

Bit

```

1 // This file is part of www.nand2tetris.org
2 // and the book "The Elements of Computing Systems"
3 // by Nisan and Schocken, MIT Press.
4 // File name: projects/03/a/Bit.hdl
5
6 // File coded by Jared Orange
7
8 /**
9  * 1-bit register:
10  * If load[t] == 1 then out[t+1] = in[t]
11  *                               else out does not change (out[t+1] = out[t])
12  */
13
14 CHIP Bit {
15     IN in, load;
16     OUT out;
17
18     PARTS:
19     // A bit is designated to store a bit of data.
20     // It stores this bit until another bit is stored.
21     Mux(a=out1, b=in, sel=load, out=in1);
22     DFF(in=in1, out=out1, out=out);
23 }
24

```

Hardware Simulator (2.5) - C:\Users\Jared Orange\Nand2Tetris\03\A\Bit.hdl

File View Run Help

[illegible]

End of script - Comparison ended successfully

Register

```

9      * If load[t] == 1 then out[t+1] = in[t]
10     * else out does not change
11     */
12
13 CHIP Register {
14     IN in[16], load;
15     OUT out[16];
16
17     // DOCUMENTATION //
18
19     // A Register is a storage device used to store values over time. A multi bit register
20     // can be constructed from an array of w bit chips (In this case, we are building a 16 bit register
21     // so we need 16 1 bit chips). The input pin carries the data from the numeric input and the load allows
22     // the array cell to be written in. The output pin carries out the current state of the array cell.
23     // The read and write is very similar to the Bit.hdl
24
25     PARTS:
26     // Put your code here:
27     Bit(in=in[0], load = load, out = out[0]);
28     Bit(in=in[1], load = load, out = out[1]);
29     Bit(in=in[2], load = load, out = out[2]);
30     Bit(in=in[3], load = load, out = out[3]);
31     Bit(in=in[4], load = load, out = out[4]);
32     Bit(in=in[5], load = load, out = out[5]);
33     Bit(in=in[6], load = load, out = out[6]);
34     Bit(in=in[7], load = load, out = out[7]);
35     Bit(in=in[8], load = load, out = out[8]);
36     Bit(in=in[9], load = load, out = out[9]);
37     Bit(in=in[10], load = load, out = out[10]);
38     Bit(in=in[11], load = load, out = out[11]);
39     Bit(in=in[12], load = load, out = out[12]);
40     Bit(in=in[13], load = load, out = out[13]);
41     Bit(in=in[14], load = load, out = out[14]);
42     Bit(in=in[15], load = load, out = out[15]);
43 }
44

```

Hardware Simulator (2.5) - C:\Users\Jared Orange\Nand2Tetris\03\A\Register.hdl

File View Run Help

Animate

Program flow

Format

Decimal

View

Script

Slow

Fast

Chip No...

Register (Clocked)

Ti...

74

Input pins		Output pins	
Name	Value	Name	Value
in[16]	32767	out[16]	32767
load	1		

HDL

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press
// File name: projects/03/a/ReRegister.asm

/**
 * 16-bit register:
 * If load[t] == 1 then out[t+1] = in[t]
 * else out does not change
 */

CHIP Register {
    IN in[16], load;
    OUT out[16];
}
```

Internal pins

Name	Value

```
tick,
output;

tock,
output;

set in %B0111111111111111,
set load 0,
tick,
output;

tock,
output;

set load 1,
tick,
output;

tock,
output;

set in %B0111111111111111,
set load 0,
tick,
output;

tock,
output;

set load 1,
tick,
output;

tock,
output;

