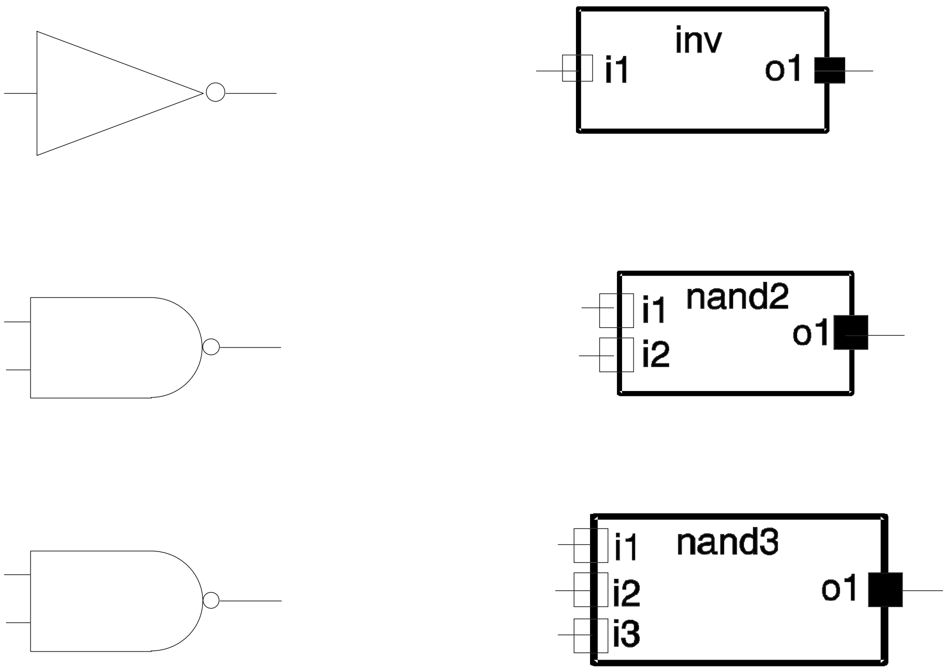
# PART 4

## Structural_Description_of_Hardware

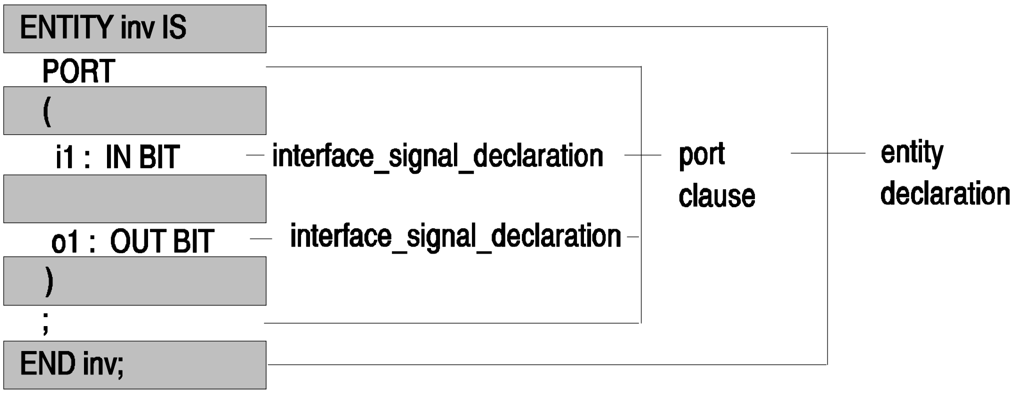
- Wiring components into larger designs
- Start with primitives
- Gates as library part
- Wire gates into general purpose components
- Use iterative constructs
- Generate test benches
- Show binding alternatives
- Use gate-based components for a larger design
- Examples include
  - Basic gates
  - Single bit comparator
  - n-bit comparator
  - Latch design: 1 bit SR-latch, a D-latch, a 4 bit D-latch
  - An old-new comparator

