XAPP130 (v1.4) 2000 年 12 月 18 日

Virtex Block SelectRAM+ 機能の使い方

概要

Virtex™ シリーズには、4096 個のメモリ セルを持つ読み書き可能なデュアル ポート同期 RAM の専用ブロックがオンチップにあります。Block SelectRAM+™ メモリの各ポートは、それぞれ読み出し/書き込みポート、読み出しポート、書き込みポートのいずれかにコンフィギュレーションできます。また、データ幅も指定できます。Block SelectRAM+ が提供する新機能によって、デザインを簡素化できます。

はじめに

Virtex シリーズのデバイスは、Block SelectRAM+ 機能が追加されています。XC4000X ファミリで使用できる分散 RAM も使用できます。分散 RAM を使用すると、各 CLB に RAM を配置してデバイス全体に浅い RAM 構造を作成できます。Block SelectRAM+ メモリを使用すると、高速、複数、かつ大規模な RAM のブロックを利用できるようになります。各 Block SelectRAM+ セルはデバイスの列に配置されており、4 つの CLB に広がっています。

Block SelectRAM+ は、4096 個のメモリ セルを持つ完全に同期した読み書き可能なデュアル ポート RAM です。各ポートの制御信号は独立しているので、柔軟な構造の RAM を作成できます。各ポートのデータ幅は独立してコンフィギュレーションできるので、バス幅変換を組み込むことができます。

表 1 に、Block SelectRAM+ メモリの深さと幅の関係を示します。

表 1: Block SelectRAM+ ポートの深さと幅の関係

幅	深さ	アドレス バス	データ バス
1	4096	ADDR<11:0>	DATA<0>
2	2048	ADDR<10:0>	DATA<1:0>
4	1024	ADDR<9:0>	DATA<3:0>
8	512	ADDR<8:0>	DATA<7:0>
16	256	ADDR<7:0>	DATA<15:0>

同期メモリの基本事項

分散コンポーネントとしてのデュアル ポート同期 RAM とシングル ポート同期 RAM には、読み出しと書き込みについてさまざまな動作モードがあります。主要な動作モードは、次の 4 つです。

リード スルー (1 クロック エッジ)

読み出しアドレスは読み出しポートのクロック エッジでレジスタに格納され、RAM のアクセス時間後にデータが出力されます。セットアップ タイムに対して clock-to-out を高速にするため、ラッチやレジスタをメモリの出力に配置する場合があります。しかし、このようにすると読み出しが非同期になるので、一般にあまり良い方法とは考えられていません。また、読み出しパルス クロックを生成するときにアドレス/制御ラインの遷移を検出できなくなる可能性が増加します。

パイプライン化読み出し (2 クロック エッジ)

読み出しアドレスは、読み出しポートのクロック エッジでレジスタに格納されます。データは、2 番目の読み出しクロック エッジの後でレジスタに格納されて、出力上に現れます。

ライト バック (1 クロック エッジ)

書き込みアドレスは、書き込みポートのクロック エッジでレジスタに格納されます。データ入力はメモリに書き込まれ、書き込みポート入力に同じ内容が出力されます。

ライト スルー (1 クロック エッジ)

書き込みアドレスは、書き込みポートのクロック エッジでレジスタに格納されます。データは書き込みポート入力に同じ内容が出力され、書き込みアドレスと読み出しアドレスが一致する場合は読み出しポート出力にも出力されます。メモリに対するデータの書き込みと読み出しは、同じサイクルで行われます。

Block SelectRAM+ メモリには、リード スルー機能とライト バック機能があります。出力に CLB レジスタを追加するだけで、パイプライン化読み出し機能が必要なデザインを実現できます。このようにすると、デバイスのエリアをほとんど消費せずに Block SelectRAM+ メモリの clock-to-out のタイミングを改善できます。

