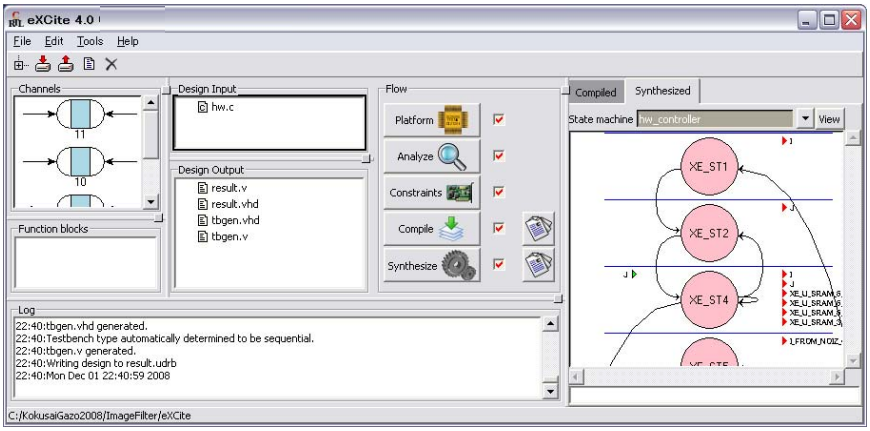
Soliton

eXCite C言語設計事例

Vol：001

eXCite を用いた基本設計編
「Avalon インタフェースを持った回路設計」



C言語高位合成 eXCite を用いてハードウェアへ実装する手順をご紹介します。

eXCite はC言語を入力として、論理合成可能な RTL HDL を生成するC言語高位合成ツールです。本資料では、ANSI-C で記述されたソートアルゴリズムを例題に、アルテラ社 Avalon インタフェースを持った回路に実装する一連の設計ステップをご紹介します。

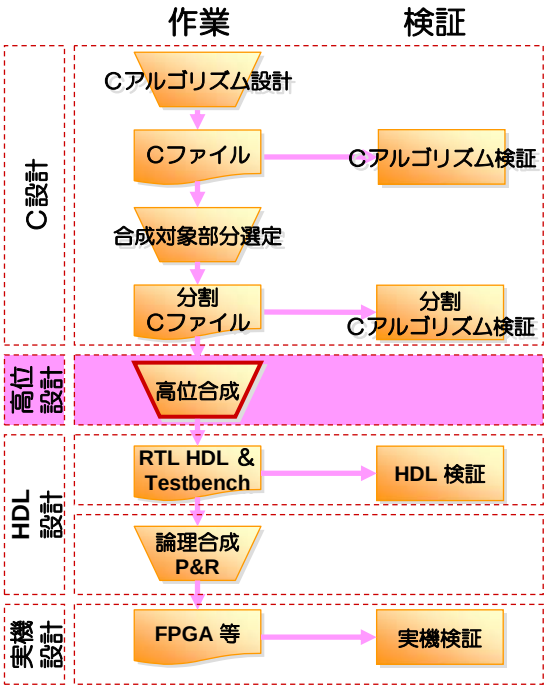
■ 設計ステップ

C言語でハードウェアを設計する場合、図１にあるようなステップを経て設計を行います。eXCite で設計する場合、C言語上で アルゴリズムの設計＆検証を行った後、合成対象部分の選択と分割Cファイルによる動作検証を行います。

Cアルゴリズムとして問題が無ければ高位合成を eXCite で行い、生成された RTL HDL とテストベンチを用いて HDL シミュレータで検証し、合成結果の品質や内容を確認します。

出来上がる HDL はCアルゴリズムの機能モジュールですので、それを必要な周辺回路と接続し、通常の論理合成、配置配線とハードウェア設計を進めることになります。なお、今回ご紹介のアルテラ社 Avalon インタフェースを生成した場合、周辺回路との接続はアルテラ社 SOPC Builder で行うことになり、HDL を書くこと無しに設計を進めることも可能です。

図１：設計ステップ



■ オリジナルソースコードの確認

ハードウェアへの第一歩はC言語上での分割検討から始まります。合成対象部分のどの変数が外部と通信する必要があるのか、またその通信をどの様なインタフェースにするかを決める必要があります。合成対象部分を決定したら、その記述が参照している変数を確認します。

今回の例題である「func_quick_sort」関数（図２）は外部とのやり取りが必要な変数は関数の引数である配列変数「sort_data[SORT_SIZE]」のみであることが分かります。今回はこの配列変数を通信するインタフェースを検討することにします。

