XMOS の XS1-G4 であそぶ

テスト基板 XC-1 で遊んで気がついたことのメモを書いていきます。

・XC-1 は XMOS Semiconductor 社 の XS1-G の評価基板です。

・Design Wave Magazine の 2009/3,4 合併号に XS1-G についての特集記事（下記参考）が載っています。

http://www.cqpub.co.jp/dwm/Contents/0136/dwm013600710.pdf

XC 言語を使った XS1-G4 の具体的な使い方

いくつかの Tutorial でも説明されているのですが、いままで XMOS サイト内で見つけた中でいまのところ一番体系だって、同期、シリアライズ、デシリアライズ、timed I/O の説明がされているのは、 XMOS トップから、Support > Documentation > Software Tools のページにある、XC-XS1 User Manual (Preview) (2009/02/19) でした。

★現在、上記の PDF は、Programming XC on XCore XS1 Devices という名前になっているようです。Support > Documentation > Design Tools のところにページ構成が変わりました。

なんか LED のポートが資料 (XC1 Board Layout and Port Map) と違う気が ..... (古い情報)

2009/3 現在の XC-1 では、以下のようになってるみたいですよ。
追記：新しい資料では訂正されていますが、Tutorial はふるい説明のままの物もある模様。

```
out port cled0 = XS1_PORT_4C; /* 12,1,2,3 */
out port cled1 = XS1_PORT_4A; /* 4,5,6,7 */
out port cled2 = XS1_PORT_4B; /* 8,9,10,11 */
```

なんか石が熱いんすけど ……

使わないコアやリンクのパワーを制御できるようです。
出典：xlinkers forum http://www.xlinkers.org/node/65
の より。

```
//  setRegVal(processor,psctl,reg,val);
setRegVal(0, 0, 7, 128);  // CPU 0 - Low power link mode (Reg 7)
setRegVal(1, 1, 6, 128);  // CPU 1 - Low power mode (Reg 6)
setRegVal(2, 1, 6, 128);  // CPU 2 - Low power mode (Reg 6)
setRegVal(3, 1, 6, 128);  // CPU 3 - Low power mode (Reg 6)
```

このへんのことは、XMOS サイトの Support > Documentation > Silicon にある、"XS1 System Specification (2009/01/27)" という資料の 3.6 章 (コマンドの発行方法)、4.2 章、4.3 章 (レジスタの内容) に記述があります。

外付け SPI ROM からブートしたい (XC-1, 古い情報)

※ XMOS から SPI ROM がついたバージョン、XC-1A が出ました (2009/12)

ファームウェアバージョン 1.1 以降では、ブート（USB 接続）時に B ボタンを押下すると、SPI ブートするという記述があります。

ファームウェアバージョンは、1.1 以降のファームウェアであれば、USB 接続直後、デモ起動前に点灯する LED で判別できます。(下記は XC-1 Development Kit Addendum から引用)

> You can check the firmware version installed when you connect the XC-1 to the USB cable. The button LEDs light up to indicate the major version number and the clock LEDs indicate the minor number. For example, version 1.1 is identified when Button A LED and Clock LED 1 light up. If no LEDs light up the firmware is version 1.0.

SPI FLASH メモリの接続方法、プログラムのフォーマットは、前述の XS1 System Specification の第２章にかかれています。

SPI FLASH(ROM) といってもいろいろあります。XMOS の豪華版のほうのキット、XDK では EPCS1(Altera の FPGA コンフィグレーション用) が使われていますので、これは確実に使えそうです。

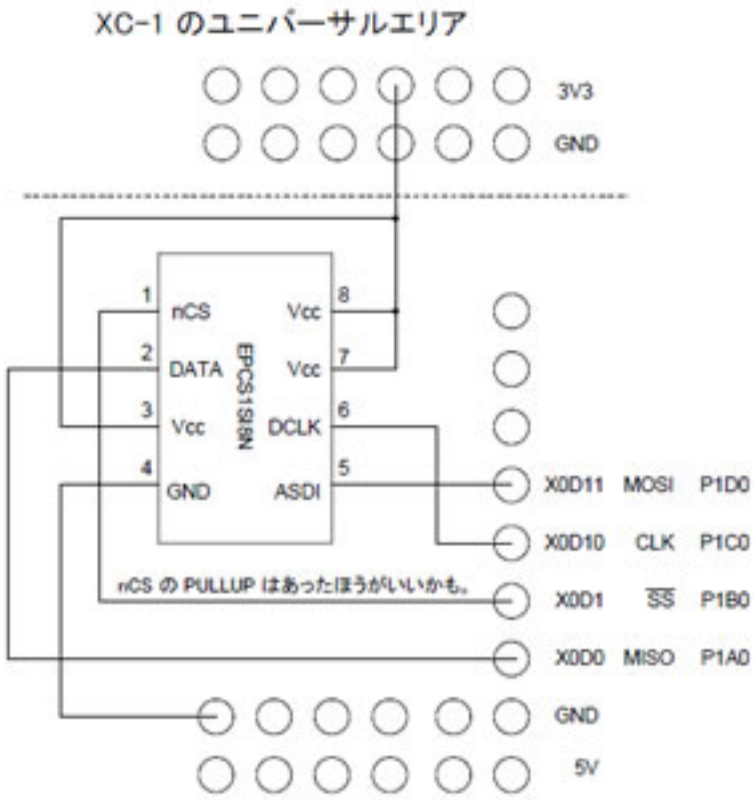
他にも Xlinkers フォーラムを見ていると、AT25F512 や M25PE10 などの名前が出てきます。データシートを見ると、たしかにコマンドは互換っぽいです。