Block SelectRAM+ の特徴

1. すべての入力はポート クロックでレジスタに格納され、クロックに対するセットアップのタイミングが規定されています。
2. すべての出力は、リード スルー機能またはライト バック機能になります。これは、ポートの WE ピンの状態によって異なります。ポート クロックに関係する出力は、規定されている clock-to-out のタイミングの後に有効になります。
3. ブロック RAM は純粋な SRAM メモリであり、アドレスから出力への組み合わせパスはありません。CLB の LUT SelectRAM+ セルでは、組み合わせパスを使用して同等の機能を実現します。
4. 各ポートは、クロック、制御、アドレス、読み出し / 書き込み機能、データ幅などが完全に独立しています。
5. 書き込みで必要なのは 1 クロック エッジだけです。
6. 読み出しで必要なのは 1 クロック エッジだけです。
7. グリッチがない読み出しを保証するため、独自にタイミングを与える回路によって出力ポートがラッチされます。出力ポートの状態は、そのポートで新たに読み出しまたは書き込みを行うまで変化しません。

ライブラリ プリミティブ

図 1 と図 2 に、2 種類のジェネリックな Block SelectRAM+ のライブラリ プリミティブを示します。表 2 に、合成とシミュレーションに使用できるすべてのプリミティブを示します。

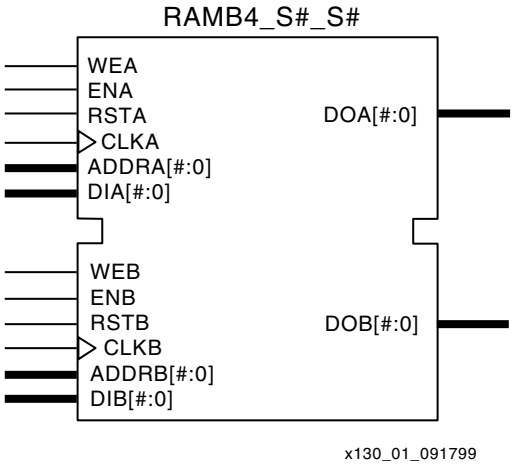

図 1: デュアル ポート Block SelectRAM+ メモリ

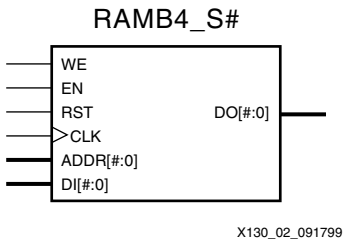

図 2: シングル ポート Block SelectRAM+ メモリ

表 2: 使用できるライブラリ プリミティブ

プリミティブ	ポート A の幅	ポート B の幅
RAMB4_S1	1	**該当なし**
RAMB4_S1_S1		1
RAMB4_S1_S2		2
RAMB4_S1_S4		4
RAMB4_S1_S8		8
RAMB4_S1_S16		16
RAMB4_S2	2	**該当なし**
RAMB4_S2_S2		2
RAMB4_S2_S4		4
RAMB4_S2_S8		8
RAMB4_S2_S16		16
RAMB4_S4	4	**該当なし**
RAMB4_S4_S4		4
RAMB4_S4_S8		8
RAMB4_S4_S16		16
RAMB4_S8	8	**該当なし**
RAMB4_S8_S8		8
RAMB4_S8_S16		16
RAMB4_S16	16	**該当なし**
RAMB4_S16_S16		16

ポートの信号

Block SelectRAM+ の各ポートは同じ 4096 個のメモリ セルにアクセスしますが、動作は独立しています。

クロック - CLK[A|B]

各ポートは、独立したクロック ピンによって完全に同期しています。すべてのポート入力ピンには、CLK ピンに対して参照されるセットアップ タイムがあります。データ出力バスには、CLK に対して参照される clock-to-out タイムがあります。

イネーブル - EN[A|B]

イネーブル ピンは、ポートの読み出し、書き込み、リセットの各機能に影響を与えます。イネーブル ピンがアクティブでないポートでは出力ピンの状態が変化せず、メモリ セルにデータは書き込まれません。

