DIGITAL LOGIC DESIGN

Trinh Quang Kien Ph.D.

kien.trinh@lqdtu.edu.vn# quangkien82@gmail.com

Course URL:

https://sites.google.com/view/trinhquangkien/Courses/dld

LECTURE 1.2 OUTLINE

- Hardware description language
  - HDL: VHDL and Verilog
  - A brief history of VHDL
  - Structure of a VHDL module
  - Basic data types and objects

Verilog is a portmanteau of the words "verification" and "logic"

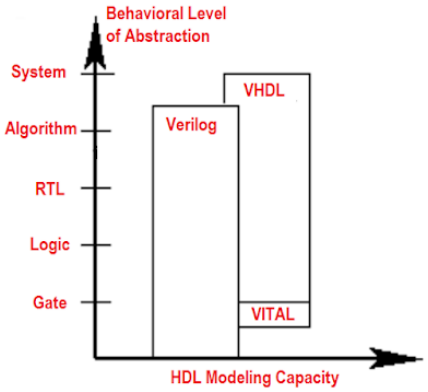
VHDL
Very High Speed Integrated Circuit Hardware Description Language

VHDL VS VERILOG

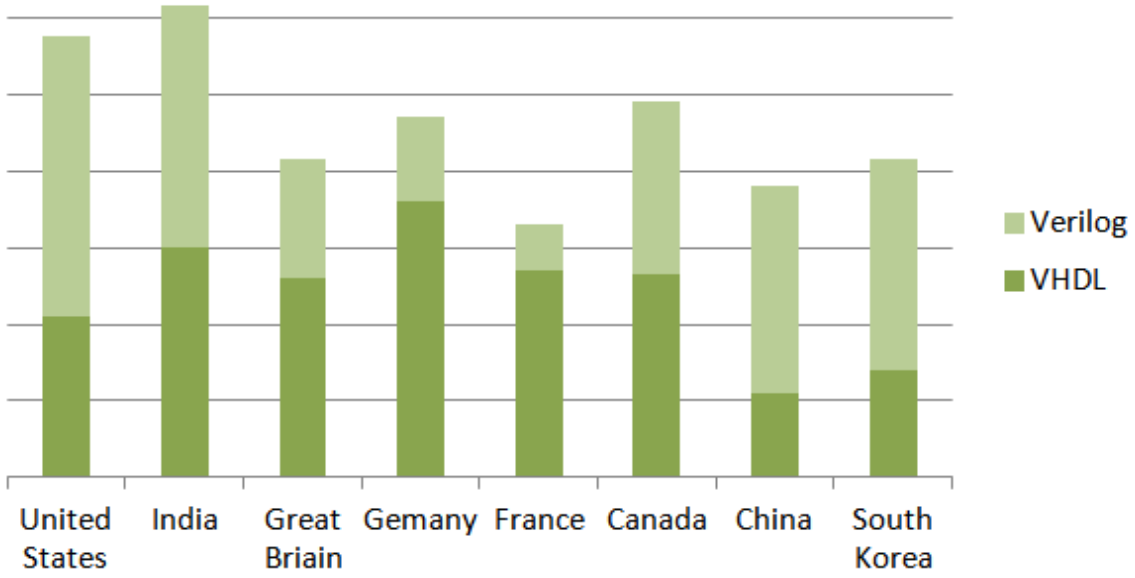
- Verilog is like C programming language, while VHDL is like Ada or Pascal programming language
- Verilog is case-sensitive while VHDL is not. It means that DAta1 and Data1 are two different signals in Verilog, but both are the same signals in VHDL.
- In Verilog, to use a component instance in a module, you just need to instantiate it in the module with a correct port map. In VHDL, before instantiating the instance, the component generally needs to be declared
- VHDL Complex data types vs Verilog simple data types (Verilog does not support user-defined data type)
- VHDL is strongly typed vs Verilog is loosely typed: signal and data assignment in VHDL need to be fully matched
- VHDL support Library management, package for reusing and common references, configuration

HDL and Verilog are equally capable languages. You should base your decision on which language to learn based on what's most popular for your location and circumstances

VHDL vs. Verilog Google Searches Worldwide: July 2013 to July 2014 *



HDL modeling capacity**



VHDL vs. Verilog Google Searches By Country: July 2013 to July 2014 *

*https://www.nandland.com/articles/vhdl-or-verilog-for-fpga-asic.html
*https://www.fpga4student.com/2017/08/verilog-vs-vhdl-explain-by-example.html

