationsタブの”Sources”のTestAppLab2.cをremoveしてください。

４）lab3¥TestApp_Peripheral¥src¥TestAppLab2.cを削除します。

５）source_files¥forlab3¥TestAppLab3.c をlab3¥TestApp_Peripheral¥srcにコピーしてください。

＜ソースコード編集の前に以下の作業をします＞

６）lab3フォルダにあるsystem.mhsの以下を修正してください。

- INSTANCE=LEDs_4Bitの以下のパラメータをコメントしてください。

```
# PARAMETER C_INTERRUPT_PRESENT = 1
```

- INSTANCE=LEDs_4Bitの以下のPORT宣言をコメントしてください。

```
# PORT IP2INTC_Irpt = LEDs_4Bit_IP2INTC_Irpt
```

- INSTANCE=Push_Buttons_Positionに以下のパラメータを設定してください。

```
PARAMETER C_INTERRUPT_PRESENT = 1
```

- INSTANCE=Push_Buttons_Positionに以下の太字の行を追加してください。

```
BUS_INTERFACE SOPB = opb
PORT IP2INTC_Irpt = Push_Buttons_Positon_IP2INTC_Irpt
PORT GPIO_IO = fpga_0_Push_Buttons_Position_GPIO_IO
```

- INSTANCE=opb_intc_0の

```
PORT Intr = LEDs_4Bit_IP2INTC_Irpt
```

を以下のように書き換えてください。

```
PORT Intr = Push_Buttons_Positon_IP2INTC_Irpt
```

上記はPushボタンの割り込み信号を割り込みコントローラに
接続する修正です。

７）メニューのHardware -> Generate Bitstreamを実行します。
＜実行中に下記のソースコード編集を行ってください＞

８）TestAppLab3.cは、割り込みでLEDブリンクする記述です。未完成ですので
incompleteの箇所に記述を追加し完成させてください。

９）Applicationsタブの”Sources”にTestAppLab3.cを追加してください。

１０）［７が終了しているのを確認してから］
Update Bitstream, Download Bitstreamを実行してください。
エラーがあれば、TestAppLab3.cを修正してください。
（※ソースコード以外のエラーは講師に聞いてください）

１１）Pushボタンを押すたびにLEDが点灯・消灯する事を確認してください。
Pushボタンは SW3-7のどれでも構いません。

時間に余裕がある人はDS3,11,12,13,14のLEDもブリンクさせてみてください。

１２）メニューのFile -> Close Projectでプロジェクトを終了してください

Lab4: Codec コントローラ

《このLabで習得する事》

ユーザIPの追加の仕方を習得します。

１）メニューのFile -> Open Project でlab4の下のsystem.xmpを選択してください。

２）メニューの Hardware -> Create or Import Peripheral を実行し、下記の要件を満たすユーザIPを作成してください。

- Name : opb_codec_cntlr
- Version : 1.00.a
- 使用するバス：OPB
- “User logic interrupt support”はなし
- register数：１、データ幅：32bit

（記載のない項目はデフォルト設定）

３）IP Catalogタブの”Project Repository”にopb_codec_cntlrがある事を確認し、右クリックのAdd IPでopb_codec_cntlrを追加してください。
画面中央のSystem Assembly View1上にopb_codec_cntlrがある事を確認してください。

４）source_files¥forlab4¥opb_codec_cntlr_v1_00_a¥hdl¥vhdlのVHDLファイルすべてをlab4¥pcores¥opb_codec_cntlr_v1_00_a¥hdl¥vhdlにコピーしてください。
（user_logic.vhdとopb_codec_cntlr.vhdは上書き保存してください）

５）lab4¥pcores¥…¥vhdl のcodec_if.vhdとuser_logic.vhdは未完成です。
ソースファイル中の(1)～(4)の箇所に下記の動作をする記述を追加してください。

- codec_if.vhd

(1)rx_intrにright_en1_d2==1の時1を代入、0の時0を代入

(2)pre_rx_wenにleft_en1_d1==1もしくはright_en1_d1==1の時、1を代入、それ以外は0を代入

(3)ある条件の時に pre_rx_addr に 01 を代入、pre_rx_data にright_dataを代入

(4)pcm_right に代入する記述(pcm_left の記述を参考にしてください)

○ user_logic.vhd

(1)im_rx_addr に以下のビットアサインで代入

(30 to 31) ← 00

(28 to 29) ← rx_addr(1 downto 0)