■ Cファイルの分割

オリジナルのCアルゴリズムから合成対象部分を切り出します。合成対象部分は eXCite で合成するために図３にあるように３つの約束ごとがあります。まず、処理部分は「while(1)」内に記述すること。次に「通信が必要な変数に合成指示子を設定」すること、そして通信 API が宣言されている「#include "yx_bcl.h"」を呼ぶことです。

今回は、通信が必要な配列変数をメモリ系通信として指示し、同期用の信号として「start」、「done」という変数に同期系通信を割り付けています。同期用通信は、通信位置を固定するために「tmp=start」、「done=1」のような記述をしています。

次に切り出した側であるオリジナルのCファイル向けには、図３の変数宣言一覧と同様の記述から通信用 API を生成するコマンドがある。生成した API は合成部分記述を抜き出した場所へ記述することで分割後のオリジナルCファイルが完成する。また、図３の記述からもコマンドにより API が記述された分割Cシミュレーションファイルを作成することができ、それぞれのCファイルを Visual C++などのCコンパイラでコンパイルし、実行ファイルし動作を確認できる。

なお、分割C検証を行うと eXCite が RTL HDL 検証向けに生成するテストベンチのテストベクタが自動的に作成される。これを用いることでCでの動作との差異や性能比較が可能となる。

図２：クイックソート関数例

```
void func_quick_sort(unsigned short sort_data[SORT_SIZE])
{
    int axis, lp, rp, left, right, depth, mem_right[32];
    unsigned short low=0, high=0;

    left = 0;
    right = SORT_SIZE-1;
    depth=0;

    while(1){
        axis = sort_data[ (left + right)>>1 ];
        lp = left-1;
        rp = right+1;

        while(1){
            while(1){
                lp++;
                if( (high=sort_data[lp]) >= axis) break;
            }
            while(1){
                rp--;
                if( (low=sort_data[rp]) <= axis) break;
            }

            if(lp >= rp) break;
            sort_data[lp] = low;
            sort_data[rp] = high;
        }
        if(sort_data[left] == sort_data[right]){
            depth--;
            left = right+1;
            if(left>=SORT_SIZE) break;
            right = mem_right[depth];
        }else{
            mem_right[depth] = right;
            depth++;
            right = rp;
        }
    }
}
```

図３：eXCite 入力用C記述

```
#include "yx_bcl.h"

void main(void)
{
    //yx channel_transform 1 message blocking read
    unsigned char start;
    //yx channel_transform 2 message blocking write
    unsigned char done;
    //yx channel_transform 10 memory
    unsigned short sort_data[SORT_SIZE];

    unsigned char tmp;

    while(1){
        tmp=start;
        func_quick_sort(sort_data);
        done=1;
    }
```

■ C高位合成

eXCite の高位合成エンジンは、通信合成エンジン、IP 再利用データベース、および各種最適化などを同時に行うことで、C記述でも高品質なハードウェアの合成を実現しています。ツールの操作は、基本的にボタンを上から順に押していくだけです。合成後には、論理合成可能な VHDL、Verilog を生成します。また、同時にテストベンチも生成します。

■ 通信合成

eXCite は入力記述で指定した各変数の ID に通信制約を割り付けることで各種インタフェースを自動生成することができます。図５の例では図３で設定した ID にアルテラ社の Avalon インタフェースを割り付けた例です。合成後にはアルテラ社 SOPC Builder で読込みかつ接続可能な機能 IP が生成されます。

■ テストベンチを用いた RTL 検証

eXCite は合成対象記述を検証するためのテストベンチを自動生成します。このテストベンチは、VHDL または Verilog で記述され、特殊な環境を必要としません。各種 HDL シミュレータで動作を確認できます。シミュレーションが終了すると、図６にあるようにソフトウェアと同じ処理をした場合のサイクル数やソフトウェア結果との差異の有無などがレポートされています。これにより、合成結果の等価性と品質を確認できます。

