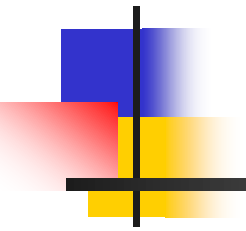
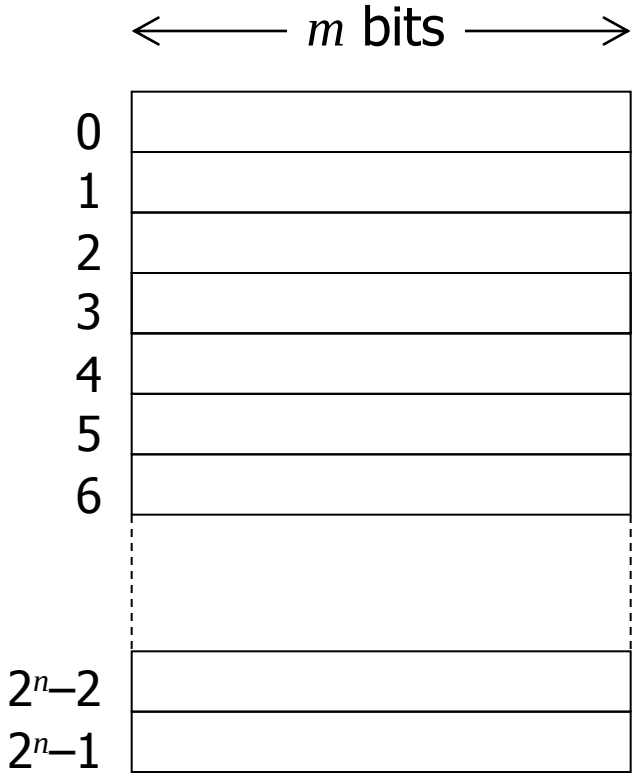


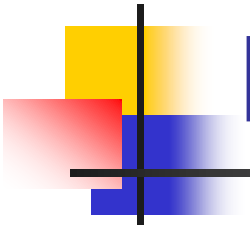
Digital Design: An Embedded Systems Approach Using Verilog

Chapter 5
Memories

General Concepts

- A memory is an array of storage locations
  - Each with a unique address
  - Like a collection of registers, but with optimized implementation
- Address is unsigned-binary encoded
  - n address bits $\Rightarrow 2^n$ locations
- All locations the same size
  - $2^n \times m$ bit memory

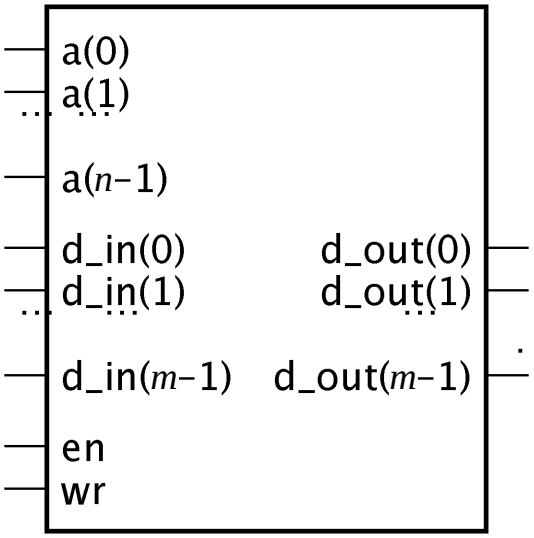




Memory Sizes

- Use power-of-2 multipliers
  - Kilo (K): $2^{10} = 1{,}024 \approx 10^3$
  - Mega (M): $2^{20} = 1{,}048{,}576 \approx 10^6$
  - Giga (G): $2^{30} = 1{,}073{,}741{,}824 \approx 10^9$
- Example
  - 32K × 32-bit memory
  - Capacity = 1,024K = 1Mbit
  - Requires 15 address bits
- Size is determined by application requirements

