GP6MSPC-001
初版: 2012 年 5 月 1 日

GP600M実習環境説明書

国立大学法人奈良先端科学技術大学院大学情報科学研究科
コンピューティング・アーキテクチャ講座
低電力高性能プロセッサ研究グループ

目 次

図 目 次

表 目 次

Chapter 1

GP600Mの物理構成

1.1 概要

図 1.1 に GP600M の外観，図 1.2 に GP600M のブロック図を示す．各ユニットへは 25MHz のクロックが供給されており，XC2V6000 内部については，DCM モジュールにより 50MHz 等の逓倍クロックを生成して使用することができる．以下，各ユニットについて説明する．

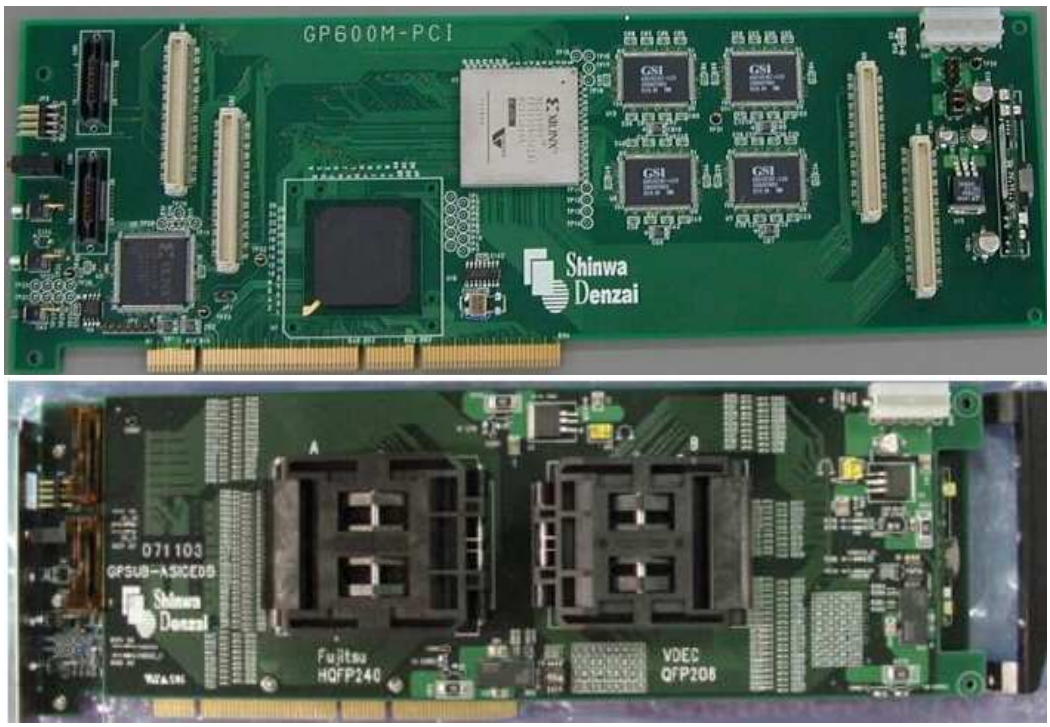

図 1.1: GP600M の概観（下はドータボード搭載時）

大規模 FPGA（XC2V6000）

大規模 FPGA は，CLK ジェネレータ，ロジックアナライザ接続コネクタ，波形観測用タップ，コンフィグレーション用 CPLD，ZBT-SSRAM，PCI ブリッジに接続されている．また，ドータボード搭載モデルでは，ドータボード上の LSI ソケットと接続されている．

CLK ジェネレータ

FPGA のピン番号および HDL 最上位階層 top における信号名は，AK17:CLK_25M である．

ロジックアナライザ接続用 MICTOR コネクタ（CN5，CN6）

括弧内は，FPGA のピン番号および HDL 最上位階層 top における信号名である．

CN5 CLK0（AM2:MONACLK1），A3-7（AM7:MONA3(7)），A3-6 （AP6:MONA3(6)），A3-5

（AM6:MONA3(5)），A3-4（AP5:MONA3(4)），A3-3（AN5:MONA3(3)），A3-2（AP4:MONA3(2)），A3-1（AN4:MONA3(1)），A3-0（AN3:MONA3(0)），A2-7（AL9:MONA2(7)），A2-6（AJ9:MONA2(6)），A2-5（AH9:MONA2(5)），A2-4（AN8:MONA2(4)），A2-3（AM8:MONA2(3)），A2-2（AL8:MONA2(2)），A2-1（AP7:MONA2(1)），A2-0（AN7:MONA2(0)），CLK1 （AM9:MONACLK0），A1-7（AJ11:MONA1(7)），A1-6（AG11:MONA1(6)），A1-5（AN10:MONA1(5)），A1-4（AL10:MONA1(4)），A1-3（AK10:MONA1(3)），A1-2（AJ10:MONA1(2)），A1-1（AP9:MONA1(1)），A1-0（AN9:MONA1(0)），A0-7（AG12:MONA0(7)），A0-6（AF12:MONA0(6)），A0-5（AE12:MONA0(5)），A0-4（AP11:MONA0(4)），A0-3（AN11:MONA0(3)），A0-2（AM11:MONA0(2)），A0-1 （AL11:MONA0(1)），A0-0（AK11:MONA0(0)）

CN6 CLK0（AH12:MONBCLK1），A3-7（AH13:MONB3(7)），A3-6（AG13:MONB3(6)），A3-5（AF13:MONB3(5)），A3-4（AE12:MONB3(4)），A3-3（AP12:MONB3(3)），A3-2（AN12:MONB3(2)），A3-1（AM12:MONB3(1)），A3-0（AL12:MONB3(0)），A2-7（AG14:MONB2(7)），A2-6（AF14:MONB2(6)），A2-5（AE14:MONB2(5)），A2-4（AP13:MONB2(4)），A2-3（AM13:MONB2(3)），A2-2（AL13:MONB2(2)），A2-1（AK13:MONB2(1)），A2-0（AJ13:MONB2(0)），CLK1 （AK14:MONBCLK0），A1-7（AH15:MONB1(7)），A1-6（AG15:MONB1(6)），A1-5（AF15:MONB1(5)），A1-4（AE15:MONB1(4)），A1-3（AP14:MONB1(3)），A1-2 （AN14:MONB1(2)），A1-1（AM14:MONB1(1)），A1-0（AL14:MONB1(0)），A0-7（AJ16:MONB0(7)），A0-6（AH16:MONB0(6)），A0-5（AG16:MONB0(5)），A0-4（AE16:MONB0(4)），A0-3（AD16:MONB0(3)），A0-2（AP15:MONB0(2)），A0-1（AM15:MONB0(1)），A0-0（AL15:MONB0(0)）

デジタルオシロ接続用タップ（ボード裏面 **TP**）

括弧内はFPGAのピン番号およびHDL最上位階層topにおける信号名である．TP11（A31），TP12（B31），TP13（B32），TP14（C33），TP15（C2:DO_TP15），TP16 （B3:DO_TP16），TP17（B4:DO_TP17），TP18（A4）

コンフィグレーション用 **CPLD**

XC2V6000のコンフィグレーション制御用CPLD．コンフィグレーション用コネクタ（JP3）またはフラッシュROMからのコンフィグレーション機能を提供する．ただし通常はホストPCからのコンフィグレーションを行うため，JP3は使用しない．JTAGコネクタ（JP2）はPLD専用につきユーザは使用不可．

ZBT-SSRAM

GSI社のGS8320Z36．36bit幅× 4Mワードを2組搭載．並置し，64bit幅× 4Mワードの外部メモリ機構として使用する．HDL最上位階層topにおける信号名は，SSRAMA_A(0-19)，SSRAMA_BW(0-3)，SSRAMA_D(0-35)，SSRAMA_LBO，SSRAMA_OE，SSRAMA_RW，SSRAMB_A(0-19)，SSRAMB_BW(0-3)，SSRAMB_D(0-35)，SSRAMB_LBO，SSRAMB_OE，SSRAMB_RW，SSRAM_CE(0-7) である．

PCI ブリッジ（QL5064）

PCI64Bit/66Mhz インタフェースのためのブリッジ機能．PCIバスマスター機能を有し，独立する2chのDMAを搭載．内部の概要を図1.3に示す．なお，PCIブリッジとXC2V6000を接続する汎用バスをローカルバスと呼ぶ．詳細はQuickPCI_Localブリッジ IPコア仕様書 第1版を参照．HDL最上位階層topにおける信号名は，L_AD(0-63)，L_ADSX，L_BEX(0-7)，L_BSTMX，L_DACKX，L_DBSYX，L_DRQX，L_INTX，L_RDYIX，L_RDYOX，L_RSTX，L_S64X，L_WEX である．

ドータボード接続コネクタ

GP600Mには，ドータボード接続用拡張I/Oピンが装備されている．計4個の高密度コネクタによりドータボードと接続可能．各140ピン中I/Oは100ピン（I/O合計400ピン）．I/Oピン番号は共通で，11-20, 23-32, 35-44, 47-56, 59-68, 71-80, 83-92, 95-104, 107-116, 119-128.