(0 to 27) ← All 0

※im_rx_addr は Bit0 が MSB です。

(2)tx_data に im_tx_data(12 to 31)を代入

(3)intr_trg にリセット時は 0,req_rx_buf==1 の時 1 を代入。req_rx_buf==0 の時は値を保持。(clock は positive edge を使用)

(4)codec_wrapper のインスタンス記述(rx_intr は rx_intr_w と接続してください)

６）source_files¥forlab4¥opb_codec_cntlr_v1_00_a¥data の MPD ファイルと PAO ファイルを lab4¥pcores¥opb_codec_cntlr_v1_00_a¥data に上書き保存してください。

７）lab4¥pcores¥…¥data の MPD と PAO は未完成です。ファイル中の incomplete の箇所に追記し完成させてください。

８）lab4 フォルダの system.mhs は未完成です。ファイル中の incomplete の箇所に追記してください。
また、ファイル一番下の opb_codec_cntlr 部分に以下を記述してください。

```
BEGIN opb_codec_cntlr                          →既に書かれている
   PARAMETER INSTANCE = opb_codec_cntlr_0      →既に書かれている
   PARAMETER HW_VER = 1.00.a                   →既に書かれている
   PARAMETER C_BASEADDR = 0x40040000
   PARAMETER C_HIGHADDR = 0x40040fff
   BUS_INTERFACE SOPB = opb
   PORT OPB_Clk = sys_clk_s
```

```
    PORT lm_bit_clk = lm_bit_clk
    PORT reset_n = sys_rst_s
    PORT lm_sdata_in = lm_sdata_in
    PORT lm_sdata_out = lm_sdata_out
    PORT lm_sync = lm_sync
    PORT lm_reset_n = lm_reset_n
    PORT im_rx_en = im_rx_en
    PORT im_rx_wen = im_rx_wen
    PORT im_rx_addr = im_rx_addr
    PORT im_rx_data = im_rx_data
    PORT rx_intr = rx_intr
    PORT im_tx_data = im_tx_data
    PORT im_tx_en = im_tx_en
    PORT im_tx_wen = im_tx_wen
    PORT im_tx_addr = im_tx_addr
END
```

９）lab4work¥data¥system.ucf は UCF ファイルです。特に編集する必要はありませんが、確認だけしておいてください。

９－５）Hardware ->Clean Hardware を実行

１０） メニューの Hardware -> Generate Bitstream を実行します。
（エラーがあれば修正してください）

＜しばらく様子をみて、ツール処理が流れているようであれば、その間に下記のＣソース修正を行ってください＞

１１）lab4¥TestApp_Peripheral¥src¥TestApp.c は未完成です。incomplete の箇所に記述を追加し完成させてください。

11－５）Software ->Clean Programs を実行

１２）[１０が終了していたら]Update Bitstream, Download Bitstream を実行してください。

１３）PC で音楽データを再生してください。音が聞こえなければ C ソースに問題があるかもしれません。見直してみてください。

Lab5: ノイズ付加

《このLabで習得する事》

IIRフィルタを利用したソフト処理が行えるようになります。

１）lab4フォルダを同じ階層にコピーし、lab5にリネームしてください。

２）TestApp.cを編集して、IIRフィルタを利用した下記のサイン波「ノイズ」を生成し、元の音楽データに付加してください。

- サイン波の周波数 5KHz
- サンプリング周波数 48KHz

ノイズ付加は割り込みハンドラ(CDC_Handler)で行ってください。

３）記述が完成したら、Update Stream, Download Streamを実行してください。音楽に混じって、異音が聞こえるはずです。

４）次はプッシュスイッチを使ってノイズのON/OFFをする記述を追加してください。

- プッシュスイッチが押されている間、ノイズOFF
- プッシュスイッチを離している間、ノイズON

プッシュスイッチからの値はGPIOを使って取得してください。
GPIOで入力値を取得する記述はテキスト第3章を参考にしてください。

５）完成したら、ダウンロードしてON/OFFできるか試してみてください。

（**おまけ**）

時間に余裕のある人はデバッガを立ち上げてBreakPoint設定やメモリを表示させながら実行したりしてみてください。

デバッガの起動：

１．メニューのDebug -> Launch XMD

２．設定画面でそのままSaveしてください

３．メニューのDebug -> Launch Software Debugger

４．開いたWindowのメニューのRun -> Connect to target

５．設定画面で以下を確認し、OKをクリック

Target：Remote/TCP:XMD

Hostname：localhost

Port：1234

６．メニューのRun -> Run

７．これでBreakPoint設定、ステップ実行、continue実行など可能です。

View -> Memoryでメモリを表示できます。

また、View -> Local Variablesでローカル変数の表示が可能です。