VHDLとCPLDを用いたカスタムロジック回路の製作事例

VHDL と CPLD を用いたカスタムロジック回路の製作事例

豊田朋範
分子科学研究所　装置開発室

１．はじめに

従来、ロジック回路の設計は、論理ゲート回路やフリップフロップ、カウンタなどの機能を持つ標準ロジック IC を組み合わせて実現してきた。近年では、FPGA や CPLD 等のプログラマブルデバイスを用いることで回路を小さな面積に集積できるだけでなく、動作周波数のより高い回路に対応することも可能である。また、回路修正は配線の変更や IC の追加を必要とせず、回路が大規模になるほど有利である。

筆者はハードウェア記述言語の1つである VHDL を用いてカスタムロジック回路の開発を行っている。本稿でハードウェア記述言語、特に VHDL とプログラマブルデバイスの概要、並びに DDS のシリアル通信制御とスイッチ入力処理に CPLD を用いたオートコリレータ用発振器の製作について報告する。

２．ロジック回路製作手法の相違

従来のロジック回路は、入出力の相関関係を示すカルノー図を描き、それを基に回路図を描いてから回路基板を製作することで構成する(図 1)。回路情報はすべて回路図に集約されるため、回路図は必須である。

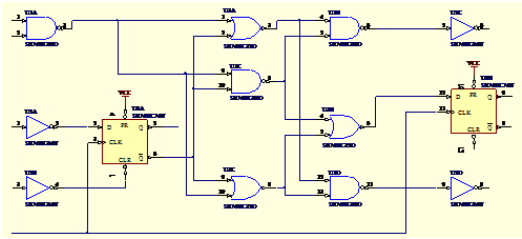

図 1：回路図によるロジック回路設計の例

標準ロジック IC はセカンドソースが豊富で、故障時も IC の交換ですぐに対応できる。また、回路の機能や信号の流れを理解しやすい。一方、回路の修正は配線の変更や IC の追加で行うが、大規模・複雑な回路ほど手間がかかる。

対してハードウェア記述言語では、PC 上でプログラミングのように回路を設計し(図 2)、コンパイルして作成した回路構成ファイルを後述のプログラマブルデバイスに転送することで構成する。回路情報は入出力のみで良いため、ブラックボックス化できる。ピン配置は電源と GND を除いて自由に設定でき(図 3)、回路の修正はソースの変更で行えるので、仕様変更に対して柔軟に対応できる。また、回路を適用するデバイスは開発中でも変更できる。デバイスが製造中止などで入手できなくても、別のデバイスに適用すれば良い。

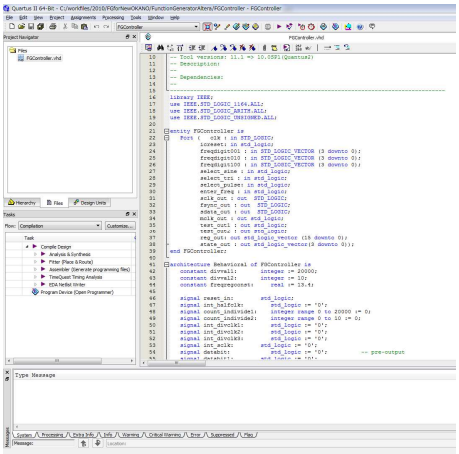

図 2：ハードウェア記述言語によるロジック回路設計の例

反面、デバイスが高価であり、形状は表面実装のみであるため実装や交換が難しい。また、ブラックボックスに出来ることで製作者以外は回路の把握

や修正が難しくなる。

回路図設計とハードウェア記述言語設計を比較したものを表 1 に示す。ハードウェア記述言語は習得が難しく、書き込み器の導入コストが必要であるが、回路の秘匿性、変更に対する柔軟性、基板専有面積などの面で有利である。

