

# BA2 DSD Résumé

**Logical operators:**

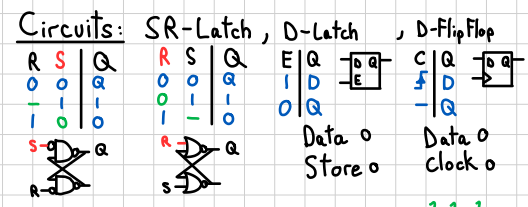
	conjunction "and"		disjunction "or"		exclusive or (logic exclusion) "xor"
	A	B	A · B	A + B	A ⊕ B = $\bar{A} \cdot B + A \cdot \bar{B}$
n inputs → 2 <sup>n</sup> combinations					
"not" $\bar{A}$	0	0	0	0	0
A $\neg A$	1	0	0	1	1
1	0	1	0	1	1
0	1	1	1	1	0
first → last	ANSI				
$\neg (\cdot) \cdot \oplus +$	IEC				

**Properties:**  $A \cdot A = A$   $A + A = A$   $A \oplus A = 0$   $A \cdot 0 = 0$   $A \cdot 1 = A$   $A + 0 = A$   $A + 1 = 1$   $A \oplus 0 = A$   $A \oplus 1 = \bar{A}$   
 $A \cdot (A+B) = A$   $A + (A \cdot B) = A$   $A \cdot \bar{A} = 0$   $A + \bar{A} = 1$   $A \oplus \bar{A} = 1$   $\bar{\bar{A}} = A$   
 $A \cdot (B+C) = (A \cdot B) + (A \cdot C)$   $A + (B \cdot C) = (A+B) \cdot (A+C)$   $A \cdot (B \oplus C) = (A \cdot B) \oplus (A \cdot C)$   $A \oplus (B \oplus C) = (A \oplus B) \oplus C$   
 $\Delta A \oplus (B \cdot C) \neq (A \oplus B) \cdot (A \oplus C)$   
**"NAND"**  $\bar{A} \cdot B = \overline{A + B}$  **"NOR"**  $\overline{A + B} = \bar{A} \cdot \bar{B}$

**Karnaugh groups:** 2<sup>m</sup> minterms/maxterms → m variables don't care (Symmetry) (E)  
 var E is don't care when 2<sup>m-1</sup> minterms/maxterms where E=1 and 2<sup>m-1</sup> where E=0  
 var. E influences when all 2<sup>m</sup> minterms/maxterms are where E=1 or where E=0

**Analogy:** range of values  
**Digital:** two values (1, 0)  
 0.F 1.T  
**Result = 1:** minterms  
**Result = 0:** maxterms  
 f: positive edge,  $\bar{f}$ : negative edge

**disjunctive form** = sum of products  
**conjunctive form** = product of sums



**Clock:** T<sub>1</sub> time logic level 1  
 T<sub>0</sub> time logic level 0  
 T<sub>p</sub> = T<sub>1</sub> + T<sub>0</sub>, f<sub>c</sub> = 1/T<sub>p</sub>

**Representations:** nibble: 4 bits, byte: 8 bits, MSB...LSB, encoding: thermometer: 00011111b 00000001b  
 Sign & Magnitude: 00b +0, 10b -0, 11b -0, 101b -5, 1101b -5  
 invert 00b +0, 11b -0, 1011b -5, 1011b -5  
 add 2 numbers -MSB, and + carry, two's comp. [2<sup>k-1</sup>, 2<sup>k-1</sup>-1]  
 excess N, fixed point 11.01 3,25, [N, 2<sup>k</sup>-N-1]

**Operations:** dec → B=7, 768<sub>8</sub>=10, 2145<sub>8</sub>=7  
 Adder:  $\begin{matrix} & 1 & 1 & 1 \\ & 2 & 1 & 2 & B=3 \\ + & 2 & 1 & B=3 \\ \hline 1 & 0 & 1 & 0 & B=3 \end{matrix}$   
 Subtract:  $\begin{matrix} & 1 & 1 & 1 \\ & 1 & 1 & 2 & B=4 \\ - & 1 & 3 & B=4 \\ \hline 0 & 1 & 3 & B=4 \end{matrix}$   
 Div:  $\begin{matrix} 1011 & 10b \\ : & 0101b \\ \hline 011 & \\ - & 011 \\ \hline 0 & \\ & 10 \\ & - & 10 \\ & & 0 & \end{matrix}$   
 Mult:  $\begin{matrix} 0101b \\ \times & 11b \\ \hline 0101b \\ + & 0101b \\ \hline 0111b \\ \text{Overflow} & 111b \end{matrix}$

**FSM:** code table to binary State → Code, transition table + inputs to new outputs, Output table Code → encoding, never use the clock, FSM: V inputs possibilities at least (complete) & at most (consistent) one active transition

**CT + TT:** Medvedev | + OT: Moore, + to output logic: Mealy  
 encoding n states:  $\lceil \log_2(n) \rceil$  bits, unused codes: ghost states, POR: power-on-reset directly to memory (asynchronous), gated clocks