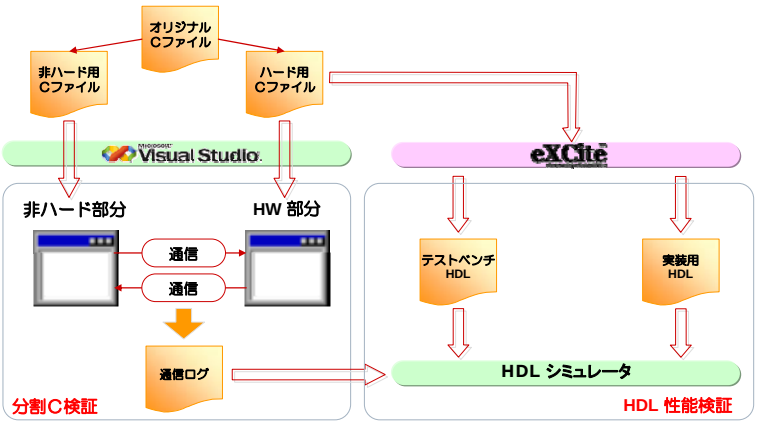


図４：eXCite 合成エンジン

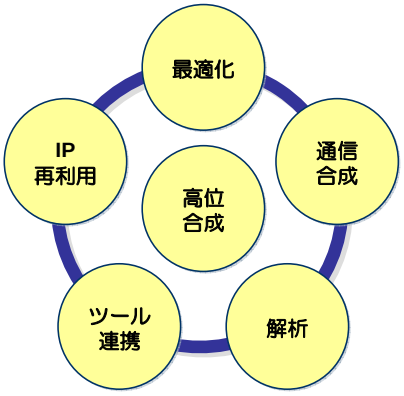


図５：通信合成制約

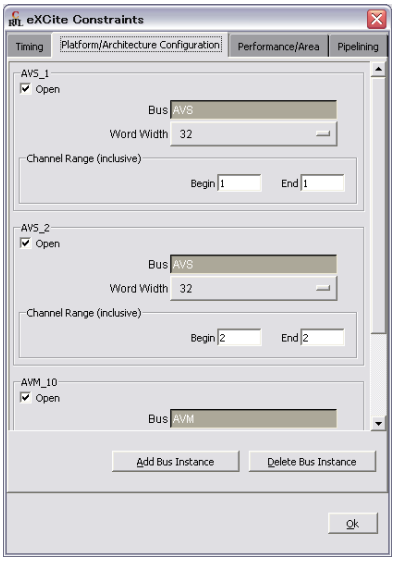
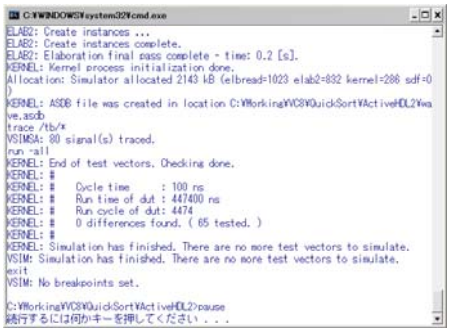


図６：HDL シミュレーション結果



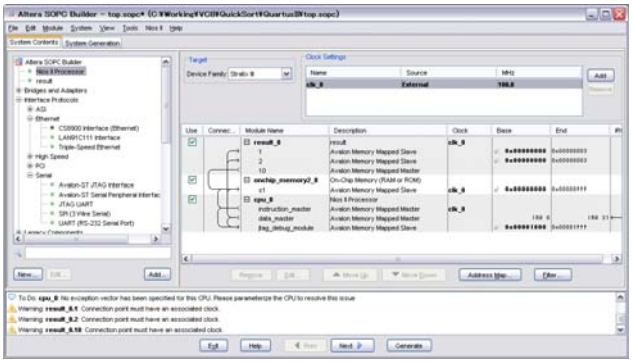
■ Avalon インタフェースと SOPC Builder

Avalon インタフェースを持った回路を合成した場合、アルテラ社 SOPC Builder で部品として読み込むことができます。SOPC Builder はアルテラ社が提供する様々なペリフェラル IP と容易に接続することができます。これにより、HDL 設計無しの FPGA 設計も可能となります。図7は SOPC Builder の画面ですが、今回合成したモジュールを result という名前で部品化し、メモリなどに接続しています。

■ まとめ

eXCite が提供する環境は、ソフトウェア設計者でもハードウェアを設計可能にする様々な工夫があります。まず、特殊なC言語ではなく標準のC言語を採用しています。また、ハードウェアを確実に設計するために、設計と検証をステップ毎に行えます。デバイスや FPGA への実装向けには Avalon インタフェースなどを利用することで HDL 設計レスの設計も可能です。これらの技術により、今まで FPGA 設計は難しいとあきらめていた方もハードウェア設計より身近なものにすることができます。

図7：SOPC Builder へ読み込み



本資料に関するお問い合わせ：
株式会社ソリトンシステムズ
組み込みシステム部
eXCite 担当

Email：at@soliton.co.jp
TEL:：+81-3-5360-3851