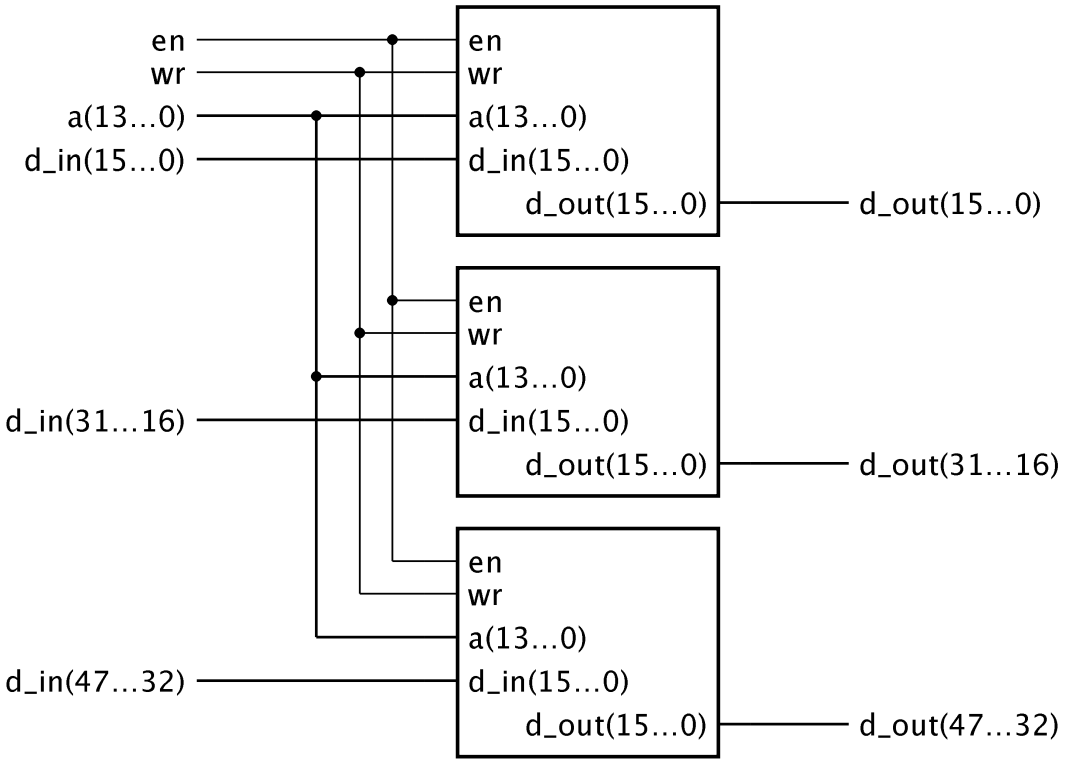
Basic Memory Operations



- a inputs: unsigned address
- d_in and d_out
  - Type depends on application
- Write operation
  - en = 1, wr = 1
  - d_in value stored in location given by address inputs
- Read operation
  - en = 1, wr = 0
  - d_out driven with value of location given by address inputs
- Idle: en = 0

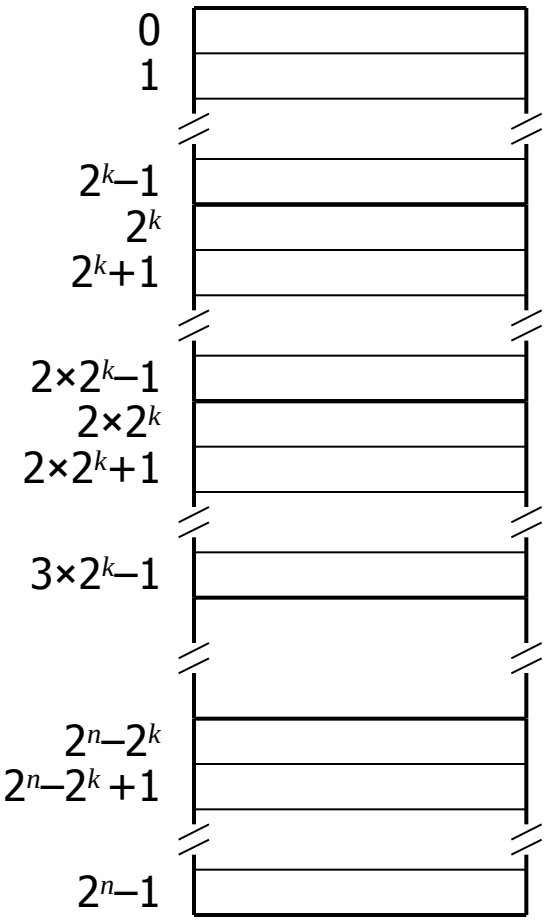
Wider Memories

- Memory components have a fixed width
  - E.g., ×1, ×4, ×8, ×16, ...
- Use memory components in parallel to make a wider memory
  - E.g, three 16K×16 components for a 16K×48 memory