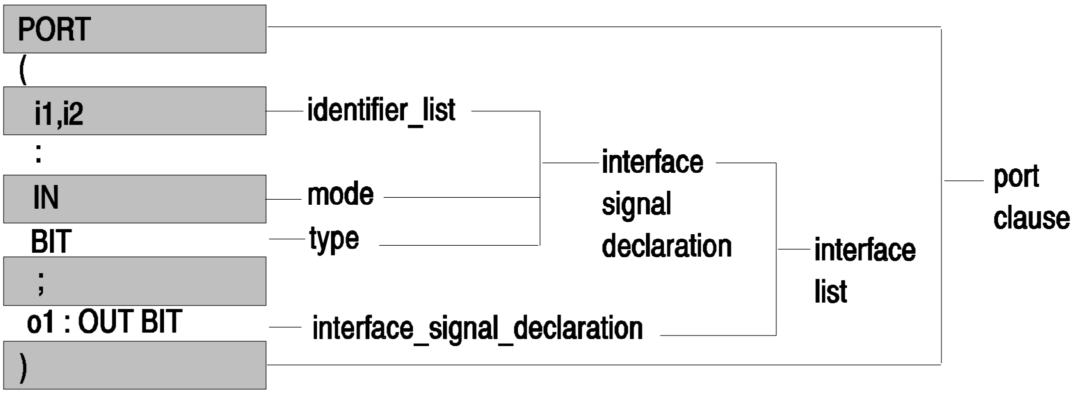
# Structural_specification.basic_components




- Three basic components are used
- Will use as the basic components of several designs
- A graphical notation helps clarify wiring

# Structural...basic_components.syntax_details

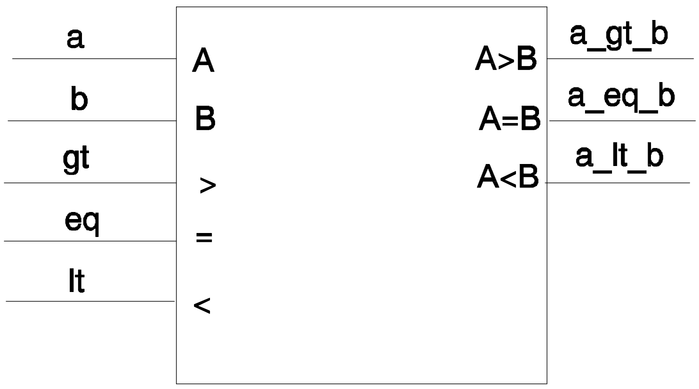



- Figure shows syntax of *inv*
- Figure shows port clause of *nand2*

# Structural...single_bit_comparator.logic_design




| > \ a,b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | | 1 |
| 1 | 1 | | 1 | 1 |