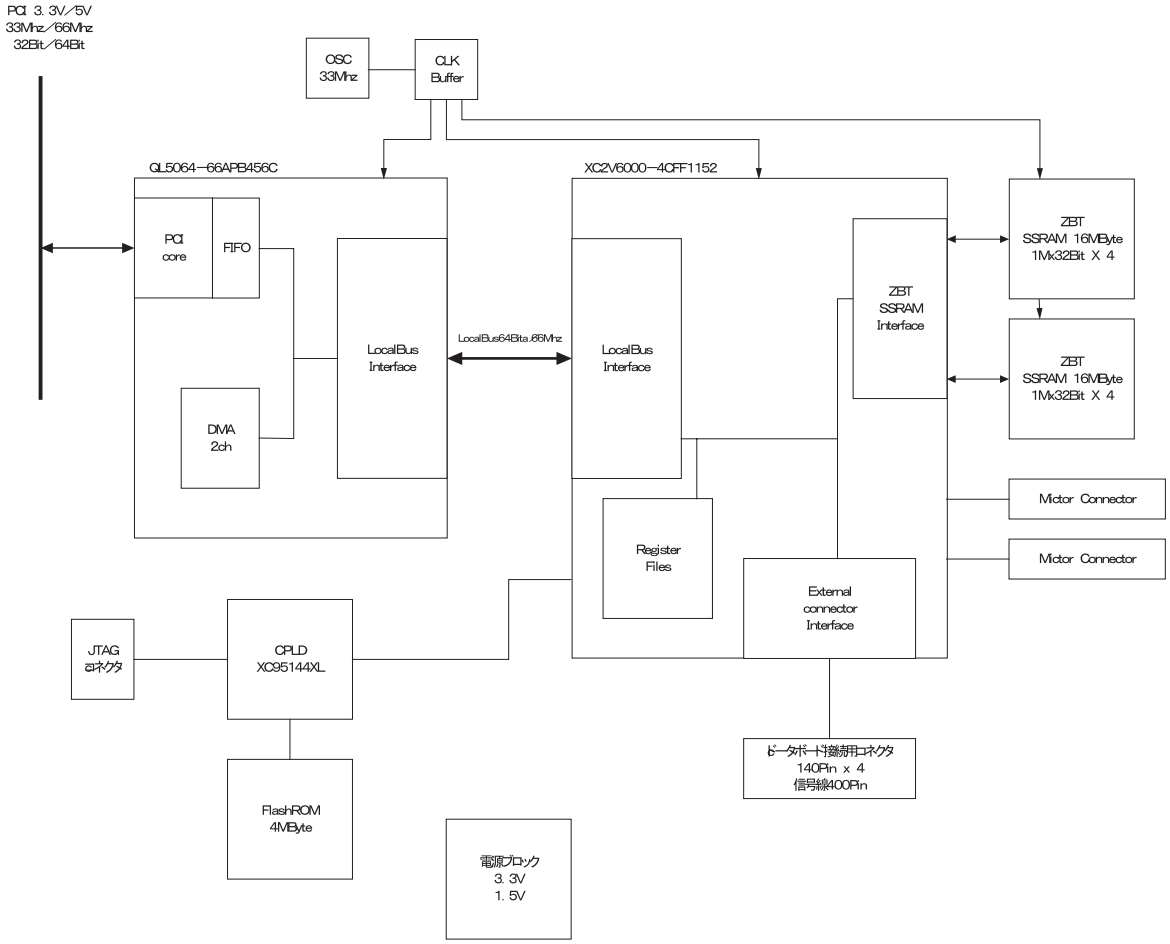

図 1.2: GP600M のブロック図

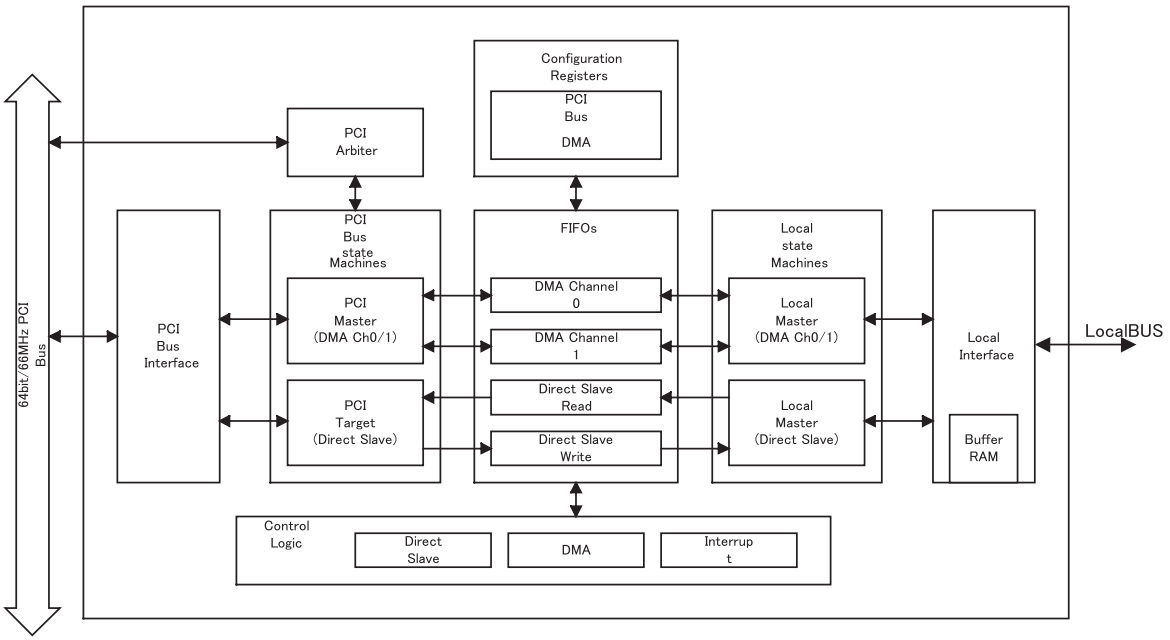

図 1.3: XC2V6000 と PCI のインタフェースユニット

1.2 XC2V6000 と ZBT-SSRAM の接続

GP600M は 36bit × 1M ワードの ZBT-SSRAM を計 8 個搭載しており，36bit × 4M ワード× 2 組の外部メモリとして使用できる．表 1.1 に HDL 最上位階層 top との接続，図 1.4 にタイミングチャートを示す．

表 1.1: ZBT-SSRAM の接続

ZBT-SSRAM 側信号	説明
A(入力)	SRAMA/B_A19-0 に接続（アドレス信号）
BA-BD(入力)	SRAMA/B_BW3-0 に接続（バイト毎書き込み，Active-LOW）
W(入力)	SRAMA/B_RW に接続（書き込みイネーブル，Active-LOW）
E1(入力)	SRAM_CE7-0 に接続（チップイネーブル，Active-LOW）
G(入力)	SRAMA/B_OE に接続（出力イネーブル，常時 LOW）
ADV(入力)	LOW 固定（常時アドレスセット可）
CKE(入力)	LOW 固定（クロック入力バッファ常時イネーブル）
ZZ(入力)	LOW 固定（パワーダウン未使用）
FT(入力)	N.C.（Pipeline モード）
LBO(入力)	SRAMA/B_LBO に接続（リニアバースト，常時 LOW）
CK(入力)	CLK_25M に接続（クロック）
DQA-DQD(入出力)	SRAMA/B_D35-0 に接続

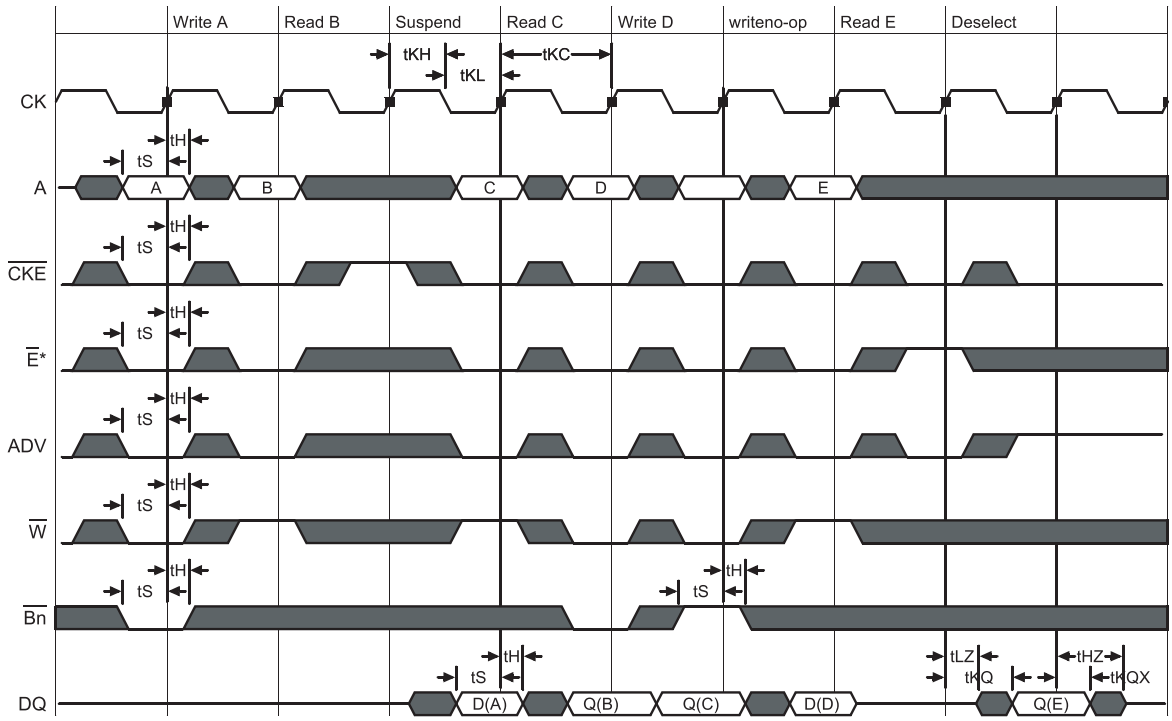

図 1.4: ZBT-SSRAM のタイミングチャート

1.3 XC2V6000 と PCI ブリッジの接続

ホスト PC は，PCI ブリッジを経由して，ZBT-SSRAM や FPGA 内レジスタを参照することができる．表 1.2 に HDL 最上位階層 top との接続，図 1.5 および図 1.6 にタイミングチャートを示す．