set load 1,
tick,
output;
```

End of script - Comparison ended successfully

PC

```

6  /**
7   * A 16-bit counter with load and reset control bits.
8   * if      (reset[t] == 1) out[t+1] = 0
9   * else if (load[t] == 1)  out[t+1] = in[t]
10  * else if (inc[t] == 1)   out[t+1] = out[t] + 1 (integer addition)
11  * else
12  *
13  */
14  CHIP PC {
15      IN in[16], load, inc, reset;
16      OUT out[16];
17
18      PARTS:
19      // Put your code here:
20
21      // increment the output of the register
22      Inc16(in = feedback, out = pc);
23
24      // "Sequential chips always consist of a layer of DFFs sandwiched
25      // between optional combinational logic layers"
26      // The next 3 lines are a combinational logic layer to figure
27      // out what gets fed to the register. Either the program counter,
28      // the incremented pc, the input, or zeros on a reset
29
30      Mux16(a = feedback, b = pc, sel = inc, out = w0);
31      Mux16(a = w0, b = in, sel = load, out = w1);
32      Mux16(a = w1, b = false, sel = reset, out = cout);
33
34      // the output from the register also needs to get fed back through
35      // the combinational logic to get processed for the next clock cycle.
36      Register(in = cout, load = true, out = out, out = feedback);
37  }

```

Hardware Simulator (2.5) - Z:\Fall2018\CS2200\CS2200F18HW02\Nand2Tetris\03\PC.hdl

File View Run Help

Chip Name: PC (Clocked) Time: 15

Input pins		Output pins	
Name	Value	Name	Value
in[16]	22222	out[16]	0
load	0		
inc	0		
reset	1		

Internal pins	
Name	Value
feedback[16]	0
pc[16]	1
w0[16]	0
w1[16]	0
cout[16]	0

HDL

```

// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/PC.hdl

/**
 * A 16-bit counter with load and reset control bits.
 * if      (reset[t] == 1) out[t+1] = 0
 * else if (load[t] == 1)  out[t+1] = in[t]
 * else if (inc[t] == 1)   out[t+1] = out[t] + 1 (integer addition)
 * else
 *
 */

CHIP PC {
    IN in[16], load, inc, reset;
    OUT out[16];

    PARTS:
    // Put your code here:

    // increment the output of the register
    Inc16(in = feedback, out = pc);

    // "Sequential chips always consist of a layer of DFFs sandwiched
    // between optional combinational logic layers"
    // The next 3 lines are a combinational logic layer to figure
    // out what gets fed to the register. Either the program counter,
    // the incremented pc, the input, or zeros on a reset

    Mux16(a = feedback, b = pc, sel = inc, out = w0);
    Mux16(a = w0, b = in, sel = load, out = w1);
    Mux16(a = w1, b = false, sel = reset, out = cout);

    // the output from the register also needs to get fed back through
    // the combinational logic to get processed for the next clock cycle.
    Register(in = cout, load = true, out = out, out = feedback);
}

```

End of script - Comparison ended successfully

RAM8

```

6  /**
7   * Memory of 8 registers, each 16 bit-wide. Out holds the value
8   * stored at the memory location specified by address. If load==1, then
9   * the in value is loaded into the memory location specified by address
10  * (the loaded value will be emitted to out from the next time step onward).
11  */
12
13 CHIP RAM8 {
14     IN in[16], load, address[3];
15     OUT out[16];
16
17     PARTS:
18     // Put your code here:
19
20     // RAM8 will consist of 8 16-bit registers.
21     // each register's input is directly connected to the RAM's input
22     Register(in=in, load=load1, out=out1);
23     Register(in=in, load=load2, out=out2);
24     Register(in=in, load=load3, out=out3);
25     Register(in=in, load=load4, out=out4);
26     Register(in=in, load=load5, out=out5);
27     Register(in=in, load=load6, out=out6);
28     Register(in=in, load=load7, out=out7);
29     Register(in=in, load=load8, out=out8);
30
31     // which register is actually being written to (if any) is given by
32     // using a dmux8 to send load to the appropriate register
33     DMux8Way(in=load, a=load1, b=load2, c=load3, d=load4, e=load5, f=load6, g=load7, h=load8, sel=address);
34
35     // The final output will be given by
36     // using a Mux8Way16 to select which register to output
37     Mux8Way16(out=out, a=out1, b=out2, c=out3, d=out4, e=out5, f=out6, g=out7, h=out8, sel=address);
38 }

```

Hardware Simulator (2.5) - Z:\Fall2018\CS2200\CS2200F18HW02\Nand2Tetris\03\RAM8.hdl

File View Run Help

Chip Name: RAM8 (Clocked) Time: 46

Input pins		Output pins	
Name	Value	Name	Value
in[16]	21845	out[16]	21845
load	0		
address[3]	7		

Internal pins	
Name	Value
load1	0
out1[16]	21845
load2	0
out2[16]	21845
load3	0
out3[16]	21845
load4	0
out4[16]	21845
load5	0
out5[16]	21845
load6	0
out6[16]	21845
load7	0
out7[16]	0

HDL