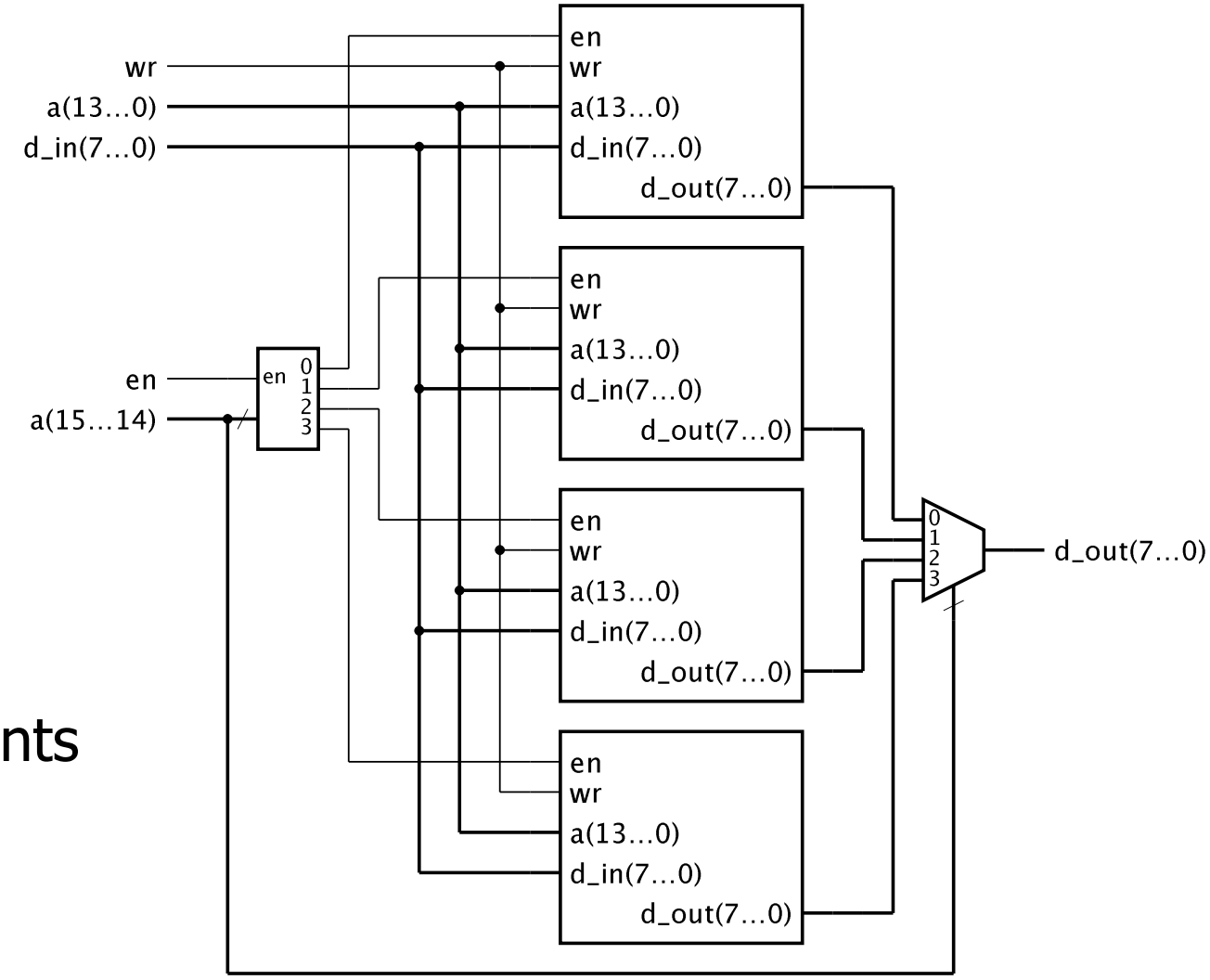
More Locations

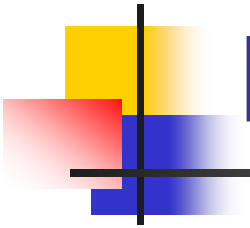
- To provide 2^n locations with 2^k-location components
  - Use $2^n/2^k$ components
- Address A
  - at offset A mod 2^k
    - least-significant k bits of A
  - in component $\lfloor A/2^k \rfloor$
    - most-significant $n-k$ bits of A
    - decode to select component