ライト イネーブル - WE[A|B]

ライト イネーブル ピンをアクティブにすると、ポートがメモリ セルにデータを書き込めるようになります。ライト イネーブルがアクティブな場合、データ入力バスの内容はアドレス バスによって指定されるアドレスの RAM に書き込まれ、新しいデータがデータ出力バスに出力されます。アクティブでない場合は読み出し動作が行われ、アドレス バスによって指定されるメモリ セルの内容がデータ出力バスに出力されます。

リセット - RST[A|B]

リセット ピンは、データ出力バスのラッチをクロック同期で強制的に 0 にします。これは RAM のメモリ セルに影響を与えず、他のポートにおける書き込みも妨げません。

アドレス バス - ADDR[A|B]<#:0>

アドレス バスは、読み出しまたは書き込みを行うメモリ セルを選択します。このバスに必要な幅は、ポートの幅によって決定されます。表 1を参照してください。

データ入力バス - DI[A|B]<#:0>

データ入力バスは、RAM に書き込む新しいデータ値を提供します。このバスとポートは同じ幅になります。表 1 を参照してください。

データ出力バス - DO[A|B]<#:0>

データ出力バスは、直前のアクティブなクロック エッジでアドレス バスが参照していたメモリ セルの内容を反映します。書き込み中は、データ入力バスを反映します。このバスの幅は、ポートの幅と同じです。可能な幅については、表 1 を参照してください。

制御ピンの反転

各ポートの 4 本の制御ピン (CLK、EN、WE、RST) は、コンフィギュレーション オプションとして別々に反転制御できます。

アドレス マップ

各ポートは、ポートの幅によって決定されるアドレス指定方法に応じて、同じ 4096 個のメモリ セルにアクセスします。特定の幅について物理的な RAM のロケーションを指定する式は、次のようになります。これが必要になるのは、2 つのポートで幅と深さの比率が異なる場合だけです。

$$Start = ([ADDR_{port} + 1] \times Width_{port}) - 1$$

$$End = ADDR_{port} \times Width_{port}$$

表 3 に、各ポート幅について低位アドレスのマップを示します。

表 3: ポートのアドレス マップ

ポート幅	ポート アドレス																
1	4095...	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
2	2047...	07		06		05		04		03		02		01		00	
4	1023...	03				02				01				00			
8	511...	01								00							
16	255...	00															

より大きい RAM 構造の作成

Block SelectRAM+ の列には専用の配線リソースがあるので、最小の配線遅延でブロックをカスケード接続できます。これによって、通常の配線チャネルを使用する場合よりも少ないタイミング ペナルティで深さや幅が大きい RAM 構造を実現できます。

ロケーション制約

Block SelectRAM+ のインスタンスに LOC プロパティを指定すると、配置を制約できます。Block SelectRAM+ の配置は CLB ロケーションの命名規則から独立しているので、アレイ間で同じような LOC プロパティを簡単に指定できます。

LOC プロパティでは、次のような形式を使用します。

LOC = RAMB4_R#C#

RAMB4_R0C0 は、デバイスの左上にある RAMB4 です。

競合の解決

Block SelectRAM+ メモリは真のデュアル ポート RAM なので、両方のポートから同じメモリ セルに同時にアクセスできます。あるメモリ セルに一方のポートが書き込んでいる場合、clock-to-clock のセットアップ期間中は読み出しか書き込みかに関係なくもう一方のポートがそのメモリ セルにアクセスしてはいけません。メモリ セルの書き込みが競合すると、次のようになります。

- 両方のポートが同じメモリ セルに同時に書き込もうとして clock-to-clock セットアップの必要条件に違反した場合、無効なデータが格納されます。
- 一方のポートが書き込もうとしているメモリ セルをもう一方のポートが同時に読み出そうとして clock-to-clock セットアップの必要条件に違反した場合、次のようになります。
  - 書き込みが成功します。
  - 書き込みポートに出力されるデータは、書き込まれたデータを正確に反映します。
  - 読み出しポートには無効なデータが出力されます。
- 競合が発生しても、物理的に破損することはありません。