とりあえず、安全パイの EPCS1 を Digikey で購入して、書き込み、SPI ブートしてみました。

★新しく出た XC-2 Ethernet キットでは、Atmel の AT25DF041A が使われています。これは EPCS1 よりも安い上に、容量も大きい (4MBit) ですので、これから買うならこちらでしょうか。

SPI FLASH への書き込みツールですが、Desktop Tools 9.2.0 をインストールすると、xflash.exe なるコマンドが一緒についてきます。コレを使うと、難しいことを考えずに接続すると、そのまま XC-1 を FLASH ライタとして使えます。

実体配線図（電源のパスコン (104) と nCS のプルアップが書いてないです。プルアップはなくても動きました）。



上記のように接続し、xflash を使って書き込むと、以下のようなメッセージを出して書き込みができました。

```
C:\Program Files\Xmos\DesktopTools\9.2.0>xflash  c:\work\xc1\tut1\Debug\tut1.xe
Building flash images...
Programmer started
Erase sectors:
 3...
 Programming page at address:
  0x000800...
SPI flash programming completed successfully.

C:\Program Files\Xmos\DesktopTools\9.2.0>
```

また、SPI ROM を実装せずに (XC-1 は接続状態) 実行すると、以下のように出ました。

```
C:\Program Files\Xmos\DesktopTools\9.2.0>xflash  c:\work\xc1\tut1\Debug\tut1.xe
Building flash images...
Programmer started
Failed to initialize access to SPI flash.
Error: F03013 Failed to complete flash programming.
```

さて、SPI に書き込めたわけですが、B ボタンを押しながら USB を挿すと、たしかに書き込んだアプリケーションが起動 ..... するはずが、するときとしないときがあります。

いろいろ調べていると、Core0 のみの .xe は起動するのですが、複数のコアを使う .xe は SPI に書き込めるものの、B ボタンでは起動しません。

xlinkers で聞いて見た所、

```
To boot multicore from flash, you need to move resistor R11 onto R12.
This will set the other cores to boot from flash.
```

？？？ R11？？？ R12 そんなレジスタあったっけ？
と数分考えて、回路図の R11 と R12 (抵抗 =resistor!=registor だ !)
ということに気づき、早速 R11（SS XC0 BS0=H でブート種別 JTAG) を R12 (SS XC0 BS0=L でブート種別 SPI) に変更。

```
★この変更 (R11->R12) のあと、何度かデバッグをやってみて、
マルチコアアプリとそうでないアプリで、xrun や IDE からのデバッグが
動作したりしなかったり、ということがありました。
なので、現在は、スイッチをつけて、SS XC0 BS0=L/H を切り替え
できるようにしました。


★ xrun 失敗 , デバッグモードに入れない現象が出る場合には、
必ず USB 接続時に A ボタン押下（プログラムモード）で A ボタン脇の
LED の 3 回点滅を確認するようにするとうまくいくようです。
```

しかし、XC-1 単独でブートするには、まだいくつかの壁があります。

ためしに携帯充電用の、+5V 電源だけきている、「ニセ USB」をつないでみましたが、起動しませんでした。

XC-1 の回路図をみると、USB バスパワーのスイッチングが FTDI の USB-Serial&JTAG の石で行われている (Q4) ので、単純に USB のところから 5V を入れるだけでは駄目で、FTDI の石がホストとのリンクを確立しないと電源が供給されないようです。

また、Q12 が FTDI の石につながっていて、これがリセットラインを引っ張っているようです。なので、このへんもなんとかしないとヘッダから 5V 入れても動かないかもしれません。

（追記）Xlinkers によると、Q4,Q12 を単純に取り除くだけで外部電源仕様になるようです。USB も動作するらしい（未確認）↓
http://www.xlinkers.org/forum/viewtopic.php?f=4&t=355