表 1.2: PCI ブリッジの接続

HDL 最上位階層信号	説明
L_RSTX(出力)	リセット信号
L_DRQX(出力)	ブリッジからの転送要求信号
L_DACKX(入力)	転送要求に対する応答（LOW の場合 Local 側がスレーブ，HIGH の場合 Local 側がマスタ）
L_ADSX(入出力)	転送開始（バスマスタが出力）
L_WEX(入出力)	LOW は WRITE，HIGH は READ に対応
L_BSTMX(入出力)	LOW はバーストモードに対応
L_RDYOX(出力)	ブリッジ側転送準備 OK
L_RDYIX(入力)	Local 側転送準備 Ok
L_AD(入出力)	ブリッジ側マスタの時はアドレス＋データ，Local 側マスタの時はアドレス＋バイト数＋データ
L_BEX(入出力)	バイトイネーブル
L_INTX(入力)	割り込み信号
L_S64X(入力)	PCI 制御レジスタ選択信号
L_DBSYX(出力)	PCI 書き込み状態の表示（ただし，特に使用しなくてよい）

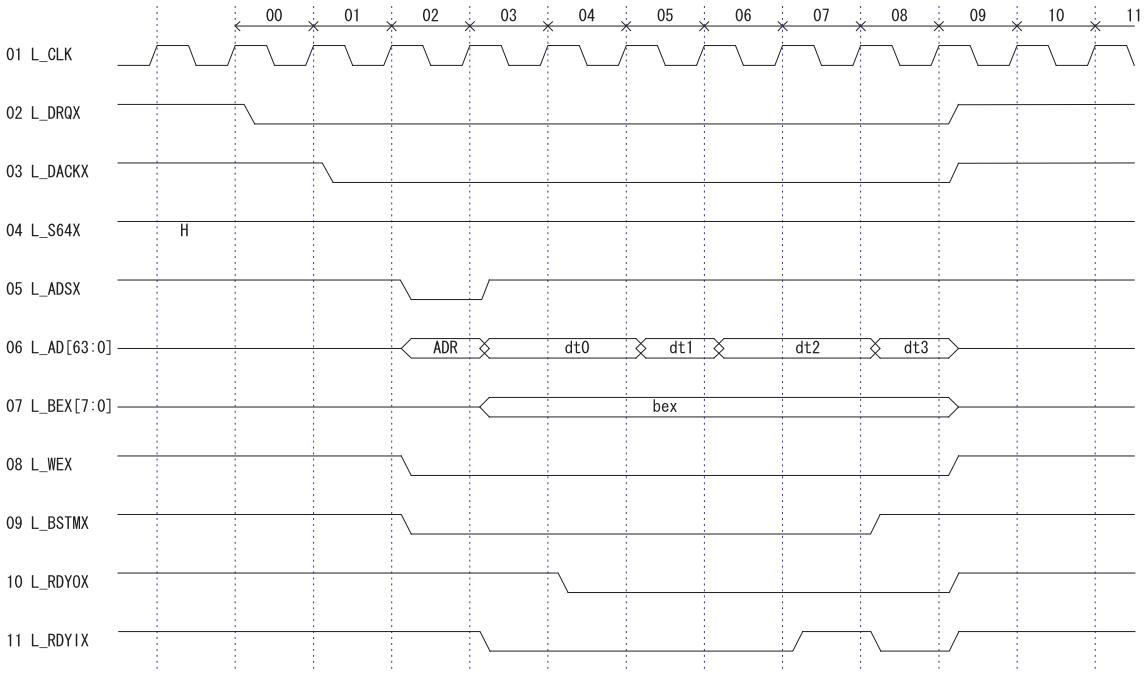

図 1.5: PCI ブリッジ⇒ Local-BUS の書き込み

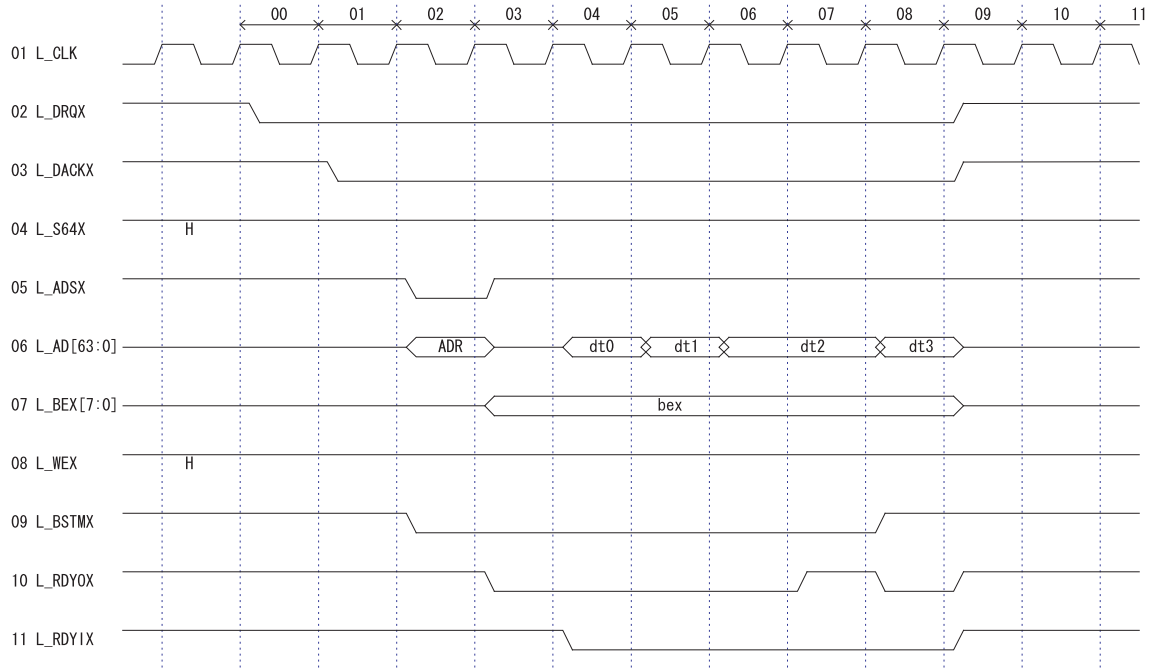

図 1.6: PCI ブリッジ⇒ Local-BUS の読み出し

1.4 ホスト PC による ZBT-SSRAM 直接参照

前述の PCI ブリッジにより，ホスト PC の主記憶空間に ZBT-SSRAM を対応付け，ホスト PC が ZBT-SSRAM を直接参照できる．図 1.7 から図 1.10 に，PCI ブリッジによる ZBT-SSRAM 直接参照のタイミングチャートを示す．図 1.11 は，PCI ブリッジから ZBT-SSRAM までの信号を網羅したタイミングチャートである．

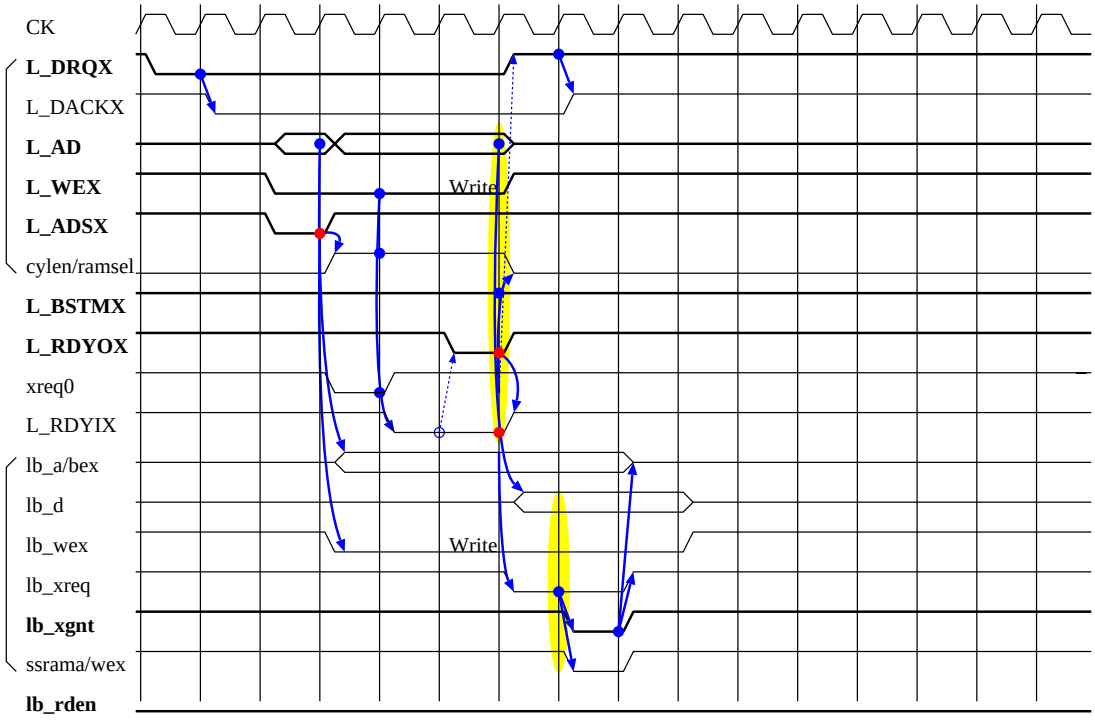

図 1.7: ホスト PC による SSRAM 直接参照 (WRITE)

