

## ブロック RAM (デュアルポート)

同期読み見出し同期書き込みのデュアルポートメモリ  
論理合成結果, ブロック RAM として実装される.

### 読み出し優先 (read-first)

書き込み前の値が読み出される.

```
module dpram(clk, load1, load2, addr1, addr2, d1, d2, q1, q2);
  parameter DWIDTH=16,AWIDTH=12,WORDS=4096;

  input clk,load1,load2;
  input [AWIDTH-1:0] addr1,addr2;
  input [DWIDTH-1:0] d1,d2;
  output [DWIDTH-1:0] q1,q2;
  reg [DWIDTH-1:0] q1,q2;
  reg [DWIDTH-1:0] mem [WORDS-1:0];

  always @(posedge clk)
  begin
    if(load1) mem[addr1] <= d1;
    q1 <= mem[addr1];
  end

  always @(posedge clk)
  begin
    if(load2) mem[addr2] <= d2;
    q2 <= mem[addr2];
  end

endmodule
```

### 書き込み優先 (write-first)

書き込み後の値が読み出される.

```
module dpram(clk, load1, load2, addr1, addr2, d1, d2, q1, q2);
  parameter DWIDTH=16,AWIDTH=12,WORDS=4096;

  input clk,load1,load2;
  input [AWIDTH-1:0] addr1,addr2;
  input [DWIDTH-1:0] d1,d2;
  output [DWIDTH-1:0] q1,q2;
  reg [AWIDTH-1:0] ad1,ad2;
  reg [DWIDTH-1:0] mem [WORDS-1:0];

  always @(posedge clk)
  begin
    if(load1) mem[addr1] <= d1;
    ad1 <= addr1;
  end

  always @(posedge clk)
  begin
    if(load2) mem[addr2] <= d2;
    ad2 <= addr2;
  end

  assign q1 = mem[ad1];
  assign q2 = mem[ad2];

endmodule
```

#### 注 1

記述をわずかに変更しても、論理合成ツール(XST)がブロック RAMとして実装できると判定しないことがあるので、注意が必要。たとえば、2つの always 文を 1 つの always 文にまとめることができるが、XST は分散 RAMとして実装してしまう。

#### 注 2

メモリの初期化は `ram.v` (シングルポートメモリ) と同様に行うことができる。ただし、`write-first` の場合は正しく論理合成できない。(XSTのバグ?)

注3

`read-first` で記述しても、論理合成結果は `write-first` になってしまうことがある。(XSTのバグ?)