VHDL HISTORY

- **VHDL VHSIC HDL** (Very-High-Speed-Intergrated-Circuit Hardware Description Language), **VHDL borrows heavily from** Ada programming language in both concepts and syntax.
- **1981:** Initiated **by US Department of Defense (DoD)**
- **1983-1985:** Development of baseline language by Intermetrics (later AverStar), IBM and TI
- **1986:** All rights transferred to IEEE
- **1987:** Publication of IEEE Standard, the first official IEEE 1076-1987
- **1994:** Revised standard (named VHDL 1076-1993)
- **2000:** Revised standard (named VHDL 1076 2000, Edition, introduced signed, unsigned)
- **2002:** Revised standard (named VHDL 1076-2002)
- **2007:** VHDL Procedural Language Application Interface standard (VHDL 1076c-2007)
- **2009:** Revised Standard (named VHDL 1076-2008)

VERILOG HISTORY

- **1984**: Verilog HDL was invented by Phil Moorby and Prabhu Goel, and become a proprietary HDL owned by Gateway Design Automation Inc.
- **1990**, Gateway Design Automation Inc was acquired by Cadence Design System,.
- **1991** Verilog documents transferred (from Cadence)to public domain under the name of Open Verilog International (OVI).
- **1995**: and became IEEE standard 1364-1995, commonly referred as Verilog-95.
- **2001**, extensions to Verilog-95 were submitted back to IEEE and became IEEE standard 1364-2001, known as Verilog-2001.
- **2005**, Verilog-2005 (IEEE Standard 1364-2005) was published with minor corrections and modifications.
- **2005** System Verilog, a superset of Verilog-2005, with many new features and capabilities to aid design verification, was published.
- **2009**, SystemVerilog and Verilog language standards were merged into SystemVerilog 2009 (IEEE Standard 1800-2009), which is one of the most popular languages for IC design and verification.
- 2013 Xilinx® Vivado Design Suite, released in 2013, can support SystemVerilog for FPGA design and verification.

```
-------------------------------------------
-- simple adder example
-------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-------------------------------------------
entity adder is
generic (N: natural:=32);
port(
    Cin  : in  std_logic;
    A    : in  std_logic_vector(N-1 downto 0);
    B    : in  std_logic_vector(N-1 downto 0);
    SUM  : out std_logic_vector(N-1 downto 0);
    Cout : out std_logic
    );
end adder;
-------------------------------------------
architecture behavioral of adder32 is
signal A_temp, B_tem, Sum_temp :
std_logic_vector(N downto 0);
begin
    A_temp <= '0'& A;
    B_temp <= '0'& B;
    sum_temp <=  a_temp + b_temp + Cin;
    SUM  <= sum_temp(31 downto 0);
    Cout <= sum_temp(32);
end behavioral;
-------------------------------------------
```

VHDL
Very High Speed Integrated Circuit
Hardware Description Language

VHDL CHARACTERISTICS

- Popularity
  - Standardized
  - Supported by majority CAD Tools and IC companies
- Technology-independent
- Multi-level description
- Exchangeable, reusable

VHDL REFERENCES

- [1] Quang-Kien Trinh et al. Thiết Kế Logic số, MTA, 2011 PDF
- [2] IEEE, IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-2008, vol. 2008, no. January. 2009.
- [3] T. U. of Technology, “VHDL Coding Rules Guideline,” 2009.
- [4] Sarah L. Harris David Money Harris, Digital Design and Computer Architecture ARM Edition. 2016.
- [5] E. O. Hwang, “Microprocessor Design: Principles and Practices with VHDL,” p. 341, 2004.
- [6] D. Perry, VHDL Programming by Example, McGraw-Hill, 2002.
- [7] T. Zeugmann et al., VHDL Programming by Example. 2011
- [8] W. F. Lee, VHDL Coding and Logic Synthesis with Synopsys. 2000
- 5 examples according to digital logic design lecture: Lecture's Examples
- Complete set of basic VHDL design for beginner and self-studying: VHDL Examples
- VHDL&FPGA Labs

VHDL: STRUCTURE OF A VHDL DESIGN

- Library declaration
  - Library and package
- Entity declaration
  - Design generic parameters
  - Design interface (input, output)
- Architecture description
  - Design functionality
  - Types of architecture
    - Behavioral
    - Structure
    - Dataflow
- Configuration

```
-----------------------------------------------
-- simple adder example
-----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-----------------------------------------------
entity adder is
generic (N: natural:=32);
port(
    Cin  : in  std_logic;
    A    : in  std_logic_vector(N-1 downto 0);
    B    : in  std_logic_vector(N-1 downto 0);
    SUM  : out std_logic_vector(N-1 downto 0);
    Cout : out std_logic
    );
end adder;
-----------------------------------------------
architecture behavioral of adder32 is
signal A_temp, B_tem, Sum_temp : std_logic_vector(N
downto 0);
begin
    A_temp <= '0'& A;
    B_temp <= '0'& B;
    sum_temp <=  a_temp + b_temp + Cin;
    SUM  <= sum_temp(31 downto 0);
    Cout <= sum_temp(32);
end behavioral;
-----------------------------------------------
```

VHDL: VHDL ELEMENTS

Library

(IEEE, Std, user-defined)