More Locations

- Example:
  64K×8 memory
  composed of
  16K×8 components



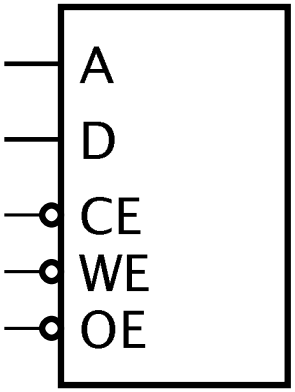


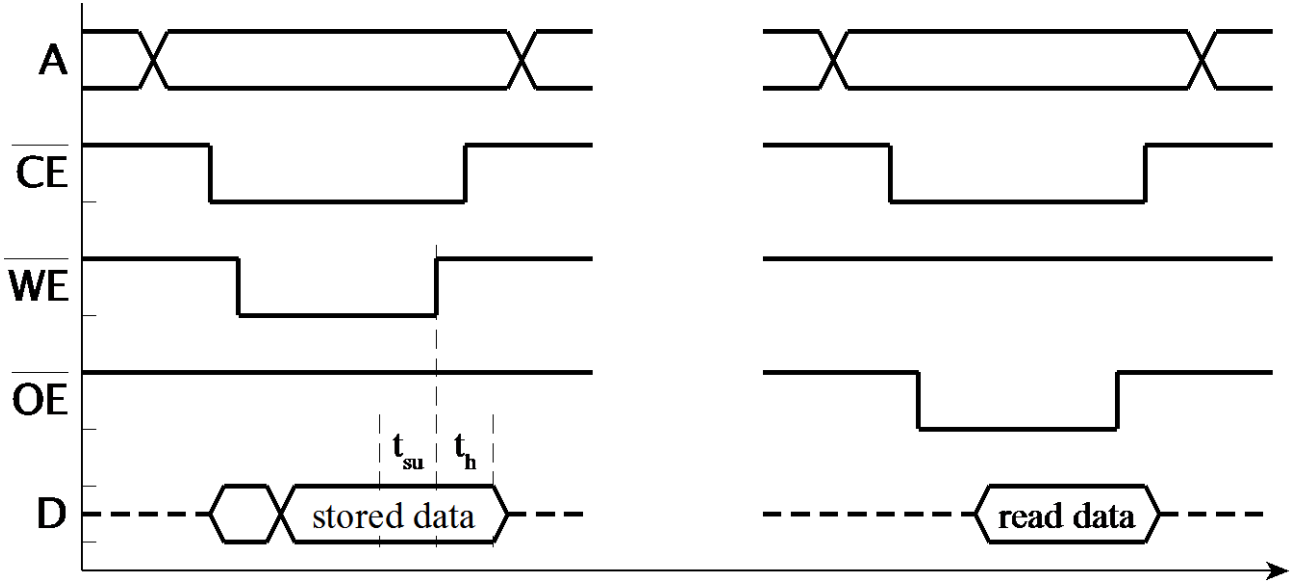
Memory Types

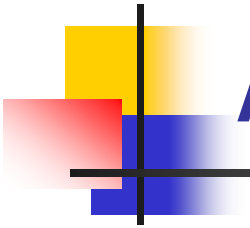
- Random-Access Memory (RAM)
  - Can read and write
  - Static RAM (SRAM)
    - Stores data so long as power is supplied
    - Asynchronous SRAM: not clocked
    - Synchronous SRAM (SSRAM): clocked
  - Dynamic RAM (DRAM)
    - Needs to be periodically refreshed
- Read-Only Memory (ROM)
  - Combinational
  - Programmable and Flash rewritable
- Volatile and non-volatile

Asynchronous SRAM

- Data stored in 1-bit latch cells
  - Address decoded to enable a given cell
- Usually use active-low control inputs
- Not available as components in ASICs or FPGAs