**Multiplexer:**  $\begin{matrix} c & Q \\ 0 & a \\ 1 & b \end{matrix}$   $Q = bc + a \cdot \bar{c}$   
 Shift-Register:  $\begin{matrix} D & Q & Q & Q & Q \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \oplus & \oplus & \oplus & \oplus & \oplus \\ c & & & & \end{matrix}$

**None Ideal behavior:** TTL: transistor to transistor logic, Hazards: gates have delay ⇒ don't work, D signal must be stable during t<sub>setup</sub> → Metastability or 1  
 Programmable Logic: ASIC: Application Specific Integrated Circuit, PAL: Programmable Array Logic, OTP: one time programmable, GAL: Generic Array Logic

**CPLD:** Complex Programmable Logic Device, IOB: Input Output Block, FPGA: Field Programmable Gate Array, LUT: Look up Table, P&R software → bit-File, RAM: random access memory, SOC: System on chip  
 HDL — synthesizer → gates implementation

**VHDL:** case-insensitive, comments --, Entity: black-box, Architecture: functionality, std\_logic: a bit, std\_logic\_vector(n-1 DOWNTO 0): n bits vector, bit value ∈ {0, 1, 'u', 'x', '-'}, SIGNAL: wire, CONSTANT: Fixed signal  
 RTL: register transfer level: from combinational logic — connect with wires, Gates: AND NAND OR NOR NOT XOR XNOR, Sensitivity: signals on the right of <=: sensitivity list (should put all)

# VHDL:

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity entity_name is
  generic (K: unsigned;
           L: std_logic := '0'); -- default
  port (clock, reset: in std_logic; -- (n-1) downto 0
        Y: out std_logic_vector(0 to 7)); -- 0 to (n-1)
end entity_name;

architecture arch_name of entity_name is
  -- Declaration
  type state is (IDLE, UPDATE, DONE); -- local to arch
  -- hex "X"EF", bin: "11101111", macro: (4 => '0', others => '1')
  signal s_Y_data: std_logic_vector(7 downto 0) := "00000000"; -- default value
  signal s_A, s_B, s_C: std_logic;
  signal s_D, s_E: unsigned(5 downto 0);

  constant c_PI: std_logic := '1';

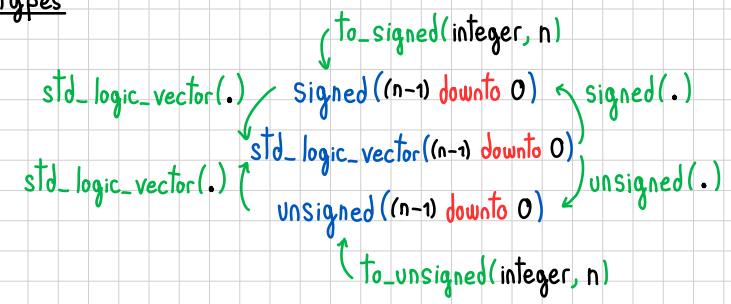
  component component_entity is
    generic (O: integer;
            P: std_logic_vector(2 downto 0));
    port (A, B: in std_logic;
          Y: out std_logic);
  end component;

```

## Help

Decimal	Bin	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

### Types



```

begin
  -- Description (don't read from output!)
  -- Implicit process
  Y <= (s_Y_data(0) or (s_Y_data(2) and (not s_Y_data(3)))) & s_Y_data(6 downto 0);
  s_B <= c_PI when s_Y_data(5)='1' else '-' when s_Y_data(6)=L else '0';
  with s_Y_data(4) select s_D <= -- puts value in s_D
    to_unsigned(0,6) when s_Y_data(0), -- tests data[4]=data[0]
    to_unsigned(1,6) when s_Y_data(2),
    to_unsigned(2,6) when others;
  -- Explicit process (latest assign matters/don't assign both in implicit and explicit)
  p_process_name: process(reset, clock, s_Y_data(1)) is -- Sensitivity list
  begin
    -- D-FlipFlop
    -- Async reset
    if reset='1' then s_A <= '0'; -- reset value
    elsif rising_edge(clock) then s_A <= s_B, -- falling_edge(clock)
    end if;
    -- Sync reset
    if rising_edge(clock) then
      if reset='1' then s_A <= '0';
      else s_A <= s_B xor s_A; -- Allowed read and write to signal
      -- (value not updated right away)
      end if;
    end if;

    case s_Y_data(1) is
      when s_Y_data(0) => -- can use if/else here
        s_E <= to_unsigned(0,6);
        s_A <= '0';
      when s_Y_data(2) => s_E <= to_unsigned(1,6);
      when others => s_E <= K;
    end case;
  end process p_process_name;

  pm_do: component_entity
    generic map (O => 9,
                P => "001");
    port map (A => s_A, B => s_B, Y => s_C);
  end arch_name;

```