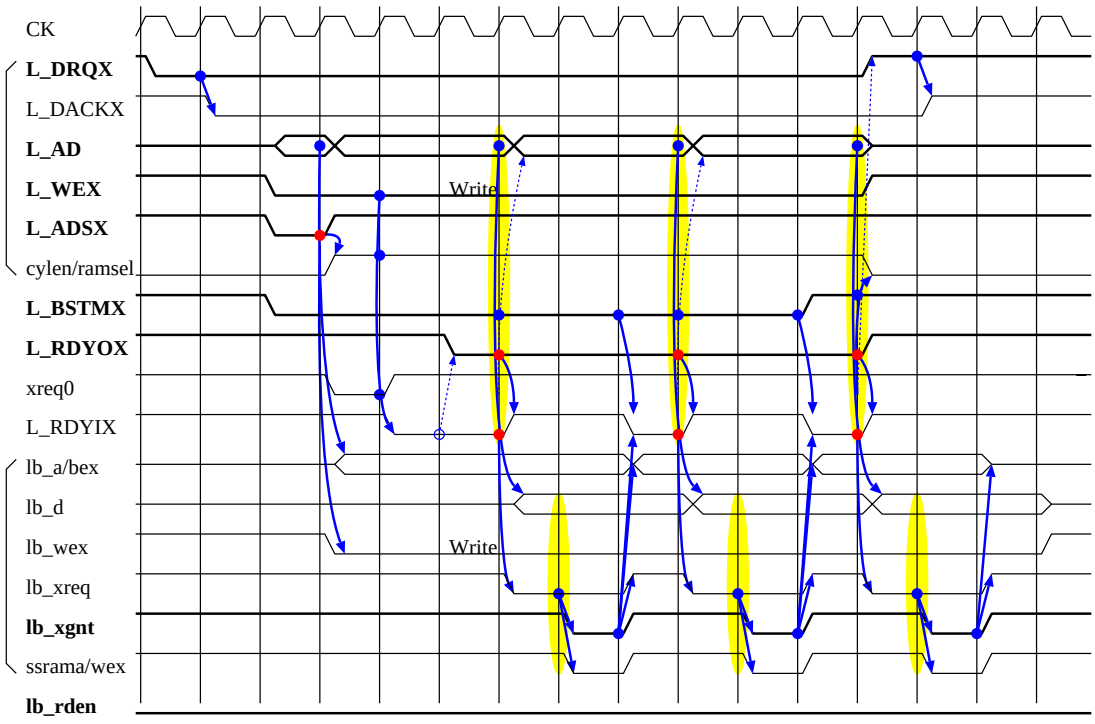

図 1.8: ホスト PC による SSRAM 直接参照 (WRITE バーストモード)

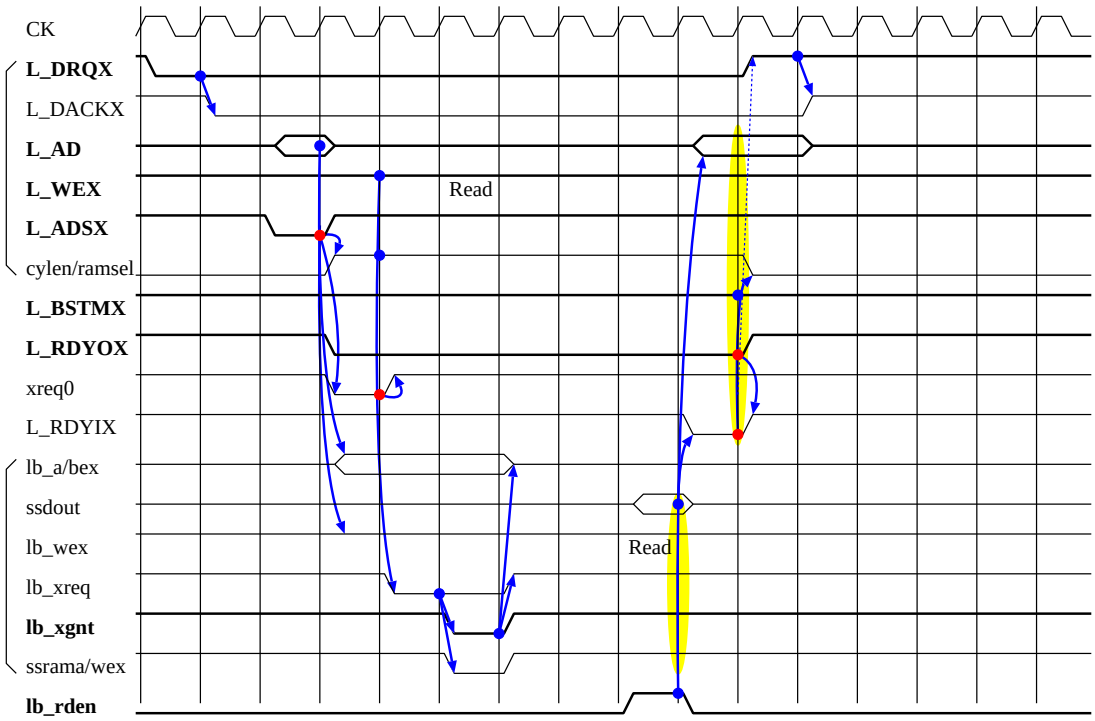

図 1.9: ホスト PC による SSRAM 直接参照 (READ)

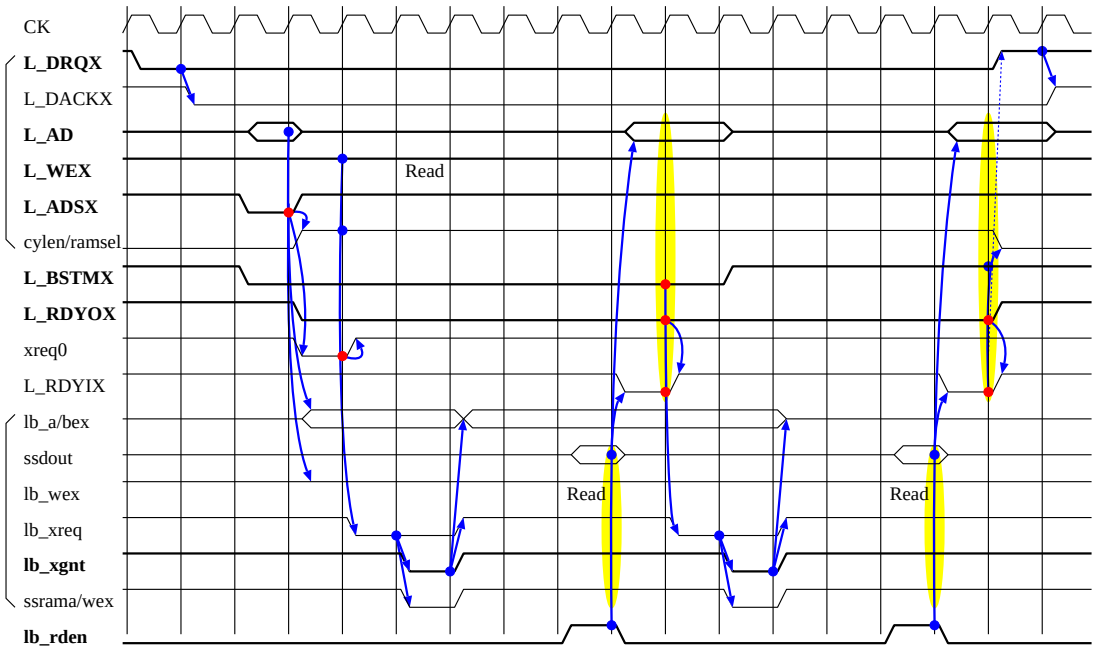

図 1.10: ホスト PC による SSRAM 直接参照 (READ バーストモード)