```

// This file is part of www.nand2tetris
// and the book "The Elements of Com
// by Misan and Schocken, MIT Press.
// File name: projects/03/a/RAM8.hdl

/**
 * Memory of 8 registers, each 16 bi
 * stored at the memory location spe
 * the in value is loaded into the m
 * (the loaded value will be emitted
 */

CHIP RAM8 {
    IN in[16], load, address[3];

```

End of script - Comparison ended successfully

RAM64

```

1 // This file is part of www.nand2tetrtris.org
2 // and the book "The Elements of Computing Systems"
3 // by Nisan and Schocken, MIT Press.
4 // File name: projects/03/a/RAM64.hdl
5
6 /**
7  * Memory of 64 registers, each 16 bit-wide. Out holds the value
8  * stored at the memory location specified by address. If load==1, then
9  * the in value is loaded into the memory location specified by address
10  * (the loaded value will be emitted to out from the next time step onward).
11  */
12
13 CHIP RAM64 {
14     IN in[16], load, address[6];
15     OUT out[16];
16
17     PARTS:
18         // RAM64 is an array of 8 RAM8 chips.
19         // RAM64 requires the use of 6 registers (2^6)
20         // Each RAM must be assigned a unique address (some integer between 0 and n-1) in order to be accessed
21         // Each input is loaded corresponding to its data value and the RAM's output will start emitting it
22         // In order to select the register specified by the address, we need a DMux8Way and Mux8Way16
23         // The DMux8Way and Mux8Way16 would be assigned the remaining 3 address.
24         // The DMux8Way would take in the 8 inputs simultaneously and the Mux8Way16 would be fed out the
25         // combinational logic circuit.
26
27         DMux8Way(in=load, sel = address[3..5], a=load1, b=load2, c=load3, d=load4, e=load5, f=load6, g=load7, h=load8);
28         RAM8(in=in, load=load1, address=address[0..2], out=out1);
29         RAM8(in=in, load=load2, address=address[0..2], out=out2);
30         RAM8(in=in, load=load3, address=address[0..2], out=out3);
31         RAM8(in=in, load=load4, address=address[0..2], out=out4);
32         RAM8(in=in, load=load5, address=address[0..2], out=out5);
33         RAM8(in=in, load=load6, address=address[0..2], out=out6);
34         RAM8(in=in, load=load7, address=address[0..2], out=out7);
35         RAM8(in=in, load=load8, address=address[0..2], out=out8);
36         Mux8Way16(a=out1, b=out2, c=out3, d=out4, e=out5, f=out6, g=out7, h=out8, sel=address[3..5], out=out);
37 }

```

Chip Name : **RAM64 (Clocked)**
Time : **81**

Input pins		Output pins	
Name	Value	Name	Value
in[16]	21845	out[16]	21845
load	0		
address[6]	61		

HDL

```

// This file is part of www.nan
// and the book "The Elements o
// by Nisan and Schocken, MIT P
// File name: projects/03/a/RAM

/**
 * Memory of 64 registers, each
 * stored at the memory locatio
 * the in value is loaded into
 * (the loaded value will be em
 */