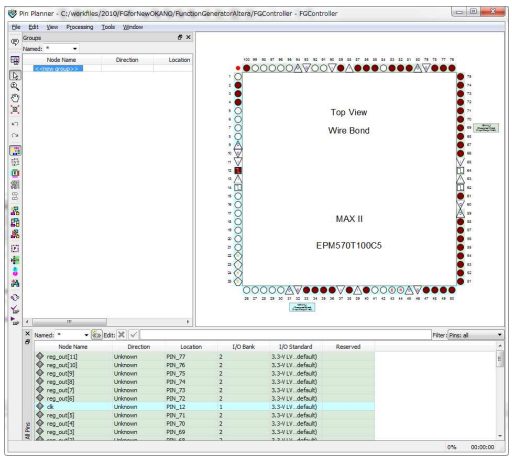

図 3：ハードウェア記述言語設計におけるピン配置設定の例

表 1：ロジック回路製作手法の比較一覧

	ハードウェア記述言語	回路図
回路図の製図	任意	必須
回路の秘匿性	高い	低い
回路の変更	ソースリストの変更	配線の変更や IC の追加
占有面積	小さい	回路規模に比例する
同期回路の最大動作周波数	数十 MHz 以上	20MHz 程度
習得のしやすさ	やや難しい	容易
導入コスト	やや高価。書き込み器が必須	安価。IC があれば構築可

３．ハードウェア記述言語とプログラマブルデバイス

3.1　ハードウェア記述言語－特に VHDL の概要と特徴

アメリカ国防総省で用いられる ASIC(特定用途向け集積回路)の開発には数年を要する一方、半導体製造プロセスは日進月歩の速度で飛躍的に進歩するため、開発中に最も高速なデバイスを想定していても、製造する頃には時代遅れのプロセスを採用することになる二律相反が顕在化してきた。その二律相反を解消するための方策として開発されたのが VHDL(VHSIC Hardware Description Language)である。

VHDL は 1981 年に国防総省の VHSIC(Very High Speed Integrated Circuit)委員会で提唱されたのを皮切りに、1983 年に仕様書作成が始まり、1985 年に作業が完了した。1986 年にはマニュアルがまとめられ、以降国防総省が調達するすべての ASIC は VHDL 記述付きでの納入が義務付けられた。その後、1986 年に IEEE(アメリカ電気電子技術者協会)における標準化作業が始まり、1987 年 12 月に承認された [1]。

シミュレータ専用言語として開発され、やはり IEEE で標準化作業を経て承認された Verilog-HDL と比較して、VHDL は入出力や変数に型宣言が必須であるなど文法が厳格である点が特徴である。

そのためテンプレートを作りやすく、記述による回路構成の違いが生じにくいなどの長所があるが、記述が煩雑になりやすいなどの短所もある。また、演算に応じたライブラリの導入が必要である。

3.2　プログラマブルデバイスー特に CPLD と FPGA の概要と特徴

CPLD(Complex PLD)と FPGA(Field Programmable Gate Array)は、どちらもハードウェア記述言語で製作したロジック回路を書き込んで使用するプログラマブルデバイスである。最大の相違点は、CPLD では回路構成ファイルを不揮発性のメモリ(Flash や EEPROM)に書き込むのに対し、FPGA は揮発性のメモリ(SRAM)に書き込む点である[2]。これは、CPLD が初期のプログラマブルデバイスである GAL(Generic Array Logic)を集積したものに相当するのに対し、FPGA はフリップフロップやゲートなどごく小規模な回路機能ブロックを組み合わせるという構造上の違いによる[2]。

CPLD はあまり大規模で複雑な回路は構成できないが、電源を切っても回路情報が消えず、回路情報の記録に外部素子が不要なためセキュリティの面で有利である。一方 FPGA は電源投入時に回路情報を転送するための外部 ROM を必要とするが、単純な構成ゆえに大規模化しやすく、PLL や DSP など高機能な回路や複雑な演算機能を内蔵できる。

CPLD や FPGA は当初 ASIC の試作に用いられていたが、製品開発サイクルの短縮や少量多品種生産への移行など民生分野でも専用 IC 製造コストの低減需要が高まったことから、現在では回路全般に対する小型集積需要への対応の他、画像・音声データの高速処理、暗号化処理を伴うルータをはじめとするネットワーク・通信機器など、幅広い分野で使用されている。

4.VHDL と CPLD を用いたカスタムロジック回路製作事例－オートコリレータ発振器－

4.1　装置の目的

筆者がこれまでに製作した VHDL と CPLD を用いたカスタムロジック回路の中から、オートコリレータ発振器(図 4)について述べる。

オートコリレータ発振器は、1～199Hz 程度の低周波でスピーカを振動させてレーザー機構を微調整する装置である。光路の変動が実験精度に大きく影響するため、周波数精度の高い安定性が求められる。