読み出しと書き込み

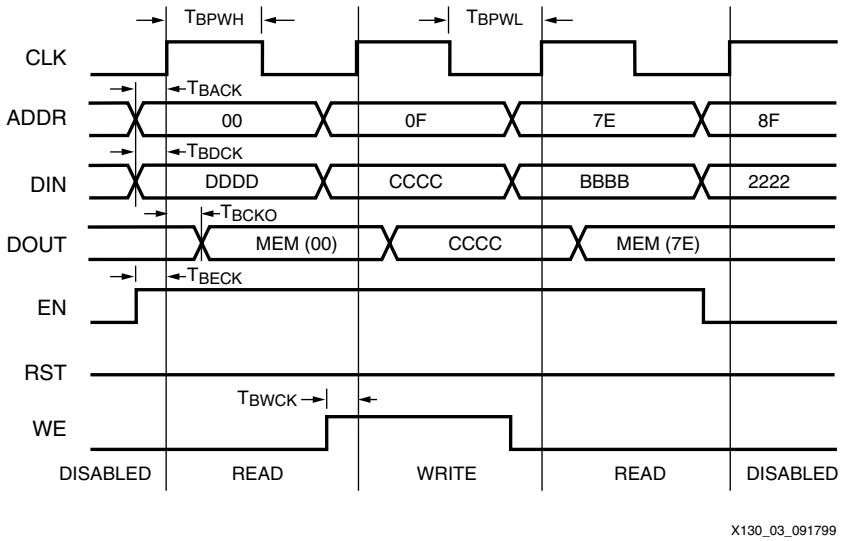

図 3: シングル ポート Block SelectRAM+ メモリのタイミング図

シングル ポートのタイミング

図 3 に、シングル ポートの Block SelectRAM+ メモリのタイミング図を示します。ブロック RAM の AC スイッチング特性については、データシートを参照してください。ブロック RAM メモリは、初期状態では無効になっています。

次の手順と図 3 を参照してください。

1. CLK ピンの 1 番目の立ち上がりエッジで、ADDR、DI、EN、WE、RST の各ピンがサンプリングされます。EN ピンが「High」、WE ピンが「Low」なので、読み出しが行われます。DO バスには、ADDR バスによって指定される 0x00 というアドレスのメモリの内容が出力されます。
2. CLK ピンの 2 番目の立ち上がりエッジで、ADDR、DI、EN、WR、RST の各ピンが再びサンプリングされます。EN ピンと WE ピンが「High」なので、書き込みが行われます。DO バスには、DI バスのデータが出力されます。DI バスのデータが、0x0F というアドレスのメモリに書き込まれます。
3. CLK ピンの 3 番目の立ち上がりエッジで、ADDR、DI、EN、WR、RST の各ピンが再びサンプリングされます。EN ピンが「High」、WE ピンが「Low」なので、読み出しが行われます。DO バスには、ADDR バスによって指定される 0x0F というアドレスのメモリの内容が出力されます。
4. CLK ピンの 4 番目の立ち上がりエッジで、ADDR、DI、EN、WR、RST の各ピンが再びサンプリングされます。EN ピンが「Low」なので、ブロック RAM は無効になります。DO バスには、最後の値が出力されたままになります。

デュアル ポートのタイミング

図 4 に、デュアル ポート ブロック RAM のタイミング図を示します。ポート A のクロックは、ポート B のクロックより周期が長くなっています。この図には、タイミング パラメータの T_{BCCS} (clock-to-clock セットアップ) が示されています。そして、T_{BCCS} の違反が 1 回発生しています。他のタイミング パラメータは、すべて図 3 に示したシングル ポートのものと同じです。