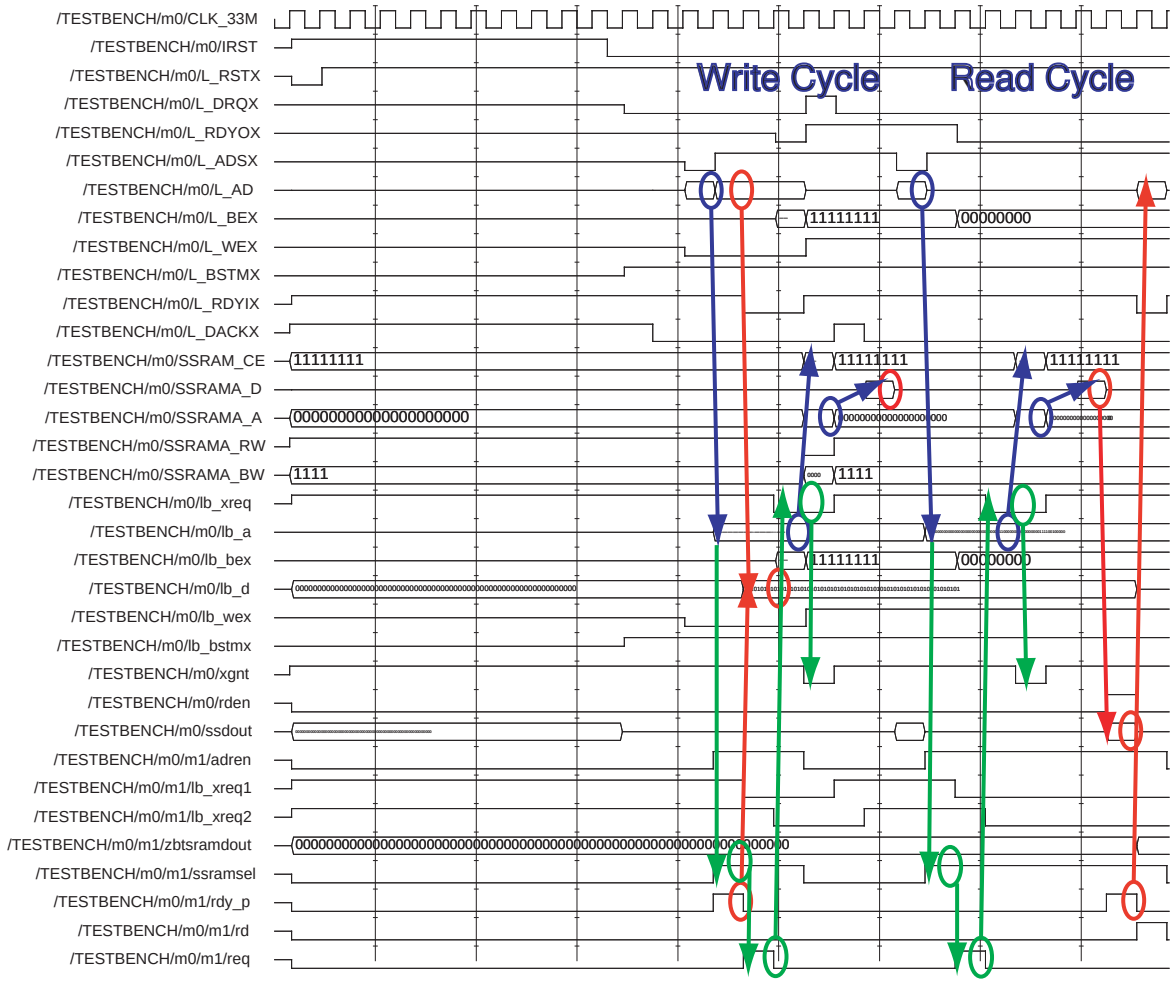

図 1.11: ホスト PC による SSRAM 直接参照の全体

Chapter 2

FPGA内部の構成

2.1 概要

FPGA 内部は，ユーザ論理回路部（HDL 階層 pe0）と，インタフェース部から構成されている．概要を図 2.1 に示す．

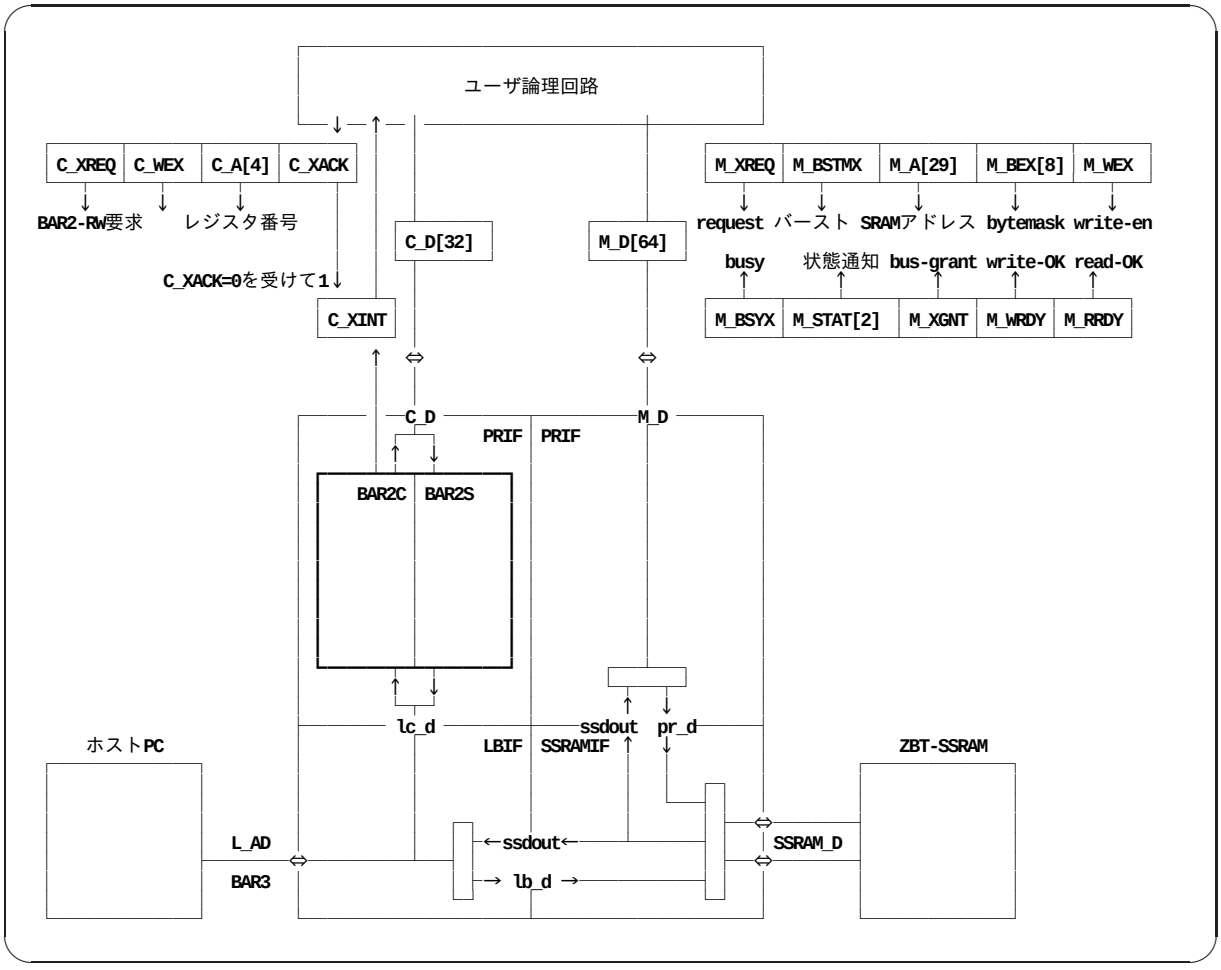

図 2.1: FPGA 内部のインタフェース

2.2 ユーザ論理回路と ZBT-SSRAM のインタフェース

表 2.1 に ZBT-SSRAM 側からみたユーザ論理回路部のインタフェース，図 2.2 から図 2.5 にタイミングチャートを示す．ビット幅や未使用信号の取扱いに関する差異を除き，表 1.1 に示した ZBT-SSRAM 物理インタフェースと同様である．ただし，ハンドシェイクのための信号 BSYX/XREQ/XGNT/WRDY/RRDY が追加されている．ZBT-SSRAM への書き込み時，ユーザ論理回路内のメモリ制御部は，BSYX 信号が OFF であることを確認した上で A/B/W/D に有効値を出力し XREQ 信号を ON にする．次に XGNT 信号の ON を観測し XREQ 信号を OFF にする．ZBT-SSRAM への書き込み完了と同時に WRDY 信号は ON，BSYX 信号は OFF となる．同様に，ZBT-SSRAM からの読み出し時，ユーザ論理回路内のメモリ制御部は，BSYX 信号が OFF であることを確認した上で A/W に有効値を出力し XREQ 信号を ON にする．次に XGNT 信号の ON を観測し XREQ 信号を OFF にする．読み出し完了と同時に RRDY 信号は ON，BSYX 信号は OFF となり，メモリ制御部は D 上の読み出しデータを取り込む．なお，ZBT-SSRAM に対する書き込み要求および読み出し要求の単位は，データ信号線幅と同じく 64 ビットである．

表 2.1: ユーザ論理回路部-ZBT-SSRAM インタフェース信号（110 本）の概要

信号名	説明
M_XREQ(入力)	ZBT-SSRAM 参照要求信号，Active-LOW
M_XGNT(出力)	要求受付完了信号，Active-LOW
M_WEX(入力)	書き込みイネーブル，Active-LOW
M_BSTMX(入力)	LOW 時はバーストモード
M_WRDY(出力)	書込完了信号，Active-HIGH．本信号の ON をもってユーザ論理回路は参照要求を消去できる
M_RRDY(出力)	読出データ valid 信号，Active-HIGH．本信号の ON をもってユーザ論理回路は参照要求を消去できる
M_A(入力)	アドレス，A31-3
M_BEX(入力)	バイト毎書き込みフラグ，Active-LOW，B7-0
M_D(入出力)	データ，D63-0
M_BSYX(出力)	動作中を示すビジー信号，Active-LOW
M_STAT(出力)	状態表示信号，STAT1-0（0:empty 2:OP-ok 3:IF-ok）