a>b

| = \ a,b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | 1 | | 1 | |

a=b

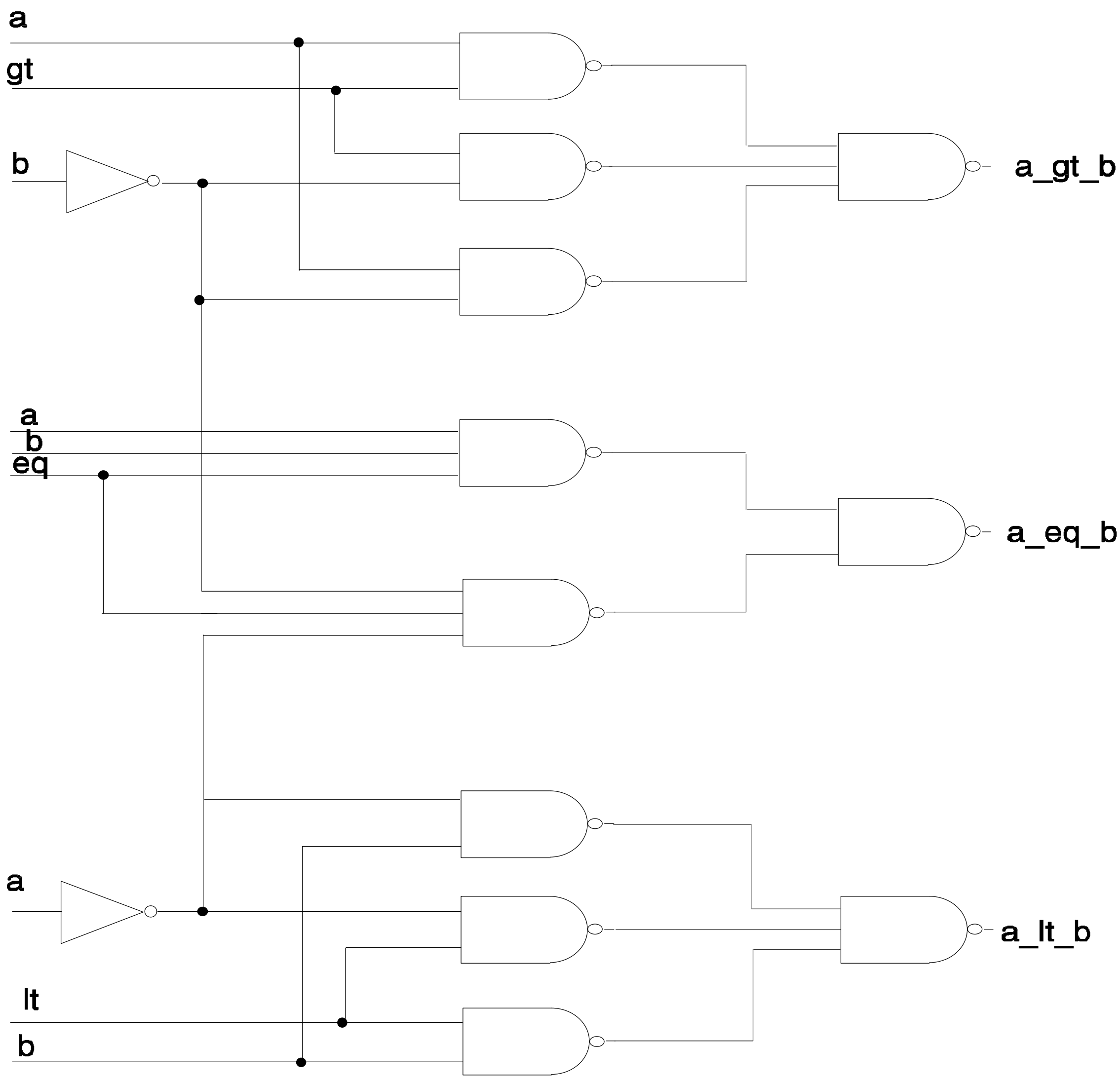
| < \ a,b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | 1 | | |
| 1 | 1 | 1 | 1 | |

a<b

- A cascadable bit comparator
- When *a* > *b* the *a_gt_b* becomes 1
- When *a* < *b* the *a_lt_b* becomes 1
- If *a* = *b* outputs become the same as corresponding inputs

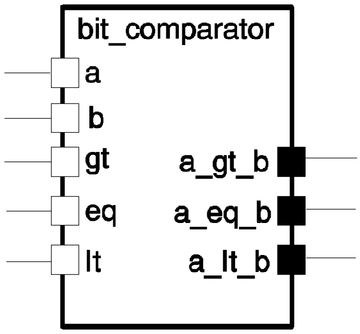
# Structural...single bit comparator.logic_design

