(* register_duplication = "{yes|no}" *)

XCF Syntax Example One

MODEL "*entity_name*" **register_duplication={yes|no|true|false};**

XCF Syntax Example Two

BEGIN MODEL "*entity_name*"

NET "*signal_name*" **register_duplication={yes|no|true|false};**

END;

ISE® Design Suite

Process > Process Properties > Xilinx Specific Options > Register Duplication

ROM Extraction (ROM_EXTRACT)

The ROM Extraction (**ROM_EXTRACT**) constraint enables or disables ROM macro inference.

Typically, a ROM can be inferred from a **case** statement where all assigned contexts are constant values.

Architecture Support

Applies to all FPGA devices. Does not apply to CPLD devices.

Applicable Elements

Applies to the entire design, or to a design element or signal.

Propagation Rules

Applies to the entity, component, module, or signal to which it is attached.

Syntax

-rom_extract {yes|no}

- **yes** (default)
- **no**
- **true** (XCF only)
- **false** (XCF only)

Syntax Examples and Settings

The following syntax examples and settings show how to use this constraint or command line option with particular tools or methods. If a tool or method is not listed, you cannot use this constraint or command line option with it.

VHDL

Declare as follows:

attribute rom_extract: string;

Specify as follows:

attribute rom_extract of {*signal_name*|*entity_name* **} : {signal|entity} is "{yes|no}";**

Verilog

Place immediately before the module or signal declaration:

(* rom_extract = "{yes|no}" *)

XCF Syntax Example One

MODEL "*entity_name*" **rom_extract={yes|no|true|false};**

XCF Syntax Example Two

BEGIN MODEL "*entity_name*"

NET "*signal_name*" **rom_extract={yes|no|true|false};**

END;

XST Command Line

xst run -rom_extract {yes|no}

ISE® Design Suite

Process > Process Properties > HDL Options > ROM Extraction

ROM Style (ROM_STYLE)

The ROM Style (**ROM_STYLE**) constraint controls the way the macrogenerator implements the inferred ROM macros.

Caution! ROM Extraction (ROM_EXTRACT) must be set to **yes** in order to use ROM Style (ROM_STYLE).

Architecture Support

Applies to all FPGA devices. Does not apply to CPLD devices.

Applicable Elements

Applies to the entire design, or to an entity, component, module, or signal.

Propagation Rules

Applies to the entity, component, module, or signal to which it is attached.

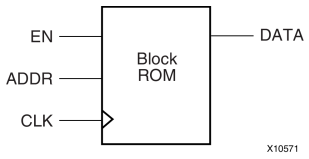
ROM With Registered Output Verilog Coding Example One

```
//
// ROMs Using Block RAM Resources.
// Verilog code for a ROM with registered output (template 1)
//


module v_rams_21a (clk, en, addr, data);

    input      clk;
    input      en;
    input      [5:0] addr;
    output reg [19:0] data;

    always @(posedge clk) begin
        if (en)
            case(addr)
                6'b000000: data <= 20'h0200A;   6'b100000: data <= 20'h02222;
                6'b000001: data <= 20'h00300;   6'b100001: data <= 20'h04001;
                6'b000010: data <= 20'h08101;   6'b100010: data <= 20'h00342;
                6'b000011: data <= 20'h04000;   6'b100011: data <= 20'h0232B;
                6'b000100: data <= 20'h08601;   6'b100100: data <= 20'h00900;
                6'b000101: data <= 20'h0233A;   6'b100101: data <= 20'h00302;
                6'b000110: data <= 20'h00300;   6'b100110: data <= 20'h00102;
                6'b000111: data <= 20'h08602;   6'b100111: data <= 20'h04002;
                6'b001000: data <= 20'h02310;   6'b101000: data <= 20'h00900;
                6'b001001: data <= 20'h0203B;   6'b101001: data <= 20'h08201;
                6'b001010: data <= 20'h08300;   6'b101010: data <= 20'h02023;
                6'b001011: data <= 20'h04002;   6'b101011: data <= 20'h00303;
                6'b001100: data <= 20'h08201;   6'b101100: data <= 20'h02433;
                6'b001101: data <= 20'h00500;   6'b101101: data <= 20'h00301;
                6'b001110: data <= 20'h04001;   6'b101110: data <= 20'h04004;
                6'b001111: data <= 20'h02500;   6'b101111: data <= 20'h00301;
                6'b010000: data <= 20'h00340;   6'b110000: data <= 20'h00102;
                6'b010001: data <= 20'h00241;   6'b110001: data <= 20'h02137;
                6'b010010: data <= 20'h04002;   6'b110010: data <= 20'h02036;
                6'b010011: data <= 20'h08300;   6'b110011: data <= 20'h00301;
                6'b010100: data <= 20'h08201;   6'b110100: data <= 20'h00102;
                6'b010101: data <= 20'h00500;   6'b110101: data <= 20'h02237;
                6'b010110: data <= 20'h08101;   6'b110110: data <= 20'h04004;
                6'b010111: data <= 20'h00602;   6'b110111: data <= 20'h00304;
                6'b011000: data <= 20'h04003;   6'b111000: data <= 20'h04040;
                6'b011001: data <= 20'h0241E;   6'b111001: data <= 20'h02500;
                6'b011010: data <= 20'h00301;   6'b111010: data <= 20'h02500;
                6'b011011: data <= 20'h00102;   6'b111011: data <= 20'h02500;
                6'b011100: data <= 20'h02122;   6'b111100: data <= 20'h0030D;
                6'b011101: data <= 20'h02021;   6'b111101: data <= 20'h02341;
                6'b011110: data <= 20'h00301;   6'b111110: data <= 20'h08201;
                6'b011111: data <= 20'h00102;   6'b111111: data <= 20'h0400D;
            endcase
    end

endmodule
```