T_{BCCS} が意味を持つのは、両方のポートのアドレスが同じときに少なくとも一方のポートが書き込みを行っている場合だけです。両方のポートで書き込みを行っている状態で clock-to-clock セットアップ パラメータに違反すると、そのロケーションにあるメモリの内容が無効になります。一方が書き込み、もう一方が読み出しを行っている状態で clock-to-clock セットアップ パラメータに違反すると、メモリには正しいデータが書き込まれますが、読み出しポートには無効なデータが出力されます。

次の手順だけでなく、7 ページの図 4 も参照してください。

1. CLKA の 1 番目の立ち上がりエッジで、メモリ ロケーション 0x00 に 0xAAAA という値が書き込まれ、この値が DOA バスに出力されます。ポート B で直前に行われた操作は、同じメモリ ロケーション 0x00 に対する読み出しでした。ポート B の DOB バスはポート A の新しい値に変化せず、それまでの値を保ちます。少し時間がたってからポート B でメモリ ロケーション 0x00 を再び読み出すと、ポート A で書き込んだ新しい値が DOB バスに出力されます。
2. CLKA の 2 番目の立ち上がりエッジで、メモリ ロケーション 0x7E に 0x9999 という値が書き込まれ、この値が DOA バスに出力されます。そして、ポート B で T_{BCCS} パラメータに違反せずに同じメモリ ロケーションを読み出すと、ポート A で書き込んだ新しい値が DOB バスに出力されます。
3. CLKA の 3 番目の立ち上がりエッジでは、メモリ ロケーション 0x0F に両方のポートで書き込もうとしており、TBCCS パラメータに違反しています。DOA バスと DOB バスにはそれぞれ DIA バスおよび DIB バスの内容が出力されますが、0x0F には無効な値が格納されます。
4. CLKA の 4 番目の立ち上がりエッジではメモリ ロケーション 0x0F を読み出していますが、無効なデータが DOA バスに出力されます。ポート B もメモリ ロケーション 0x0F を読み出していますが、やはり無効なデータが読み出されます。
5. CLKA の 5 番目の立ち上がりエッジでは、前回のポート B による 0x7E への書き込みに対して T_{BCCS} パラメータに違反しない読み出しが行われています。DOA バスには、ポート B で最後に書き込んだ値が出力されます。