図 4：オートコリレータ発振器

4.2　装置の概要

従来このような低周波発振回路においては、
NE555 など RC 発振器 IC を使用していた。今回は周波数の高い安定性が求められることから、発振器 IC は AD9833BRM(Analog Devices 社)を用いた。

AD9833BRM は 2.3V～5.5V 単一電源で動作し、DDS 方式により最高 12.5MHz の周波数を 28bit の分解能で出力できる。3 線式 SPI インターフェースを採用しており、送信用クロック sclk、イネーブル fsync、送信データ sdata を所定のタイミングで操作することで内部 16bit レジスタにデータを転送し、初期化や周波数設定、波形選択を行う。

オートコリレータ発振器のブロック図を図 5 に示す。制御回路は、周波数設定を行う 3 桁サムホイールスイッチの値(1～199Hz)を AD9833BRM の内部レジスタ用のデータに変換し、Enter スイッチで転送する。また、波形選択スイッチの状態でサイン波、方形波、三角波を選択し、所定のデー

タを AD9833BRM に転送する。

制御回路は電源投入後、図 6 の状態遷移図に従って装置の初期化を順次行う。まず INIT1 で 16bit×8 個のデータを転送して AD9833BRM を初期化する。PAUSE1 で 12 クロック分待機した後、INIT2 で周波数設定に使用する内部レジスタ番号を指定する 16bit データを転送する。PAUSE2 で再び 12 クロック分待機した後、FREQ で現在の 3 桁サムホイールスイッチの値を 16bit×2 個のデータに変換して転送する。

次に、PAUSE3 で 12 クロック分待機した後、現在の波形選択スイッチの状態で SINE_PRE→SINE(サイン波)、PULSE_PRE→PULSE(方形波)、TRI_PRE→TRI(三角波)のいずれかに分岐する。分岐後はそれぞれ波形を指定する 16bit データを転送する。最後に PAUSE4 での 12 クロック分待機を経て READY に遷移して装置の初期化を完了する。以降は Enter スイッチと波形選択スイッチを監視し、操作すると所定の状態遷移を行い READY に戻る。

一般的にこのような回路はマイコンで構成するが、VHDL と CPLD で構成すると、前述したように大規模で複雑なロジック回路を 1 つの IC に集積でき、回路の修正が容易になる。他にも(1)高速転送が可能である、(2)開発言語の環境依存性が低い(C 言語はしばしば開発環境ごとに異なるヘッダファイルの読み込みや関数の変更が必要とされる)といったメリットもある。