ROM With Registered Output Verilog Coding Example Two

```
//
// ROMs Using Block RAM Resources.
// Verilog code for a ROM with registered output (template 2)
//

module v_rams_21b (clk, en, addr, data);

    input      clk;
    input      en;
    input      [5:0] addr;
    output reg [19:0] data;
    reg        [19:0] rdata;


    always @(addr) begin
        case(addr)
                6'b000000: rdata <= 20'h0200A;   6'b100000: rdata <= 20'h02222;
                6'b000001: rdata <= 20'h00300;   6'b100001: rdata <= 20'h04001;
                6'b000010: rdata <= 20'h08101;   6'b100010: rdata <= 20'h00342;
                6'b000011: rdata <= 20'h04000;   6'b100011: rdata <= 20'h0232B;
                6'b000100: rdata <= 20'h08601;   6'b100100: rdata <= 20'h00900;
                6'b000101: rdata <= 20'h0233A;   6'b100101: rdata <= 20'h00302;
                6'b000110: rdata <= 20'h00300;   6'b100110: rdata <= 20'h00102;
                6'b000111: rdata <= 20'h08602;   6'b100111: rdata <= 20'h04002;
                6'b001000: rdata <= 20'h02310;   6'b101000: rdata <= 20'h00900;
                6'b001001: rdata <= 20'h0203B;   6'b101001: rdata <= 20'h08201;
                6'b001010: rdata <= 20'h08300;   6'b101010: rdata <= 20'h02023;
                6'b001011: rdata <= 20'h04002;   6'b101011: rdata <= 20'h00303;
                6'b001100: rdata <= 20'h08201;   6'b101100: rdata <= 20'h02433;
                6'b001101: rdata <= 20'h00500;   6'b101101: rdata <= 20'h00301;
                6'b001110: rdata <= 20'h04001;   6'b101110: rdata <= 20'h04004;
                6'b001111: rdata <= 20'h02500;   6'b101111: rdata <= 20'h00301;
                6'b010000: rdata <= 20'h00340;   6'b110000: rdata <= 20'h00102;
                6'b010001: rdata <= 20'h00241;   6'b110001: rdata <= 20'h02137;
                6'b010010: rdata <= 20'h04002;   6'b110010: rdata <= 20'h02036;
                6'b010011: rdata <= 20'h08300;   6'b110011: rdata <= 20'h00301;
                6'b010100: rdata <= 20'h08201;   6'b110100: rdata <= 20'h00102;
                6'b010101: rdata <= 20'h00500;   6'b110101: rdata <= 20'h02237;
                6'b010110: rdata <= 20'h08101;   6'b110110: rdata <= 20'h04004;
                6'b010111: rdata <= 20'h00602;   6'b110111: rdata <= 20'h00304;
                6'b011000: rdata <= 20'h04003;   6'b111000: rdata <= 20'h04040;
                6'b011001: rdata <= 20'h0241E;   6'b111001: rdata <= 20'h02500;
                6'b011010: rdata <= 20'h00301;   6'b111010: rdata <= 20'h02500;
                6'b011011: rdata <= 20'h00102;   6'b111011: rdata <= 20'h02500;
                6'b011100: rdata <= 20'h02122;   6'b111100: rdata <= 20'h0030D;
                6'b011101: rdata <= 20'h02021;   6'b111101: rdata <= 20'h02341;
                6'b011110: rdata <= 20'h00301;   6'b111110: rdata <= 20'h08201;
                6'b011111: rdata <= 20'h00102;   6'b111111: rdata <= 20'h0400D;
        endcase
    end

    always @(posedge clk) begin
        if (en)
            data <= rdata;
    end
endmodule
```

