ブロック RAM を用いたスタック

パラメータ

パラメータ名	既定値	
AWIDTH	10	ブロック RAM のアドレス幅
N	1024	スタック（ブロック RAM）の要素数

入出力ポート

	ポート名	既定値	
入力	clk	1	グローバルクロック
入力	reset	1	グローバルリセット
入力	load	1	1 のとき clk の立ち上がりでスタックトップに d の値を書き込み
入力	push	1	1 のとき clk の立ち上がりでスタックをプッシュ
入力	pop	1	1 のとき clk の立ち上がりでスタックをポップ
入力	pop2	1	1 のとき clk の立ち上がりでスタックを 2 つポップ（ポップ 2 回分）
入力	thru	1	1 のとき thru を qtop に出力
入力	d	16	スタックトップに書き込む値
入力	dthru	16	thru が 1 のとき dthru を qtop に出力
出力	qtop	16	thru が 1 のとき dthru，0 のときスタックトップの値
出力	qnext	16	スタックの 2 番目の値

PUSH 命令 (メモリから読み出したデータをスタックにプッシュ) を 1 クロックサイクルで実行するために，thru と dthru を導入．
メモリから読み出すのに 1 サイクル必要で，その読み出したデータをスタックトップに書き込むのにさらに１サイクル必要となる．そこでスタックに書き込むかわりに，メモリからの読み出しデータを dthru に入力し，それをスタックトップ qtop の値とする．つまり，フリップフロップ q[0] に書かれているスタックトップの値をスルーして，メモリの出力をスタックトップ qtop として

出力する.

ソースコード

```
module stackm(clk, reset, load, push, pop, pop2, thru, d, dthru, qtop, qnext);
parameter AWIDTH = 10, N = 1024;

  input clk, reset, load, push, pop, pop2, thru;
  input [15:0] d, dthru;
  output [15:0] qtop, qnext;
  reg [AWIDTH-1:0] ptop, ptopf, addra, addrb;
  reg [15:0] mem [N-1:0];
  reg [15:0] qtopf;
  wire [15:0] pnextf;
  wire thruwrite;

  always @(push or pop or pop2 or ptop)
  if(push) ptopf = ptop - 1;
  else if(pop) ptopf = ptop + 1;
  else if(pop2) ptopf = ptop + 2;
  else ptopf = ptop;

  always @(posedge clk or negedge reset)
  if(!reset) ptop <= 0;
  else ptop <= ptopf;

  assign  pnextf = ptopf + 1;
  assign thruwrite = !(push||pop) && thru;

  always @(load or thruwrite or d or dthru)
  if(load) qtopf = d;
  else if(thruwrite) qtopf = dthru;
  else qtopf = 16'hxxxx;

  always @(posedge clk)
  begin
    if(load || thruwrite) mem[ptopf] <= qtopf;
    addra <= ptopf;
  end

  always @(posedge clk)
  begin
    if(push && thru) mem[pnextf] <= dthru;
    addrb <= pnextf;
  end

  assign qtop = (thru ? dthru : mem[addra]);
  assign qnext = mem[addrb];

endmodule
```