Asynch SRAM Timing

- Timing parameters published in data sheets
- Access time
  - From address/enable valid to data-out valid
- Cycle time
  - From start to end of access
- Data setup and hold
  - Before/after end of WE pulse
  - Makes asynch SRAMs hard to use in clocked synchronous designs

Example Data Sheet

CYPRESS

CY7C1041BV33

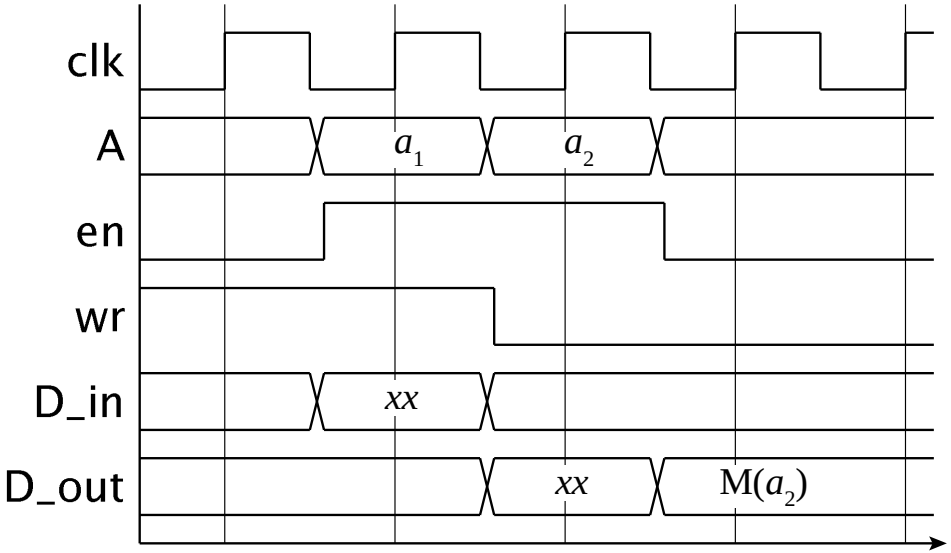
Switching Characteristics[4] Over the Operating Range

Parameter	Description	-12		-15		-17		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
READ CYCLE								
t_{RC}	Read Cycle Time	12		15		17		ns
t_{AA}	Address to Data Valid		12		15		17	ns
t_{OHA}	Data Hold from Address Change	3		3		3		ns
t_{ACE}	$\overline{CE}$ LOW to Data Valid		12		15		17	ns
t_{DOE}	$\overline{OE}$ LOW to Data Valid		6		7		8	ns
WRITE CYCLE[7, 8]								
t_{WC}	Write Cycle Time	12		15		17		ns
t_{SCE}	$\overline{CE}$ LOW to Write End	10		12		12		ns
t_{AW}	Address Set-Up to Write End	10		12		12		ns
t_{HA}	Address Hold from Write End	0		0		0		ns
t_{SA}	Address Set-Up to Write Start	0		0		0		ns
t_{PWE}	$\overline{WE}$ Pulse Width	10		12		12		ns
t_{SD}	Data Set-Up to Write End	7		8		9		ns
t_{HD}	Data Hold from Write End	0		0		0		ns
t_{LZWE}	$\overline{WE}$ HIGH to Low Z[6]	3		3		3		ns
t_{HZWE}	$\overline{WE}$ LOW to High Z[5, 6]		6		7		8	ns
t_{BW}	Byte Enable to End of Write	10		12		12		ns

Synchronous SRAM (SSRAM)