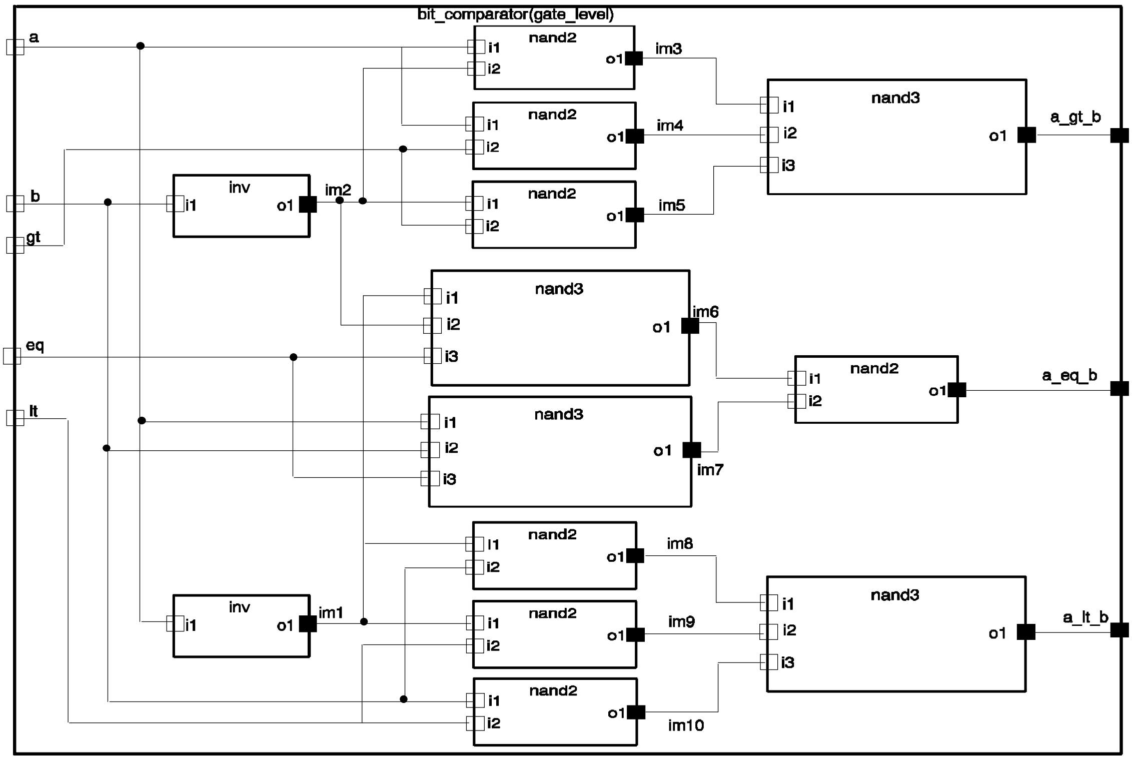



- Logic design uses our basic components
- The less-than and greater-than outputs use the same logic

# Structural...single_bit_comparator.aspect_notation




- Interface aspect and composition aspect help clarify wiring

# Structural...single_bit_comparator.description

```
ENTITY bit_comparator IS
  PORT (a, b,                  -- data inputs
       gt,                     -- previous greater than
       eq,                     -- previous equal
       lt : IN BIT;            -- previous less than
       a_gt_b,                 -- greater
       a_eq_b,                  -- equal
       a_lt_b : OUT BIT);      -- less than
END bit_comparator;
--
ARCHITECTURE gate_level OF bit_comparator IS
   . . .
BEGIN
   . . .
END gate_level;
```