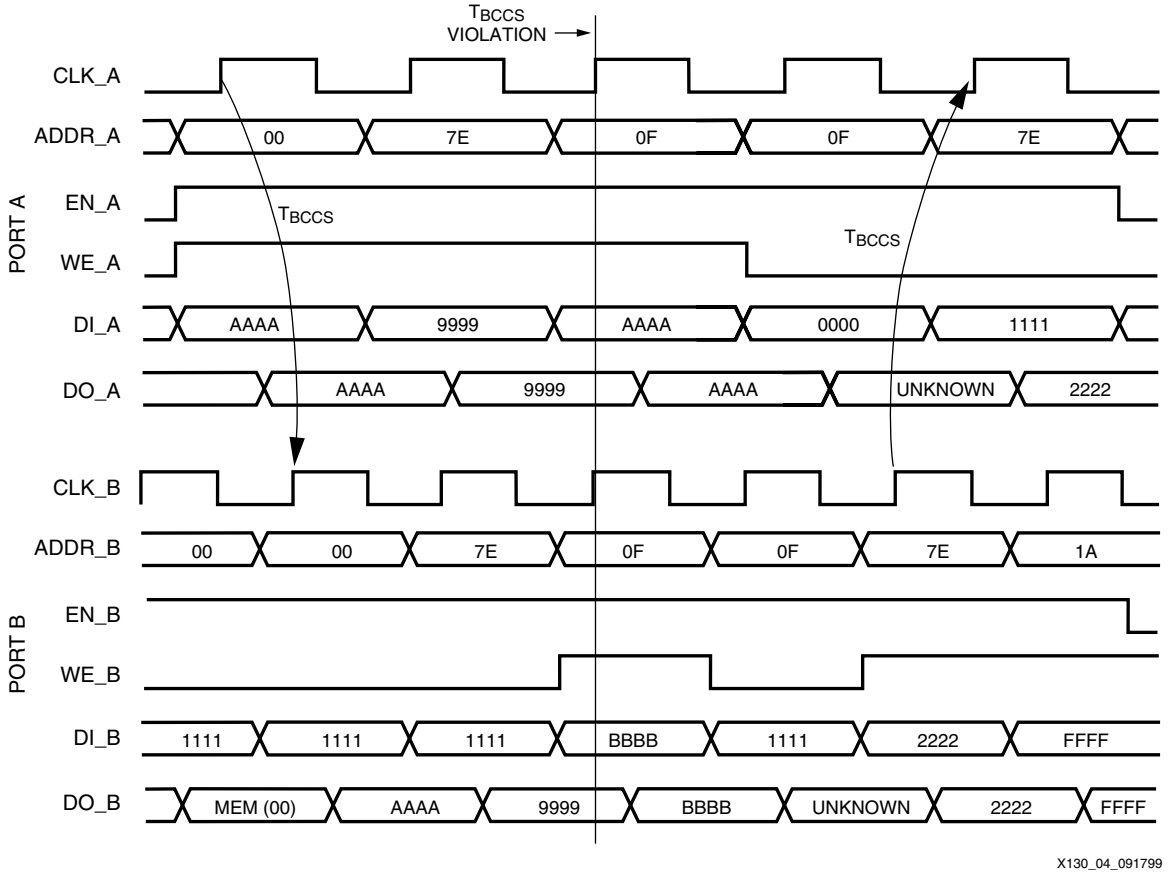

図 4: デュアル ポート Block SelectRAM+ メモリのタイミング図

デザイン例

32 ビット シングル ポート RAM の作成

ブロック RAM のデュアル ポート機能を利用すると、単一のブロック RAM セルを使用して図 5 に示すような深さ 128 ビット X 幅 32 ビットのシングル ポート RAM を作成できます。

メモリ スペースをインターリーブし、ポート A のアドレス バスの LSB を 1 (V_{CC})、ポート B のアドレス バスの LSB を 0 (GND) に設定すると、32 ビット幅のシングル ポート RAM を作成できます。

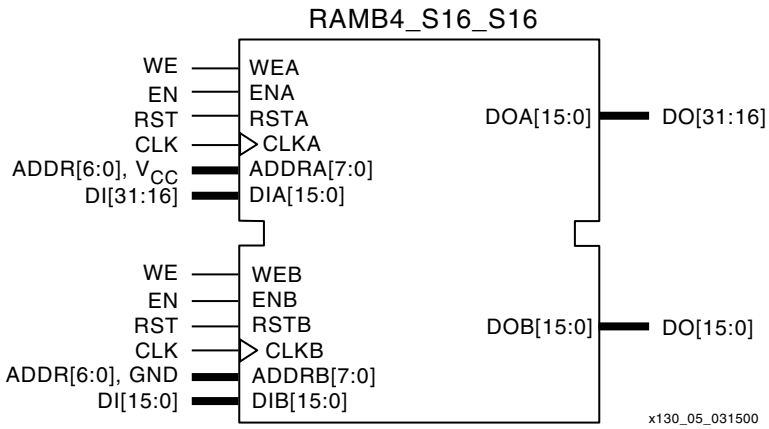

図 5: シングル ポートの 128 X 32 RAM

2 つのシングル ポート RAM の作成

ブロック RAM のデュアル ポート機能を利用すると、図 6 に示すように 1 つの RAM を各 2048 ビットの 2 つのシングル ポート メモリに分割できます。

この例では、1 つのブロック RAM から 512 X 4 の RAM (ポート A) と 128 X 16 の RAM (ポート B) を作成しています。RAM のアドレス スペースは、上位 2048 ビットについてはポート A の MSB を 1 (V_{CC})、下位 2048 ビットについてはポート B の MSB を 0 (GND) に固定することによって分離しています。