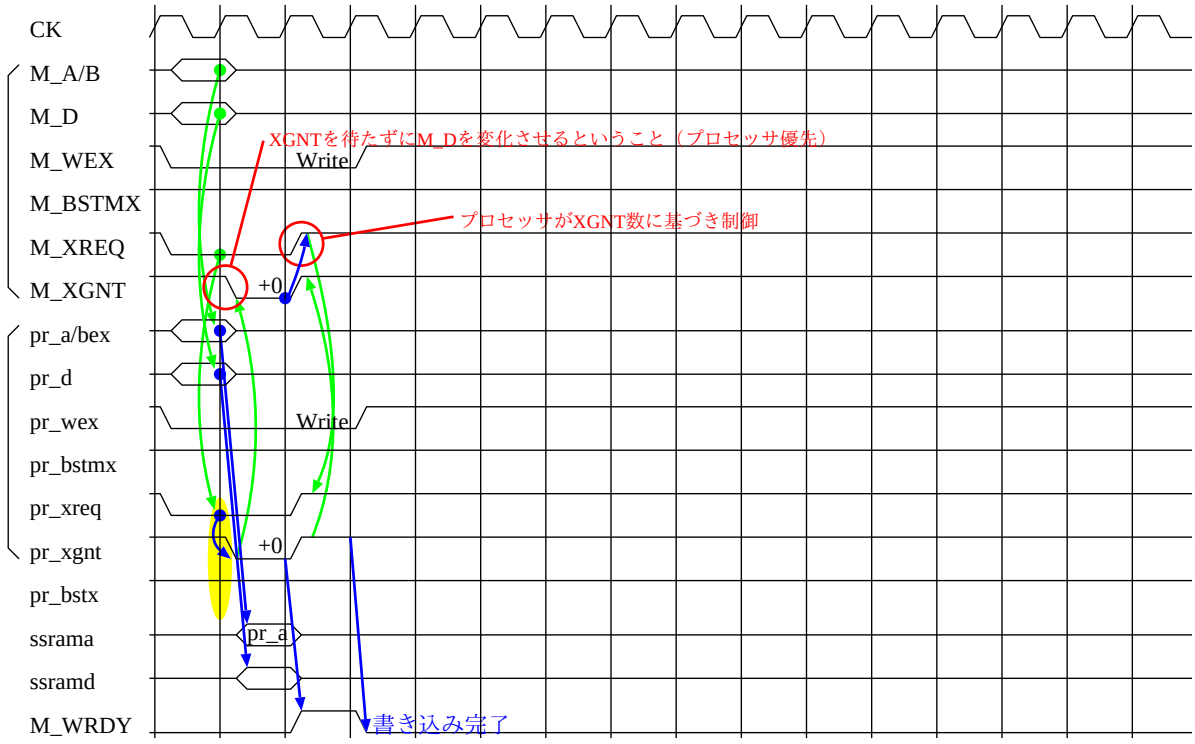

図 2.2: ZBT-SSRAM インタフェース信号のタイミングチャート (WRITE)

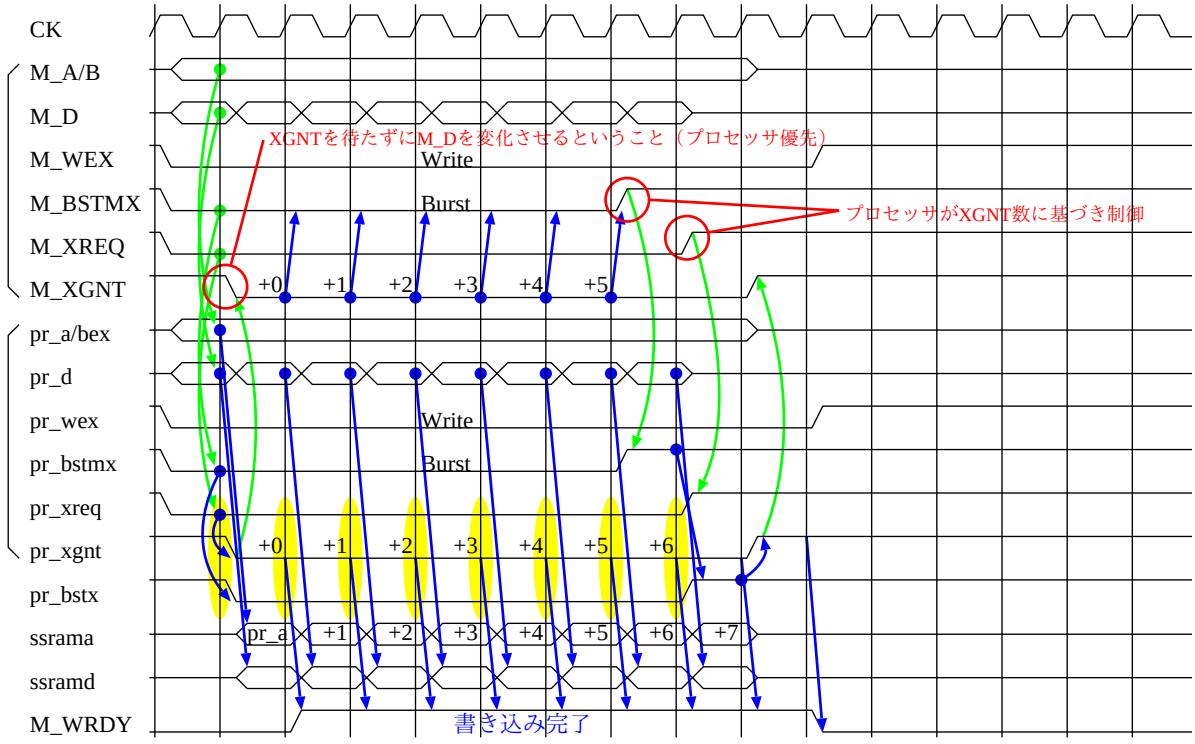

図 2.3: ZBT-SSRAM インタフェース信号のタイミングチャート (WRITE バーストモード)

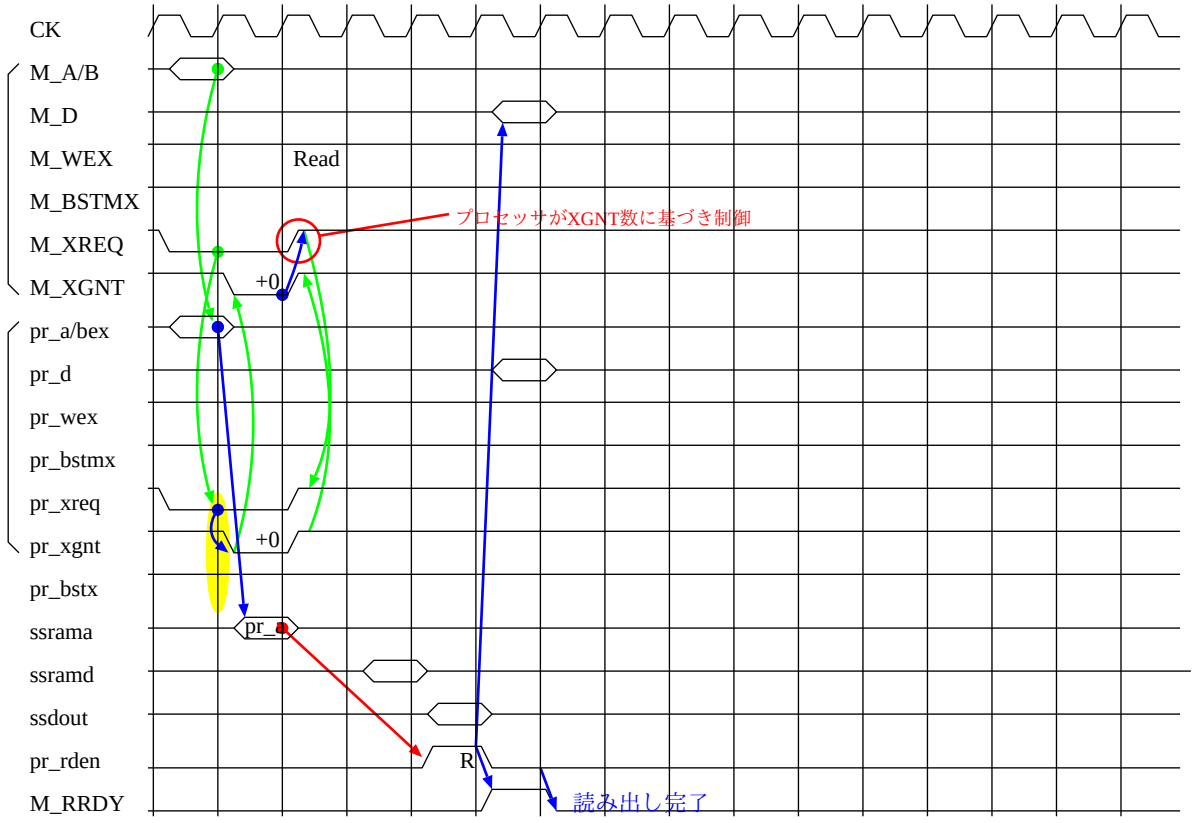

図 2.4: ZBT-SSRAM インタフェース信号のタイミングチャート (READ)