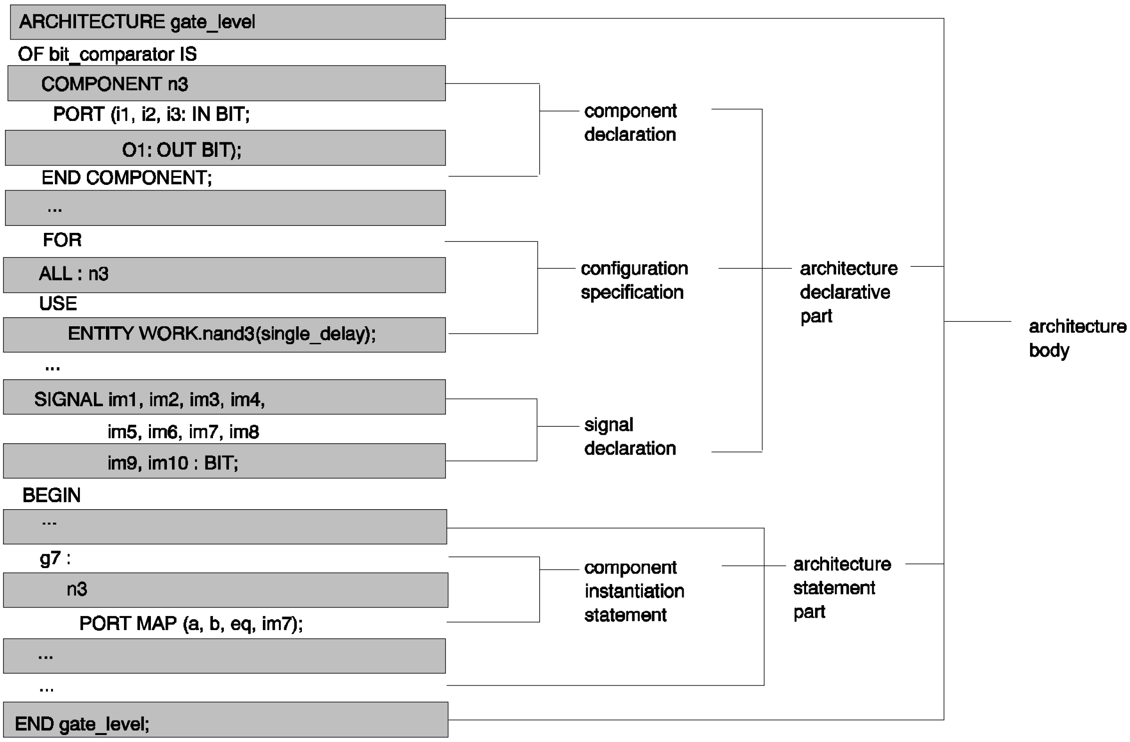
- VHDL description consists of an entity and an architecture
- Interface description is done by the entity declaration
- Description of the operation will be given in the architecture
- Signal names of interface and composition aspects are used

# Structural...single_bit_comparator.description

```
ENTITY bit_comparator IS
  . . .
END bit_comparator;
--
ARCHITECTURE gate_level OF bit_comparator IS
  COMPONENT n1 PORT (i1: IN BIT; o1: OUT BIT);   END COMPONENT;
  COMPONENT n2 PORT (i1, i2: IN BIT; o1: OUT BIT);   END
COMPONENT;
  COMPONENT n3 PORT (i1, i2, i3: IN BIT; o1: OUT BIT);
  END COMPONENT;
  FOR ALL : n1 USE ENTITY WORK.inv (single_delay);
  FOR ALL : n2 USE ENTITY WORK.nand2 (single_delay);
  FOR ALL : n3 USE ENTITY WORK.nand3 (single_delay);
-- Intermediate signals
  SIGNAL      im1,im2, im3, im4, im5, im6, im7, im8, im9, im10 : BIT;
BEGIN
-- a_gt_b output
  g0 : n1 PORT MAP (a, im1);
  g1 : n1 PORT MAP (b, im2);
  g2 : n2 PORT MAP (a, im2, im3);
  g3 : n2 PORT MAP (a, gt, im4);
  g4 : n2 PORT MAP (im2, gt, im5);
  g5 : n3 PORT MAP (im3, im4, im5, a_gt_b);
-- a_eq_b output
  g6 : n3 PORT MAP (im1, im2, eq, im6);   g7 : n3 PORT MAP (a, b, eq, im7);
  g8 : n2 PORT MAP (im6, im7, a_eq_b);
-- a_lt_b output
  g9 : n2 PORT MAP (im1, b, im8);
  g10 : n2 PORT MAP (im1, lt, im9);
  g11 : n2 PORT MAP (b, lt, im10);
  g12 : n3 PORT MAP (im8, im9, im10, a_lt_b);
END gate_level;
```