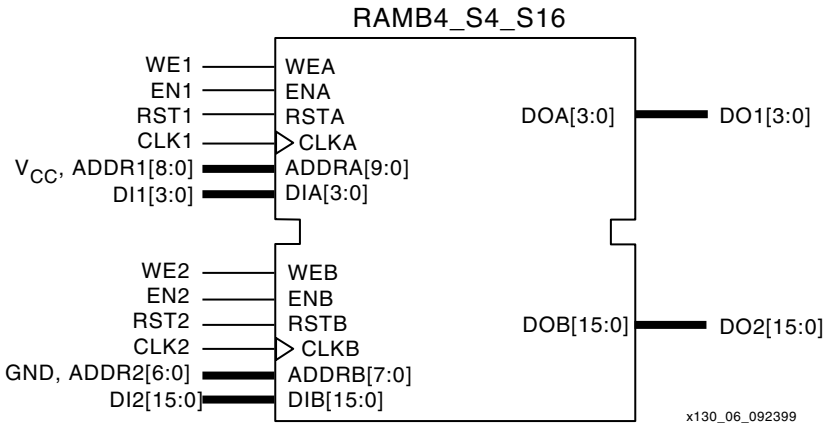

図 6: 512 X 4 の RAM と 128 X 16 の RAM

ブロック メモリの生成

CORE Generator プログラムは、Block SelectRAM+ 機能を使用してメモリ構造を生成します。このプログラムは、デザインに含めるための EDIF ファイルと VHDL または Verilog のシミュレーション コード テンプレートを出力します。

初期化

Block SelectRAM+ メモリは、デバイスのコンフィギュレーション シーケンス中に初期化できます。各 RAM を初期化するには、それぞれ 64 個の 16 進値から構成される 16 個の初期化プロパティ (合計で 4096 ビット) を使用します。これらのプロパティを、表 4 に示します。初期化プロパティの値を明示的に設定しない場合、0 が設定されます。部分的に初期化すると、残りの部分は 0 になります。64 個を越える 16 進値を初期化に使用すると、エラーが発生します。

初期値を使用して RAM をシミュレーションするには、VHDL シミュレータの場合はジェネリック文、Verilog シミュレータの場合はパラメータを使用します。

VHDL および Synopsys における初期化

シミュレーションと合成の両方について、VHDL で Block SelectRAM+ 構造を初期化して、EDIF 出力ファイルに含めることができます。VHDL コードのシミュレーションでは、ジェネリック文を使用して初期値を設定します。現在、Synopsys 社 の FPGA コンパイラではジェネリック文がサポートされていません。このため、初期値は Synopsys に組み込まれている dc_script を使用して RAM の属性として設定します。ジェネリック文が合成変換されないようにするには、translate_off 文を使用します。次に示すコードは、これらの手法を使用したモジュールです。

表 4: RAM の初期化プロパティ

プロパティ	メモリ セル
INIT_00	255 ～ 0
INIT_01	511 ～ 256
INIT_02	767 ～ 512

表 4: RAM の初期化プロパティ (続き)

プロパティ	メモリ セル
INIT_03	1023 ～ 768
INIT_04	1279 ～ 1024
INIT_05	1535 ～ 1280
INIT_06	1791 ～ 2047
INIT_07	2047 ～ 1792
INIT_08	2303 ～ 2048
INIT_09	2559 ～ 2304
INIT_0a	2815 ～ 2560
INIT_0b	3071 ～ 2816
INIT_0c	3327 ～ 3072
INIT_0d	3583 ～ 3328
INIT_0e	3839 ～ 3584
INIT_0f	4095 ～ 3840