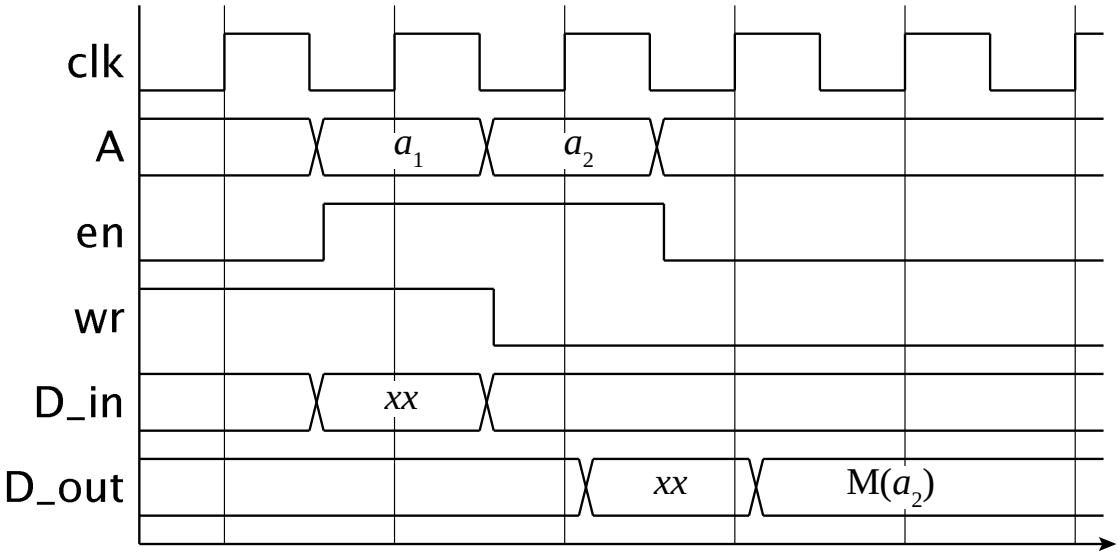
- Clocked storage registers for inputs
  - address, data and control inputs
  - stored on a clock edge
  - held for read/write cycle

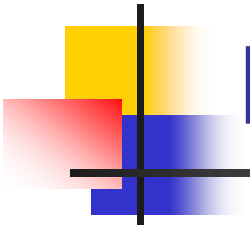
- Flow-through SSRAM
  - no register on data output



Pipelined SSRAM

- Data output also has a register
  - More suitable for high-speed systems
  - Access RAM in one cycle, use the data in the next cycle





Memories in Verilog

- RAM storage represented by an array variable

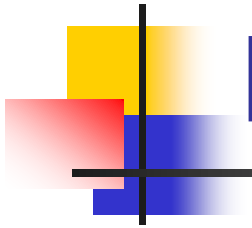
```
reg [15:0] data_RAM [0:4095];
...
always @(posedge clk)
  if (en)
    if (wr) begin
      data_RAM[a] <= d_in;  d_out <= d_in;
    end
    else
      d_out <= data_RAM[a];
```

Example: Coefficient Multiplier

```
module scaled_square ( output reg signed [7:-12] y,
                       input      signed [7:-12] c_in, x,
                       input             [11:0]  i,
                       input                     start,
                       input                     clk, reset );

  wire                c_ram_wr;
  reg                 c_ram_en, x_ce, mult_sel, y_ce;
  reg signed [7:-12]  c_out, x_out;

  reg signed [7:-12]  c_RAM [0:4095];

  reg signed [7:-12]  operand1, operand2;

  parameter [1:0] step1 = 2'b00, step2 = 2'b01, step3 = 2'b10;
  reg       [1:0] current_state, next_state;

  assign c_ram_wr = 1'b0;
```

Example: Coefficient Multiplier

```
always @(posedge clk)  // c RAM - flow through
  if (c_ram_en)
    if (c_ram_wr) begin
      c_RAM[i] <= c_in;
      c_out    <= c_in;
    end
    else
      c_out <= c_RAM[i];

always @(posedge clk)  // y register
  if (y_ce) begin
    if (!mult_sel) begin
      operand1 = c_out;
      operand2 = x_out;
    end
    else begin
      operand1 = x_out;
      operand2 = y;
    end
    y <= operand1 * operand2;
  end
```



Example: Coefficient Multiplier

```
  always @(posedge clk)  // State register
    ...
  always @*  // Next-state logic
    ...
  always @* begin  // Output logic
    ...
endmodule
```

Pipelined SSRAM in Verilog

```
reg         pipelined_en;
reg [15:0] pipelined_d_out;
...

always @(posedge clk) begin
  if (pipelined_en) d_out <= pipelined_d_out;
  pipelined_en <= en;
  if (en)
    if (wr) begin
      data_RAM([a] <= d_in;  pipelined_d_out <= d_in;
    end
    else
      pipelined_d_out <= data_RAM[a];
end
```

output register

SSRAM

Example: RAM Core Generator