- Architecture completes the VHDL description

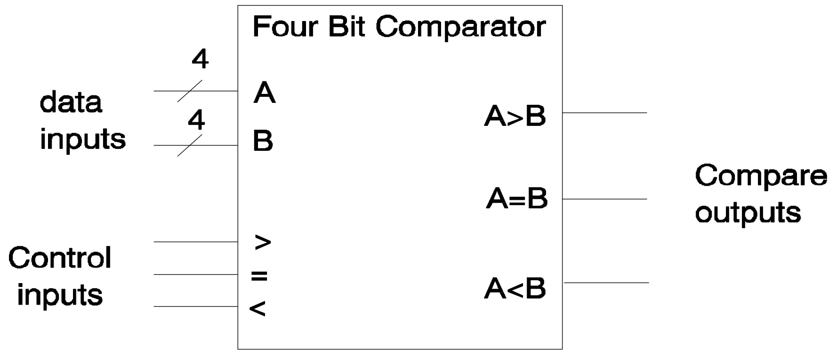
# Structural...single_bit_comparator.syntax_details

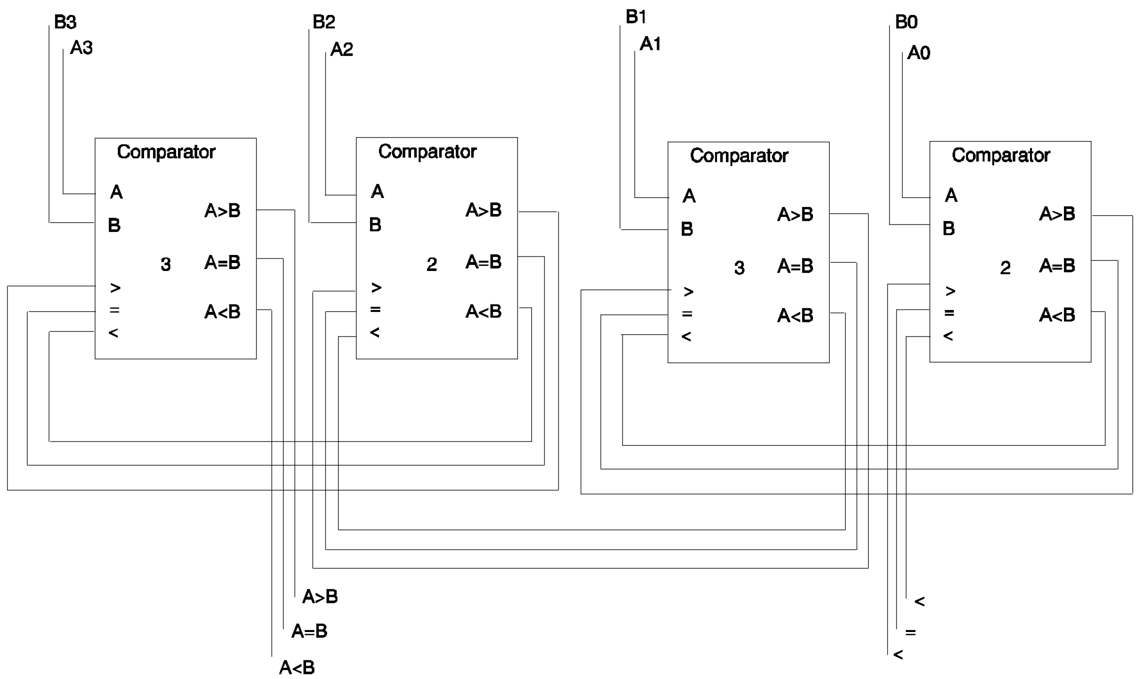



- Syntax details shows syntax of architecture declaration and statement part

# Structural...iterative_wiring.logic_diagram

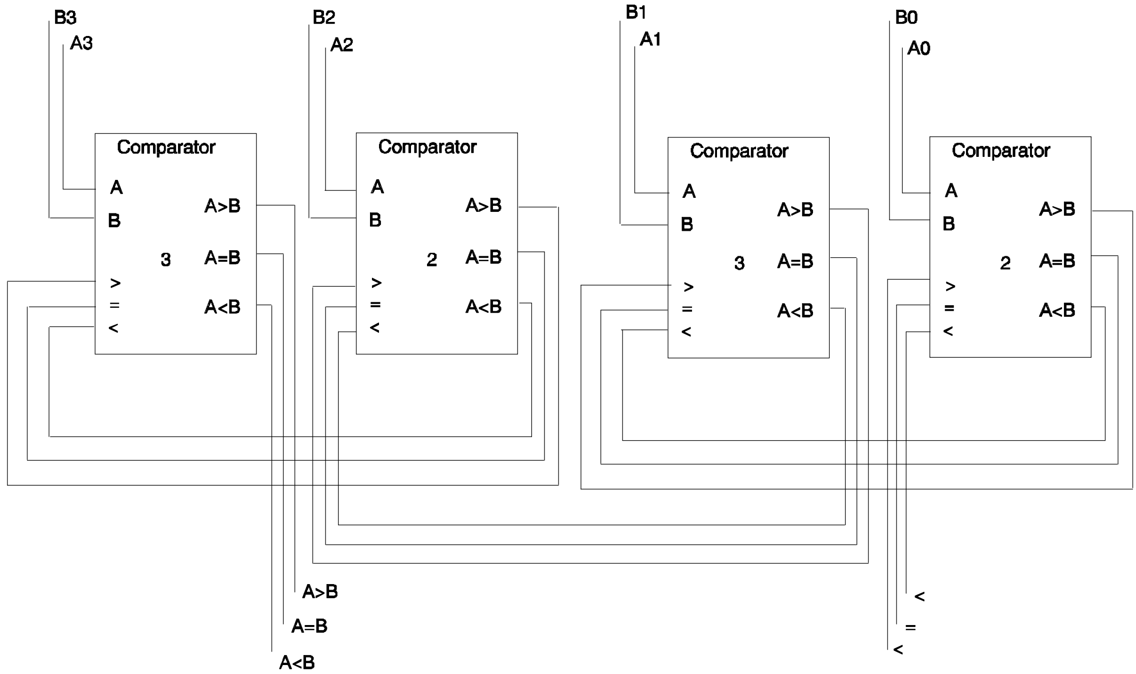



- Cascading single bit into a four bit comparator
- Least significant control inputs control outputs when *A=B*

# Structural...iterative_wiring.logic_diagram

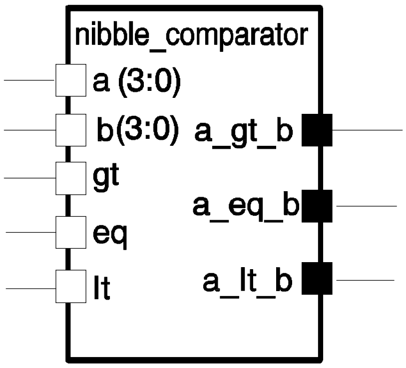



| | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|
| **A:** | **0** | **1** | **0** | **0** | **result propagates from bit 1** |
| **B:** | **0** | **1** | **1** | **0** | |
| **A:** | **1** | **0** | **1** | **1** | **bit 3 produces the result immediately** |
| **B:** | **0** | **0** | **1** | **1** | |