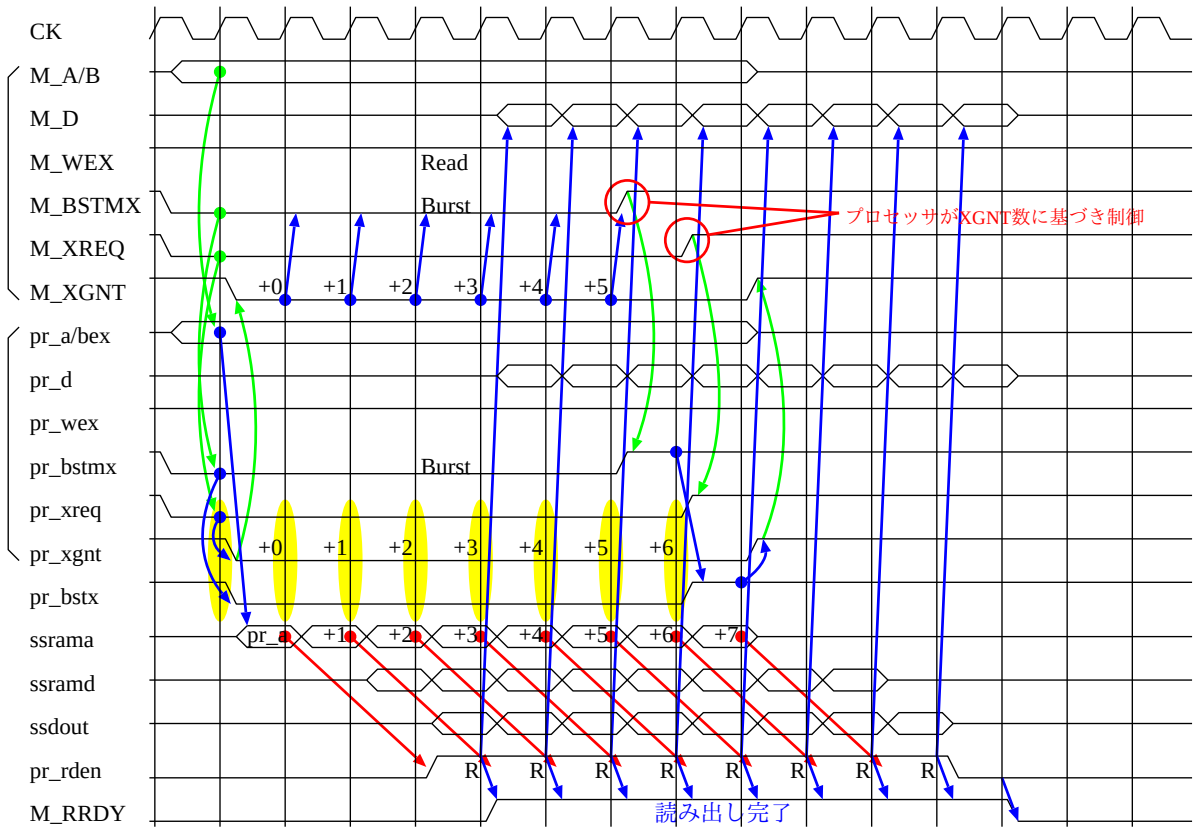

図 2.5: ZBT-SSRAM インタフェース信号のタイミングチャート (READ バーストモード)

2.3 ユーザ論理回路とホストPCのインタフェース

表 2.2 にユーザ論理回路とホスト PC のインタフェース，図 2.6 にタイミングチャートを示す．XINT は，ホスト PC が後述する BAR2 空間先頭ワードへの書き込みを完了し，ユーザ論理回路に対して動作を指示する際に ON になる．ユーザ論理回路が XINT を検出し XACK を ON にした時点で，XINT は OFF にならなければならない．XREQ はユーザ論理回路が BAR2 空間に対して書き込みまたは読み出しを行う際に ON にする信号線である．

表 2.2: ユーザ論理回路部-ホスト PC インタフェース信号（40 本）の概要

信号名	説明
C_XINT(出力)	ホスト PC からのリクエスト信号，Active-LOW
C_XACK(入力)	ユーザ論理回路からの INT 受付け信号，Active-LOW
C_XREQ(入力)	ユーザ論理回路からのレジスタ参照信号，Active-LOW
C_A(入力)	アドレス，A3-0
C_WEX(入力)	書き込みイネーブル，Active-LOW
C_D(入出力)	データ，D31-0

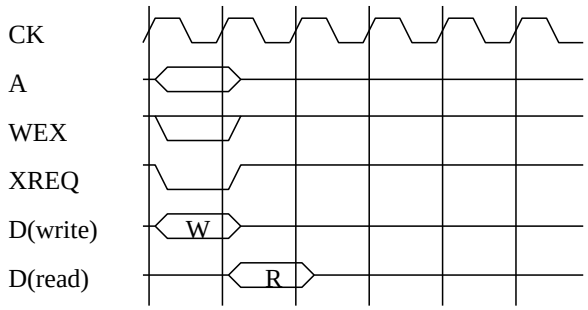

図 2.6: ホスト PC インタフェース信号のタイミングチャート (WRITE/READ)

Chapter 3

GP600M資源のホストPCへの写像

3.1 概要

GP600M は，alice110.naist.jp，alice111.naist.jp，alice112.naist.jp，alice113.naist.jp に，各々1 枚搭載されている（図 3.1）.

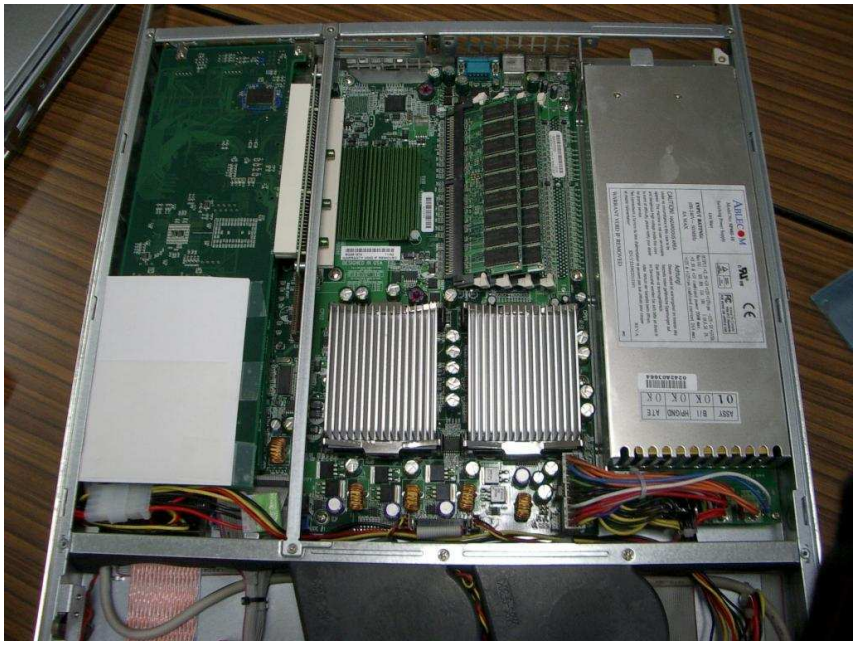

図 3.1: ホスト PC に搭載された状態の GP600M

GP600M の資源のうち，ZBT-SSRAM および BAR2 レジスタ空間が，ホスト PC の主記憶空間に写像される．なお，写像機能は，GP600M デバイスドライバの gp600m_mmap() に以下のように実装されている．

```
atop(vtophys((vm_offset_t)rman_get_virtual(bus_alloc_resource(dev, SYS_RES_MEMORY,
0x1C(PCI_CBMA3)))));
```

ユーザプログラム（esim）は，esim.h に定義されている構造体 p，および，gp600m.c に用意されている gp600m_open() を利用することにより，GP600M の資源を直接参照できる．メモリマップの概要を表 3.1 に示す.

```
/******** esim.h ********/
struct p {
  union l2ct {
    Uint ful;
    struct l2ct_0000 {
      Uint     v     : 2;      /* 0:empty 1:reserve 3:inuse */
      Uint     resv0 : 2;
      Uint     stat  : 2;      /* (input)  0:empty 2:OP-ok 3:IF-ok */
      Uint     resv1 : 1;
      Uint     flush : 1;      /* 0:normal, 1:L1-flush-only */
      Uint     t     : 6;      /* timer                     */
      Uint     type  : 4;      /* type                      */
      Uint     opcd  : 6;      /* opcd                      */
      Uint     at    : 2;      /* 0:none, 1:arm, 2:frv      */
      Uint     len   : 6;      /* length                    */
    } l2ct_0000;
  } soft_l2ct[BAR3_L2CT_REGNUM], *hard_l2ct;

  Uint *hard_l2bf;

  union bar2 {
    Uint ful;
    struct h_v {
      Uint h_c : 4;
      Uint h_p :12;
      Uint h_s :16;
    } h_v;
    struct stat {
      Uint c_stat : 4; /* bit03:00: console stat */
      Uint a_stat : 6; /* bit09-04: arm status */
      Uint f_stat : 6; /* bit15-10: frv status */
      Uint s_stat : 8; /* bit23:16: scan stat (for cycle_based control) */
      Uint p_ipctv: 4; /* bit27:24: p_swi */
      Uint p_ipctb: 4; /* bit31:28: p_swi */
    } stat;
  } soft_bar2[BAR2_REGNUM], *hard_bar2;

  Uchar *mem;
} p;

/******** gp600m.c ********/
int gp600m_open() {
  char *hard_pcimap;
  if ((fd=open(GP600M_DEVNAME,O_RDWR)) == -1) {
    printf("gp600m.c: ERROR can not open gp600m\n");
    return (-1);
  }
  if ((hard_pcimap = (char*)mmap((caddr_t)0, 0x4000000, PROT_READ|PROT_WRITE, MAP_SHARED,
fd, (off_t)0)) == MAP_FAILED) {
    printf("gp600m.c: ERROR MAP_FAILED on %s\n", GP600M_DEVNAME);
    return (-1);
  }
  p.mem       = (Uint*)       ((char*)hard_pcimap + 0x01000000);
  p.hard_bar2 = (union bar2*) ((char*)hard_pcimap + 0x02000000);
  p.hard_l2bf = (Uint*)       ((char*)hard_pcimap + 0x00000000);
  p.hard_l2ct = (union l2ct*) ((char*)hard_pcimap + 0x00000040);
  return (1);
}
```