CHIP RAM64 {
    IN in[16], load, address[6]

```

Internal pins

Name	Value
load1	0
load2	0
load3	0
load4	0
load5	0
load6	0
load7	0
load8	0
out1[16]	21845
out2[16]	21845
out3[16]	21845
out4[16]	21845
out5[16]	21845

```

set load 1,
set address $B111101,
set in $B0101010101010101,
tick,
output,
tock,
output;

set load 0,
set address $B000101,
tick,
output;
tock,
output;
set address $B001101,
eval,
output;
set address $B010101,
eval,
output;
set address $B011101,
eval,
output;
set address $B100101,
eval,
output;
set address $B101101,
eval,
output;
set address $B110101,
eval,
output;
set address $B111101,
eval,
output;

```

End of script - Comparison ended successfully

RAM512

```

9  * Memory of 512 registers, each 16 bit-wide. Out holds the value
10 * stored at the memory location specified by address. If load==1, then
11 * the in value is loaded into the memory location specified by address
12 * (the loaded value will be emitted to out from the next time step onward).
13 */
14
15 CHIP RAM512 {
16     IN in[16], load, address[9];
17     OUT out[16];
18
19     PARTS:
20     // Put your code here:
21
22     // DOCUMENTATION //
23
24     // A RAM unit can only be built using smaller RAM parts. A RAM512
25     // is an array of 8 RAM64 chips (one below in size). RAM512 requires the use of 6 registers (2^6)
26     // Each RAM must be assigned a unique address (some integer between 0 and n-1) in order to be accessed
27     // Each input is loaded corresponding to its data value and the RAM's output will start emitting it
28     // In order to select the register specified by the address, we need a DMux8Way and Mux8Way16
29     // The DMux8Way and Mux8Way16 would be assigned the remaining 2 address.
30     // The DMux8Way would take in the 8 inputs simultaneously and the Mux8Way16 would be fed out the
31     // combinational logic circuit.
32
33     DMux8Way(in=load, sel=address[6..8], a=load1, b=load2, c=load3, d=load4, e=load5, f=load6, g=load7, h=load8);
34     RAM64(in=in, load=load1, address=address[0..5], out=out1);
35     RAM64(in=in, load=load2, address=address[0..5], out=out2);
36     RAM64(in=in, load=load3, address=address[0..5], out=out3);
37     RAM64(in=in, load=load4, address=address[0..5], out=out4);
38     RAM64(in=in, load=load5, address=address[0..5], out=out5);
39     RAM64(in=in, load=load6, address=address[0..5], out=out6);
40     RAM64(in=in, load=load7, address=address[0..5], out=out7);
41     RAM64(in=in, load=load8, address=address[0..5], out=out8);
42     Mux8Way16(a=out1, b=out2, c=out3, d=out4, e=out5, f=out6, g=out7, h=out8, sel=address[6..8], out=out);
43 }
44

```

Hardware Simulator (2.5) - C:\Users\Jared Orange\Nand2Tetris\03\b\RAM512.hdl

File View Run Help

Slow Fast
Animate: Program flow
Format: Decimal
View: Script

Chip Na... **RAM512 (Clocked)** Ti... **81**

Input pins		Output pins	
Name	Value	Name	Value
in[16]	21845	out[16]	21845
load	0		
address[9]	490		

HDL

```

// This file is part of the ma
// "The Elements of Computing
// MIT Press. Book site: www.i
// File name: projects/03/b/RAM

// File coded by Jared Orange

/**
 * Memory of 512 registers, ea
 * stored at the memory locati
 * the in value is loaded into
 * (the loaded value will be e
 */

```

Internal pins

Name	Value
load1	0
load2	0
load3	0
load4	0
load5	0
load6	0
load7	0
load8	0
out1[16]	21845
out2[16]	21845
out3[16]	21845
out4[16]	21845
out5[16]	21845