ROM With Registered Address Diagram



ROM With Registered Address Pin Descriptions

IO Pins	Description

IO Pins	Description
clk	Positive-Edge Clock
en	Synchronous Enable (active-High)
addr	Read Address
data	Data Output
clk	Positive-Edge Clock

To infer ROM in BRAM, you must register either the output or the address

ROM With Registered Address Verilog Coding Example

```
//
// ROMs Using Block RAM Resources.
// Verilog code for a ROM with registered address
//

module v_rams_21c (clk, en, addr, data);

    input      clk;
    input      en;
    input      [5:0] addr;
    output reg [19:0] data;
    reg        [5:0] raddr;

    always @(posedge clk) begin
        if (en)
            raddr <= addr;
    end

    always @(raddr) begin
       case(raddr)
                6'b000000: data <= 20'h0200A;   6'b100000: data <= 20'h02222;
                6'b000001: data <= 20'h00300;   6'b100001: data <= 20'h04001;
                6'b000010: data <= 20'h08101;   6'b100010: data <= 20'h00342;
                6'b000011: data <= 20'h04000;   6'b100011: data <= 20'h0232B;
                6'b000100: data <= 20'h08601;   6'b100100: data <= 20'h00900;
                6'b000101: data <= 20'h0233A;   6'b100101: data <= 20'h00302;
                6'b000110: data <= 20'h00300;   6'b100110: data <= 20'h00102;
                6'b000111: data <= 20'h08602;   6'b100111: data <= 20'h04002;
                6'b001000: data <= 20'h02310;   6'b101000: data <= 20'h00900;
                6'b001001: data <= 20'h0203B;   6'b101001: data <= 20'h08201;
                6'b001010: data <= 20'h08300;   6'b101010: data <= 20'h02023;
                6'b001011: data <= 20'h04002;   6'b101011: data <= 20'h00303;
                6'b001100: data <= 20'h08201;   6'b101100: data <= 20'h02433;
                6'b001101: data <= 20'h00500;   6'b101101: data <= 20'h00301;
                6'b001110: data <= 20'h04001;   6'b101110: data <= 20'h04004;
                6'b001111: data <= 20'h02500;   6'b101111: data <= 20'h00301;
                6'b010000: data <= 20'h00340;   6'b110000: data <= 20'h00102;
                6'b010001: data <= 20'h00241;   6'b110001: data <= 20'h02137;
                6'b010010: data <= 20'h04002;   6'b110010: data <= 20'h02036;
                6'b010011: data <= 20'h08300;   6'b110011: data <= 20'h00301;
                6'b010100: data <= 20'h08201;   6'b110100: data <= 20'h00102;
                6'b010101: data <= 20'h00500;   6'b110101: data <= 20'h02237;
                6'b010110: data <= 20'h08101;   6'b110110: data <= 20'h04004;
                6'b010111: data <= 20'h00602;   6'b110111: data <= 20'h00304;
                6'b011000: data <= 20'h04003;   6'b111000: data <= 20'h04040;
                6'b011001: data <= 20'h0241E;   6'b111001: data <= 20'h02500;
                6'b011010: data <= 20'h00301;   6'b111010: data <= 20'h02500;
                6'b011011: data <= 20'h00102;   6'b111011: data <= 20'h02500;
                6'b011100: data <= 20'h02122;   6'b111100: data <= 20'h0030D;
                6'b011101: data <= 20'h02021;   6'b111101: data <= 20'h02341;
                6'b011110: data <= 20'h00301;   6'b111110: data <= 20'h08201;
                6'b011111: data <= 20'h00102;   6'b111111: data <= 20'h0400D;
        endcase
    end

endmodule
```