- Results propagate from least to most

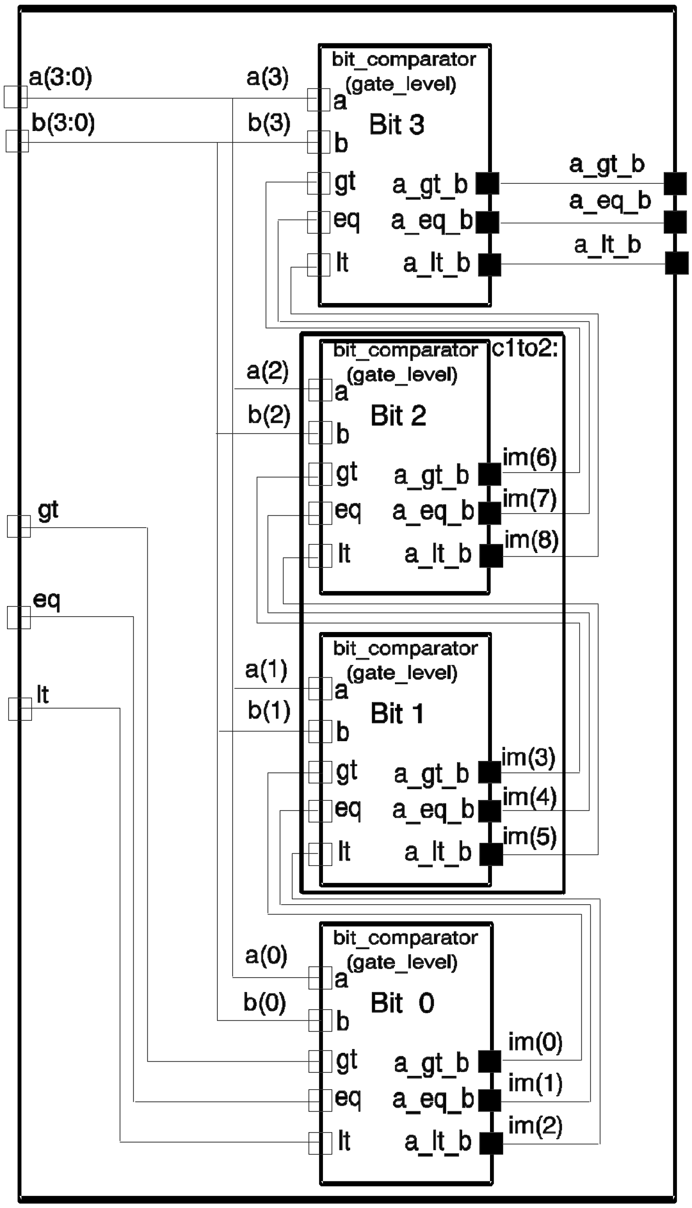
 

# Structural...iterative_wiring.aspect_notation




- Cascading single bit into a four bit comparator
- Signal names inside the box are visible in the architecture

# Structural...iterative_wiring.aspect_notation




- Aspect notation shows VHDL wiring
- VHDL description will use same intermediate signal names

# Structural...iterative_wiring.VHDL_description

```
ENTITY nibble_comparator IS
  PORT (a, b : IN BIT_VECTOR (3 DOWNTO 0);     -- a and b data inputs
        gt,                                    -- previous greater
        eq,                                    -- previous equal
        lt : IN BIT;                           -- previous less than
        a_gt_b,                                -- a > b
        a_eq_b,                                -- a = b
        a_lt_b : OUT BIT);                     -- a < b
END nibble_comparator;
--
ARCHITECTURE iterative OF nibble_comparator IS
  COMPONENT comp1
    PORT (a, b, gt, eq, lt : IN BIT; a_gt_b, a_eq_b, a_lt_b : OUT BIT);
  END COMPONENT;
  FOR ALL : comp1 USE ENTITY WORK.bit_comparator (gate_level);
  SIGNAL im : BIT_VECTOR ( 0 TO 8);
BEGIN
  c0: comp1 PORT MAP (a(0), b(0), gt, eq, lt, im(0), im(1), im(2));
  c1to2: FOR i IN 1 TO 2 GENERATE
    c: comp1 PORT MAP (a(i), b(i), im(i*3-3), im(i*3-2), im(i*3-1),
                 im(i*3+0), im(i*3+1), im(i*3+2) );
  END GENERATE;
  c3: comp1 PORT MAP (a(3), b(3), im(6), im(7), im(8),
                a_gt_b, a_eq_b, a_lt_b);
END iterative;
```

- Use BIT_VECTOR for data inputs
- Separate first and last bits from others
- Arrays for intermediate signals facilitate iterative wiring
- Can easily expand to an n-bit comparator