```

set address %B111101010,
set in %B0101010101010101,
tick,
output,
tick,
output;

set load 0,
set address %B000101010,
tick,
output;
tick,
output;
set address %B001101010,
eval,
output;
set address %B010101010,
eval,
output;
set address %B011101010,
eval,
output;
set address %B100101010,
eval,
output;
set address %B101101010,
eval,
output;
set address %B110101010,
eval,
output;
set address %B111010101,
eval,
output;

```

End of script - Comparison ended successfully

RAM4K

Chip Name: **RAM4K (Clocked)** Time: **81**

Input pins		Output pins	
Name	Value	Name	Value
in[16]	21845	out[16]	21845
load	0		
address[12]	3925		

HDL

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press
// File name: projects/03/b/RAM4K.hdl

/**
 * Memory of 4K registers, each
 * stored at the memory location
 * the in value is loaded into
 * (the loaded value will be emitted
 * from the next time step onward).
 */

CHIP RAM4K {
    IN in[16], load, address[12];
    OUT out[16];
}
```

Internal pins

Name	Value
load1	0
load2	0
load3	0
load4	0
load5	0
load6	0
load7	0
load8	0
out1[16]	21845
out2[16]	21845
out3[16]	21845
out4[16]	21845
out5[16]	21845

Script

```
set load 1,
set address $B111101010101,
set in $B0101010101010101,
tick,
output,
tock,
output;

set load 0,
set address $B000101010101,
tick,
output,
tock,
output;
set address $B001101010101,
eval,
output,
set address $B010101010101,
eval,
output,
set address $B011101010101,
eval,
output,
set address $B100101010101,
eval,
output,
set address $B1010101010101,
eval,
output,
set address $B1110101010101,
eval,
output;
set address $B111101010101,
eval,
output;
```

End of script - Comparison ended successfully

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/b/RAM4K.hdl

/**
 * Memory of 4K registers, each 16 bit-wide. Out holds the value
 * stored at the memory location specified by address. If load==1, then
 * the in value is loaded into the memory location specified by address
 * (the loaded value will be emitted to out from the next time step onward).
 */

CHIP RAM4K {
    IN in[16], load, address[12];
    OUT out[16];

    PARTS:
        // A RAM4K is an array of 8 RAM512 chips.
        // RAM4K requires the use of 12 registers (2^12)
        // Each RAM must be assigned a unique address (some integer between 0 and n-1) in order to be accessed
        // Each input is loaded corresponding to its data value and the RAM's output will start emitting it
        // In order to select the register specified by the address, we need a DMux4Way and Mux4Way16
        // The DMux8Way and Mux8Way16 would be assigned the remaining 3 address.
        // The DMux8Way would take in the 8 inputs simultaneously and the Mux8Way16 would be fed out the
        // combinational logic circuit.
        DMux8Way(in=load, sel=address[9..11], a=load1, b=load2, c=load3, d=load4, e=load5, f=load6, g=load7, h=load8);
        RAM512(in=in, load=load1, address=address[0..8], out=out1);
        RAM512(in=in, load=load2, address=address[0..8], out=out2);
        RAM512(in=in, load=load3, address=address[0..8], out=out3);
        RAM512(in=in, load=load4, address=address[0..8], out=out4);
        RAM512(in=in, load=load5, address=address[0..8], out=out5);
        RAM512(in=in, load=load6, address=address[0..8], out=out6);
        RAM512(in=in, load=load7, address=address[0..8], out=out7);
        RAM512(in=in, load=load8, address=address[0..8], out=out8);
        Mux8Way16(a=out1, b=out2, c=out3, d=out4, e=out5, f=out6, g=out7, h=out8, sel=address[9..11], out=out);
}
```

RAM16K

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/03/b/RAM16K.hdl

// File coded by Jared Orange

/**
 * Memory of 16K registers, each 16 bit-wide. Out holds the value
 * stored at the memory location specified by address. If load==1, then
 * the in value is loaded into the memory location specified by address
 * (the loaded value will be emitted to out from the next time step onward).
 */

CHIP RAM16K {
    IN in[16], load, address[14];
    OUT out[16];

    PARTS:
        // A RAM16K is an array of 4 RAM4K chips (one below in size).
        // RAM16K requires the use of 14 registers (2^14)
        // Each RAM must be assigned a unique address (some integer between 0 and n-1) in order to be accessed
        // Each input is loaded corresponding to its data value and the RAM's output will start emitting it
        // In order to select the register specified by the address, we need a DMux4Way and Mux4Way16
        // The DMux4Way and Mux4Way16 would be assigned the remaining 2 address.
        // The DMux4Way would take in the 4 inputs simultaneously and the Mux4Way16 would be fed out the 64
        // combinational logic circuit.

        // Put your code here:
        DMux4Way(in=load, sel=address[12..13], a=load1, b=load2, c=load3, d=load4);
        RAM4K(in=in, load=load1, address=address[0..11], out=out1);
        RAM4K(in=in, load=load2, address=address[0..11], out=out2);
        RAM4K(in=in, load=load3, address=address[0..11], out=out3);
        RAM4K(in=in, load=load4, address=address[0..11], out=out4);
        Mux4Way16(a=out1, b=out2, c=out3, d=out4, sel=address[12..13], out=out);
}
```