表 3.1: メモリマップの概要

hard_pcimap からの offset	esim の変数名	説明
0x00000000-0x0000003F	p.hard_l2bf	SSRAM 空間 ユーザ論理回路が内蔵キャッシュ機能を有する場合のキャッシュラインに使用
0x00000040-0x0000007F	p.hard_l2ct	SSRAM 空間 ユーザ論理回路が内蔵キャッシュ機能を有する場合の制御に使用
0x01000000-0x01FFFFFF	p.mem	SSRAM 空間（通常空間:後半 16MB）
0x02000000-0x0200003F	p.hard_bar2	BAR2 レジスタ空間

3.2 制御信号線に影響を与えるアドレス

アドレスによっては，これまでに説明した制御信号線に影響を及ぼす．表 3.2 に列挙する．

表 3.2: 制御信号線に影響を与えるアドレス

hard_pcimap からの offset	esim の変数名	説明
0x00000078-0x0000007B	p.hard_l2ct[14]	ホスト PC は bit5-4 に 0,2,3 を書き込むことができる．値が M_STAT[1:0] に表示される pe0 は bit5-4 に 0,1 を書き込むことができる．値が M_STAT[1:0] に表示される
0x02000000-0x02000003	p.hard_bar2[0]	ホスト PC が書き込むと C_XINT が 0 になる pe0 が C_XACK を 0 にすると C_XINT は 1 に戻る
0x02000010-0x02000013	p.hard_bar2[4]	ホスト PC が非ゼロを書き込むとゼロになるまで毎サイクルデクリメント．非ゼロの間，pe0 の RST が 1 となる

3.3 モニタプログラムによる画像入出力機能

ホスト PC 上で動作するモニタプログラムには，カメラキャプチャ機能およびイメージ表示機能がある．これらを含むメモリマップを表 3.3 に示す．なお，カメラキャプチャ機能を利用するには，cam_open() を呼び出した後，cam_capt() を呼び出す．左右 2 画面が，L_BUF および R_BUF に格納される．また，イメージ表示領域を利用するには，x11_open() を呼び出した後，該当領域にデータを書き込み，x11_update() を呼び出す．

表 3.3: 画像入出力機能

hard_pcimap からの offset	esim の変数名	説明
0x00E00000-0x00E4B000	p.mem[L_BUF]	イメージ表示領域の左上 320x240 に対応する．各 4 バイトが RGB0 に対応する．cam_capt() のキャプチャ画像格納領域としても使用する
0x00E80000-0x00ECB000	p.mem[R_BUF]	イメージ表示領域の右上 320x240 に対応する．各 4 バイトが RGB0 に対応する．cam_capt() のキャプチャ画像格納領域としても使用する
0x00F00000-0x00F4B000	p.mem[W_BUF]	イメージ表示領域の左下 320x240 に対応する．各 4 バイトが RGB0 に対応する
0x00F80000-0x00FCB000	p.mem[D_BUF]	イメージ表示領域の右下 320x240 に対応する．各 4 バイトが RGB0 に対応する

3.4 サンプル

ユーザ論理回路を含む RTL

arch: nakashim/proj-pcpu/fpga-20120501-gp6m/．pe0.v にユーザ論理回路を追加すれば良い．pe0.v の該当部分は以下．最初は，レジスタ DREG を使った簡単な計算をする回路を作成し，徐々に複雑な回路にすればよい．レジスタ容量を増加させ，メモリ代わりにすれば，簡単なパイプラインプロセッサを作成でき，メモリの初期値をホスト PC から送り込み，実行結果をホスト PC にて確認することができる．合成配置配線には、top/Make-top-synplify.sh を使用する（XC2V6000 の場合，ISE101 のみ使用可）．FPGA の config は，alice110,111,112,113 のいずれかに login した状態で，Config-gp600m を使用（. /Config-gp600m top.mcs）．なお，Config および後述するモニタプログラムの実行は，実

行前にユーザ間で調整し，1 つのホスト PC 上で同時に複数の Config やモニタプログラムの実行が行われないよう留意すること．

```
/* 実行 **** ここにユーザ論理を入れる *****/
else if (state==STATE_EXEC) begin
  if (execc==0) begin
    execc <= execc + 1;
    DREG[0] <= 64'h0000000000000000;
  end
  else if (execc<EXECCOUNT) begin
    execc <= execc + 1;
    DREG[0] <= DREG[0] + 64'h0000000000000001;
  end
  else state <= STATE_WRDT;
end
```

モニタプログラム

arch: nakashim/proj-pcpu/host-20120501-gp6m/．ホスト PC 上では，esim をモニタプログラムとして使用する．alice110,111,112,113 の Config が終った状態で，./esim -x 81.bin などと実行する．現在，81.bin の内容を FPGA に送り込み，処理結果を PC に戻して画面表示する．なお，FPGA との通信箇所は以下の通りである．まず，前半では，fpga の初期化を行っている．

```
if (gp600m_open() == −1) {
  printf("ERROR GP600M is not available\n");
  onintr_exit(1);
}

printf("Loading mem_file to p.mem[%08.8x] Upper-Left Window\n", L_BUF);
if (fread(p.mem+L_BUF, 4, 320*240, memfile) ≤ 0) {
  printf("Failed to load mem_file %s\n", fname);
  exit(1);
}
printf("Filling p.mem[%08.8x] Upper-Right Window\n", R_BUF);
for (i=0; i<320*240*4; i+=4) { /* RED */
  *(Uint*)&p.mem[R_BUF+i] = 0xff000000;
}
printf("Filling p.mem[%08.8x] Lower-Left Window\n", W_BUF);
for (i=0; i<320*240*4; i+=4) { /* GREEN */
  *(Uint*)&p.mem[W_BUF+i] = 0x00ff0000;
}
printf("Filling p.mem[%08.8x] Lower-Right Window\n", D_BUF);
for (i=0; i<320*240*4; i+=4) { /* BLUE */
  *(Uint*)&p.mem[D_BUF+i] = 0x0000ff00;
}
hard_memsync();

/* GP600M reset */
printf("---- Resetting GP600M\n");
p.soft_bar2[BAR2_PE0_CONTROL].ful = 512;
hard_bar2w(BAR2_PE0_CONTROL, p.soft_bar2[BAR2_PE0_CONTROL].ful);
hard_bar2sync();
do {
  p.soft_bar2[BAR2_PE0_CONTROL] = hard_bar2r(BAR2_PE0_CONTROL);
  printf("p.BAR2_PE0_CONTROL=%08.8x\n", p.soft_bar2[BAR2_PE0_CONTROL].ful);
} while (p.soft_bar2[BAR2_PE0_CONTROL].ful);
p.soft_l2ct[BAR3_L2CT_0000].ful = 0;
hard_l2ctw(BAR3_L2CT_0000, p.soft_l2ct[BAR3_L2CT_0000]);
hard_l2ctsync();
```

後半では，ユーザ論理回路を起動して終結を待ち合わせる．

```
  /*****************************************************************************/
  /* Main Loop start */
  /*****************************************************************************/
  printf("---- Starting GP600M\n");
  p.soft_l2ct[BAR3_L2CT_0000].l2ct_0000.stat = 3;
  hard_l2ctw(BAR3_L2CT_0000, p.soft_l2ct[BAR3_L2CT_0000]);
  hard_l2ctsync();

  printf("---- Waiting GP600M\n");
  do {
    p.soft_l2ct[BAR3_L2CT_0000] = hard_l2ctr(BAR3_L2CT_0000);
    printf("p.BAR3_L2CT_0000=%08.8x\n", p.soft_l2ct[BAR3_L2CT_0000].ful);
  } while (p.soft_l2ct[BAR3_L2CT_0000].l2ct_0000.v ≠ 3 ||
p.soft_l2ct[BAR3_L2CT_0000].l2ct_0000.stat ≠ 1);

  printf("---- Reading GP600M\n");
  for (i=0; i<0x20; i+=4) {
    if (i % 32 == 0) printf("%08.8x:", i);
    printf(" %08.8x", *(Uint*)&p.mem[L_BUF+i]);
    if (i % 32 == 28) printf("\n");
  }
  for (i=0; i<0x20; i+=4) {
    if (i % 32 == 0) printf("%08.8x:", i);
    printf(" %08.8x", *(Uint*)&p.mem[W_BUF+i]);
    if (i % 32 == 28) printf("\n");
  }
  /*****************************************************************************/
  /* Main Loop end */
  /*****************************************************************************/
```