X2Port[A-D] のヘッダはどこにつながってるの？

回路図を見ると、H16,H17,H18,H19 となっていますが、ユニバーサルエリアの X0Dnn みたいなシルク印刷がないので、一瞬？？？となってしまいました。

どこかで見たような気がしたので、探してみたら、XC-1 の製品ページのわきにある XC1 Tutorial(xport1.pdf) の最後の方の "XC1 Board Layout and Port Map" の core0 のポート表 (core0 の LED のポートが間違ってるやつ) の次のページの Figure 5:XC-1 Block Diagram の中に割り当てがありました。

H16:X2PortA
H17:X2PortB
H18:X2PortC

H19:X2PortD

のようです。

また見失わないように、メモしておきます。
LED と同じく、ずれているかもしれませんが .....

violates variable disjointness rules って？

XC 言語では、異なるスレッドから同じメモリへのアクセスをしようとすると、このエラーになるようです。

あんまり意味のないコードですが、例 .....

```
#include <xs1.h>
#include <platform.h>

out port cledO = XS1_PORT_4C; /* 12,1,2,3 */
out port cledG = XS1_PORT_1E;
out port cledR = XS1_PORT_1F;

unsigned share_mem[10] ; /* 共有メモリ */

/* スレッド 1 */
void produce()
{
  unsigned a;
  while(1) {
    for (a =0;a<10;a++){
      share_mem[a] = a ; // ここと
   }
  }
}

/* スレッド 2 */
void consume()
{
  unsigned a ;
  cledG <: 1 ;
  cledR <: 1 ;
  while(1){
    for (a=0;a<10;a++) {
      cledO <: share_mem[a] ; // ここと
    }
  }
}

int main()
{
  par { // ここと で、怒られる
    produce() ;
    consume() ;
  }
  return 0;
}
```

スレッド間通信はチャンネルを使え、ということらしいです。

でも、配列とか両方でアクセスできないとイヤな場合

C ファイルには、上記の制限がない（チェックがきかない？）ようです。
なので、上の配列の共有部分を C で別ファイルに、下記のように書いて、

```
unsigned share_mem[10] ;

int readmem(unsigned addr)
{
  return share_mem[addr] ;
}

void setmem(unsigned addr,unsigned data)
{
  share_mem[addr] = data ;
}
```

XC 側で、以下の宣言をして、

```
extern unsigned share_mem[10] ;
extern int readmem(unsigned addr) ;
extern void setmem(unsigned addr,unsigned data) ;
```

メモリアクセスを以下のように書き換えると、おこられなくなります。

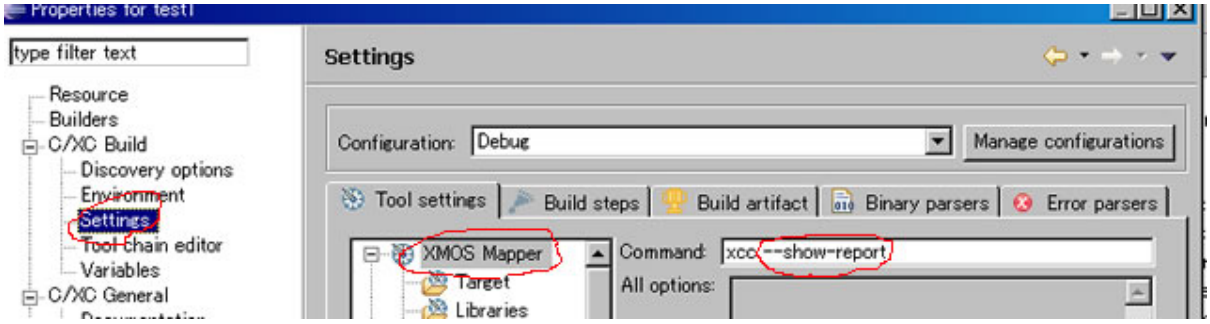
```
void produce()
{
  unsigned a;
  while(1) {
    for (a =0;a<10;a++){
      setmem(a,a) ; // ここと
    }
  }
}

void consume()
{
  unsigned a ;
  cledG <: 1 ;
  cledR <: 1 ;
  while(1){
    for (a=0;a<10;a++) {
      cled0 <: readmem(a) ; // ここ
    }
  }
}
```

コンパイル時に各コアのリソースの使用状況を知る方法 (古い情報)

Desktop Tools 9.2.0(Eclipse) の場合、Project > Properties の中の、以下の Xmos Mapper の xcc のところに "--show-report" オプションをつける（画像参照)。

★本当は Mapper ツリーの Misc のところに書くのが正しいような気がしないこともないかも :-)



XMOS の XS1-G4 であそぶ ｜ XMOS の XS1-G4 であそぶ - その 2 ＞＞