Hardware Simulator (2.5) - C:\Users\Jared Orange\Nand2Tetris\03\b\RAM16K.hdl

File View Run Help

Animate: Program flow Format: Decimal View: Script

Chip Na... RAM16K (Clocked) Ti... 81

Input pins		Output pins	
Name	Value	Name	Value
in[16]	21845	out[16]	21845
load	0		
address[14]	15701		

Internal pins	
Name	Value
load1	0
load2	0
load3	0
load4	0
out1[16]	21845
out2[16]	21845
out3[16]	21845
out4[16]	21845

HDL

```
// In order to select the
// The DMux4Way and Mux4Way16
// The DMux4Way would take in
// combinational logic circuit

// Put your code here:
DMux4Way(in=load, sel=address[
RAM4K(in=in, load=load1, addre
RAM4K(in=in, load=load2, addre
RAM4K(in=in, load=load3, addre
RAM4K(in=in, load=load4, addre
Mux4Way16(a=out1, b=out2, c=ou
}
```

```
set load 1,
set address $B11110101010101,
set in $B0101010101010101,
tick,
output,
tock,
output;

set load 0,
set address $B00010101010101,
tick,
output;
tock,
output;
set address $B00110101010101,
eval,
output;
set address $B01010101010101,
eval,
output;
set address $B01110101010101,
eval,
output;
set address $B10010101010101,
eval,
output;
set address $B10110101010101,
eval,
output;
set address $B11010101010101,
eval,
output;
set address $B11110101010101,
eval,
output;
```

End of script - Comparison ended successfully

Are the test adequate or inadequate?

RAMs: The RAM tests are adequate. The sole purpose of RAM is to provide fast read/write access to a storage device. The tst files load in an address and use ticks and tocks to count how fast is reads/writes to the device by displaying in between functions. If anything, we would have added some sort of device to calculate speed in MHz (like typical RAM inside a computer is measured), a test to find the max slots used when read/writing to the device and then a test comparing the runtime versus an actual hard drive.

Bit: Adequate. All that a bit does is store data and execute instructions in bytes. The tst file sets a value, loads it, starts counting using ticks and tocks and outputs the executable time of storing the data.

Register: Inadequate. Register and Bit have the same read/write functions, so testing Bit first is all that is necessary since n Register is made up of n Bits.

PC: Adequate. The Program Counter is working with the reset, load, and inc, (the if-else statement in the comment section) total of 3 bits and that gives us $2^3=8$ cases. In the test file, it had checked all the cases/combinations of reset, load, and inc. Therefore, the test is adequate.