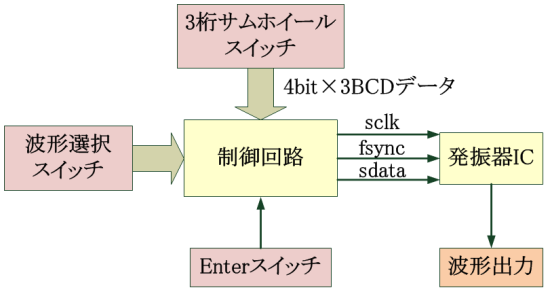

図 5：オートコリレータ発振器のブロック図

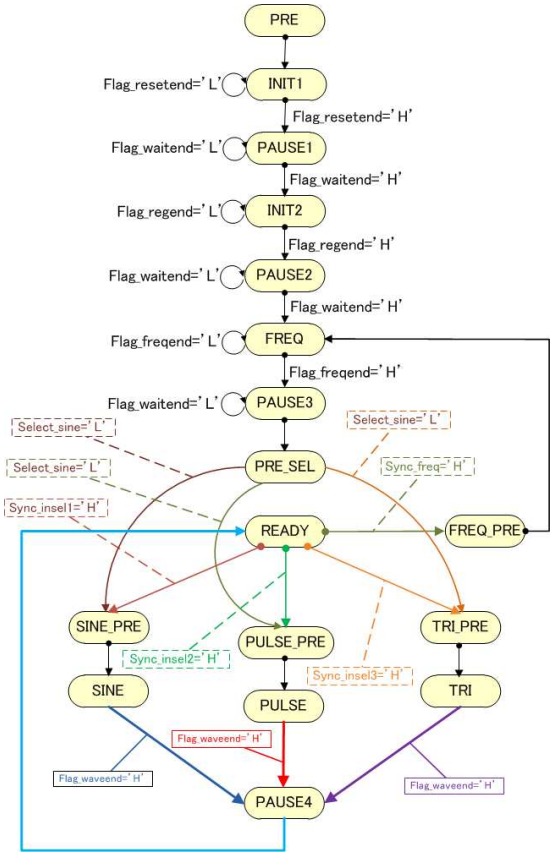

図 6：オートコリレータ発振器の状態遷移図

4.3　CPLD キットと VHDL 開発環境

今回用いた CPLD キットを図 7 に示す。CPLD キットは内部回路の容量が 570 ロジックエレメントで基板サイズが小型の EPM570T100C5N(Altera 社)を使用した。当初は 144 マクロセルの XC95144XL-TQ144-10C(Xilinx 社)を使用していたが、後述する周波数設定値のデータ変換処理回路を組み込んだところ内部回路の容量が不足したため、EPM570T100C5N に切り替えた。

入出力や動作の決定は Altera 社が提供する無償の IDE(統合開発環境)である Quartus Ⅱ で VHDL を用いてプログラミングし、コンパイルして作成した回路

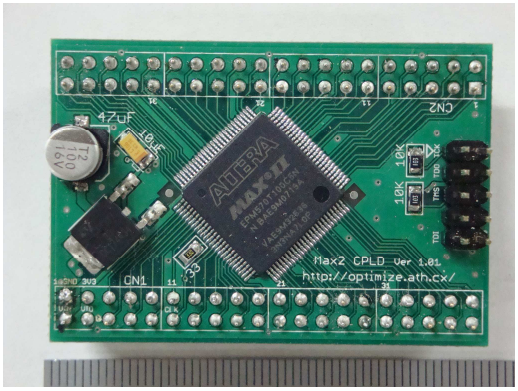

図 7：使用した CPLD キット

構成ファイルを JTAG インターフェースで CPLD に転送する(書き込む)ことで行う。PC と CPLD の接続にはソリトンウェーブ社の MAX Ⅱ マイクロキットを用いた。

4.4　VHDL によるシリアルデータ転送回路の記述

図 6 に沿って順次データを転送し、スイッチの位置によって異なる状態に遷移する回路はステートマシンで構成している。AD9833BRM のリセットデータ、内部レジスタ番号選択データ、周波数設定値の内部レジスタ変換データ、波形選択データを転送するためにそれぞれ process 文による回路ブロックを構成しており、ステートマシンで制御する内部フラグの変化を受けて動作することで、所定のデータを順次転送する。

128bit シリアルデータ転送用 process 文の一部を図 8 に示す。パワーオンリセット入力 reset_in かリセット処理を行う内部フラグ flag_reset が 0 になると、内部カウンタ count_databit1 を 127 にプリセットし、かつ sclk に接続する内部信号 pre_sclk を 0 にする(227～229 行)。flag_reset が 1 になると、システムクロック clk の立ち上がりごとに pre_sclk を反転して(232 行)、count_databit1 が 0 であれば 0 で固定し(233～234 行)、0 でなければ pre_sclk1 が 1 の時に 1 つダウンカウントする(236～238 行)。あらかじめ定義した 128bit データにおいて count_databit1 の値で示される位置の bit を sdata とすれば、clk を 1/2 分周した sclk に同期して sdata が順次出力されるシリアルデータ転送回路が実現できる。

```
225    -- Send Reset sequence
226    process (reset_in, flag_reset, clk) begin
227    if (reset_in = '0' or flag_reset = '0') then
228        count_databit1 <= 127;
229        pre_sclk1 <= '0';
230    elsif (flag_reset = '1') then
231        if (clk'event and clk = '1') then
232            pre_sclk1 <= not(pre_sclk1);
233            if (count_databit1 = 0) then
234                count_databit1 <= 0;
235            else
236                if (pre_sclk1 = '1') then
237                    count_databit1 <= count_databit1 - 1;
238                end if;
239            end if;
240        end if;
241    end if;
242    end process;
```