VHDL による初期化の例

```
library IEEE;
use IEEE.std_logic_1164.all;

entity MYMEM is
port (CLK, WE:in std_logic;
ADDR: in std_logic_vector(8 downto 0);
DIN: in std_logic_vector(7 downto 0);
DOUT: out std_logic_vector(7 downto 0));
end MYMEM;

architecture BEHAVE of MYMEM is
signal logic0, logic1: std_logic;

component RAMB4_S8
--synopsys translate_off
generic( INIT_00,INIT_01, INIT_02, INIT_03, INIT_04, INIT_05, INIT_06,
INIT_07,
INIT_08, INIT_09, INIT_0a, INIT_0b, INIT_0c, INIT_0d, INIT_0e, INIT_0f :
BIT_VECTOR(255 downto 0)
:= X"0000000000000000000000000000000000000000000000000000000000000000");
--synopsys translate_on
port (WE, EN, RST, CLK: in STD_LOGIC;
ADDR: in STD_LOGIC_VECTOR(8 downto 0);
DI: in STD_LOGIC_VECTOR(7 downto 0);
DO: out STD_LOGIC_VECTOR(7 downto 0));
end component;

--synopsys dc_script_begin
```

```
--set_attribute ram0 INIT_00
"0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF" -type
string
--set_attribute ram0 INIT_01
"FEDCBA9876543210FEDCBA9876543210FEDCBA9876543210FEDCBA9876543210" -type
string
--synopsys dc_script_end

begin
logic0 <='0';
logic1 <='1';

ram0: RAMB4_S8
--synopsys translate_off
generic map (
INIT_00 =>
X"0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF",
INIT_01 =>
X"FEDCBA9876543210FEDCBA9876543210FEDCBA9876543210FEDCBA9876543210")
--synopsys translate_on
port map (WE=>WE, EN=>logic1, RST=>logic0, CLK=>CLK,ADDR=>ADDR, DI=>DIN,
DO=>DOUT);

end BEHAVE;
```

Verilog および Synopsys における初期化

シミュレーションと合成の両方について、Verilog で Block SelectRAM+ 構造を初期化して、EDIF 出力ファイルに含めることができます。Verilog コードのシミュレーションでは、defparam 文を使用して初期値を設定します。現在、Synopsys 社 の FPGA コンパイラでは defparam 文がサポートされていません。このため、初期値は Synopsys に組み込まれている dc_script を使用して RAM の属性として設定します。defparam 文が合成変換されないようにするには、translate_off 文を使用します。次に示すコードは、これらの手法を使用したモジュールです。

Verilog による初期化の例

```
module MYMEM (CLK, WE, ADDR, DIN, DOUT);
input CLK, WE;
input [8:0] ADDR;
input [7:0] DIN;
output [7:0] DOUT;

wire logic0, logic1;

//synopsys dc_script_begin
//set_attribute ram0 INIT_00
"0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF" -type
string
//set_attribute ram0 INIT_01
"FEDCBA9876543210FEDCBA9876543210FEDCBA9876543210FEDCBA9876543210" -type
string
//synopsys dc_script_end
```

```
assign logic0 = 1'b0;
assign logic1 = 1'b1;

RAMB4_S8 ram0 (.WE(WE), .EN(logic1), .RST(logic0), .CLK(CLK), .ADDR(ADDR),
.DI(DIN), .DO(DOUT));
//synopsys translate_off
defparam ram0.INIT_00 =
256h'0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF;
defparam ram0.INIT_01 =
256h'FEDCBA9876543210FEDCBA9876543210FEDCBA9876543210FEDCBA9876543210;
//synopsys translate_on
endmodule
```

改訂履歴

次の表に、このドキュメントの改訂履歴を示します。

日付	バージョン	改訂内容
10/16/98	1.0	初期リリース
09/24/99	1.1	ドキュメント全体で情報の追加とアップデート
12/29/99	1.2	ドキュメントの再フォーマット
3/16/00	1.3	図 5 の修正
11/20/00	1.4	ドキュメントの再フォーマットと 6 ページ目のメモリ ロケーションの訂正