図 8：VHDL による 128bit シリアルデータ転送回路

この回路による初期化データの出力波形を図 9 に示す。図中 sysclk がシステムクロック 40MHz、pwreset はパワーオンリセット入力、state1～3 は現在のステートを 3bit で示すデバッグ用ステート出力である。

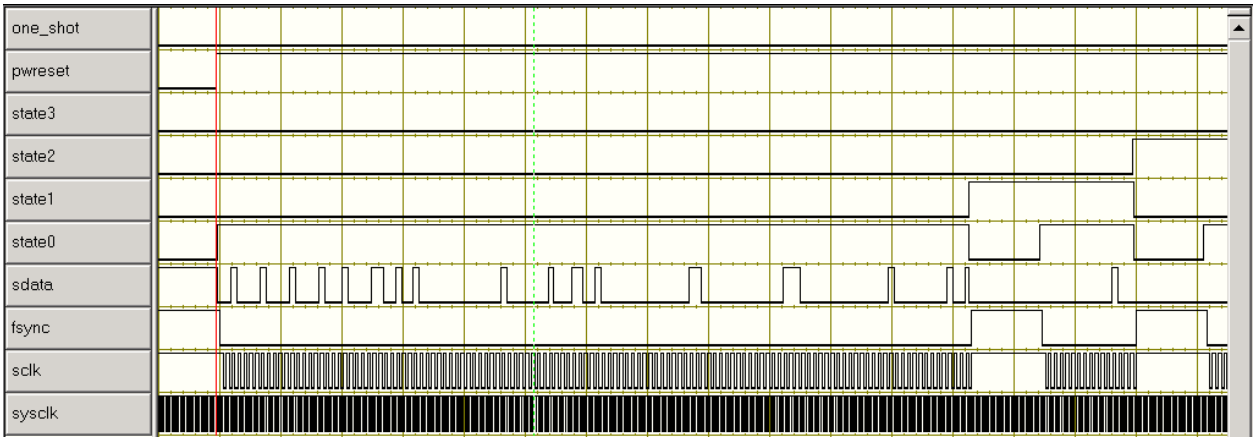

図 9：構築したシリアルデータ転送回路による初期化データの出力波形

電源投入で pwreset 入力が'L'になると回路がリセットされ、ステート出力は'000'、すなわち PRE になる。pwreset が'H'に戻るとステート出力は'001'、すなわち INIT1 に移行して初期化が開始される。fsync が'L'になり、sclk に同期して 128bit の sdata を順次出力する。完了するとステート出力は'010'、すなわち PAUSE1 になり、sclk の 12 クロック分待機する。続いてステート出力は'011'、

すなわち INIT2 になり、sclk に同期して周波数設定に使用する内部レジスタ番号を sdata で順次出力する。これらはすべて図 7 の状態遷移図に則っている。

ステート出力はデバッグを容易にするために付加したものであり、デバッグ完了後は VHDL ソースでピンへの接続をコメントアウトして無効にした。このように回路の追加や無効化を容易に行えるのはハードウェア記述言語の大きなメリットであり、従来のようにオシロスコープのプローブを当てて信号の推移を観察できないプログラマブルデバイスのデバッグ手法の 1 つである。

５．まとめ

ディジタル回路の大規模化・高機能化が進むにつれて、標準ロジック IC の組み合わせ以外の製作手法の必要性が高まってきた。ハードウェア記述言語設計の導入により、プログラミングのようにディジタル回路の製作や修正が可能になった。

習得が難しく、専用の書き込み器が必要であるため導入コストはやや高価といった難点はあるが、高速高機能ディジタル回路の集積をはじめ、秘匿性、仕様変更に対する柔軟性などハードウェア記述言語の習得メリットは大きい。

これまでに従来のディジタル回路の置換・集積から、状態遷移図に則った複雑なシーケンス動作をする同期回路まで、さまざまな製作実績を蓄積してきた。今後もディジタルフィルタや FFT など複雑な演算を伴う高速回路を製作するなど、ハードウェア記述言語による回路設計技術の向上を図っていく所存である。

６．参考文献

[1] 長谷川裕恭：「VHDL によるハードウェア設計入門」(1998), p14-15

[2] 日本アルテラ株式会社：「FPGA 入門」http://www.altera.co.jp/products/fpga.html