Package in the library IEEE

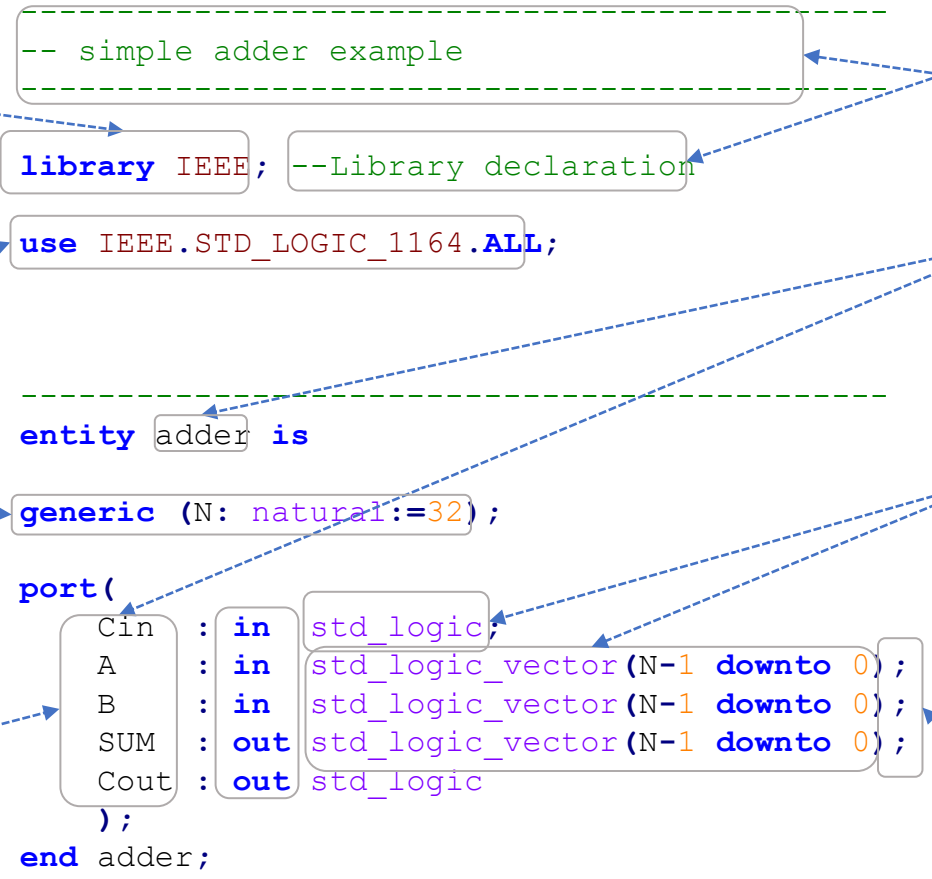
Generic parameter

N: input width,

32: the initial default value

Port modes

in,out,inout, buffer, linkage



the comments
Can be added in the middle

Entity name, signal name

(no space, @#$%..., start by a letter)

std_logic: single bit port

std_logic_vector: array of bit port

All declaration, statements end with ";"

VHDL does not differ capital and small letters !!

VHDL: DATA TYPES

- Scalar types
  - Logic (bit, std_logic)
  - Numeric (integer, real)
  - Enumerations (character, bit, Boolean)
  - Physical (time)
- Composite types
  - Array (same subtype components)
  - Record (arbitrary subtypes components)
- Data types can be predefined or user-defined

```
-- user-defined enumerations
type MULTI_LEVEL_LOGIC is (LOW, HIGH, RISING, FALLING,
AMBIGUOUS);
type BIT is ('0','1');
type SWITCH_LEVEL is ('0','1','X');
--user defined integer
type BYTE_LENGTH_INTEGER is range 0 to 255;
-- physical type
type DURATION is range -1E18 to 1E18
units
    fs; --femtosecond
    ps = 1000 fs; --picosecond
    ns = 1000 ps; --nanosecond
    us = 1000 ns; --microsecond
end units;
-- examples of an fully constrained array
type MY_WORD is array (0 to 31) of BIT;
-- examples of an fully constrained array
type MEMORY is array (INTEGER range <>) of MY_WORD;
--example of a record
type DATE is
record
    DAY : INTEGER range 1 to 31;
    MONTH : MONTH_NAME;
    YEAR : INTEGER range 0 to 4000;
end record;
```

VHDL: IEEE STD_LOGIC_1164

- logical values:
  - '1', 'H', '0', 'L'
- Metalogical values
  - 'U', 'X', 'W, '-'
  - Not physically exist
- High impedance
  - 'Z'

Name	Value	Strength
'U'	Unsolved	-
'X'	X	Strong
'0'	0	Strong
'1'	1	Strong
'Z'	High impedance	-
'W'	X	Weak
'L'	0	Weak
'H'	1	Weak
'-'	Don't care	-

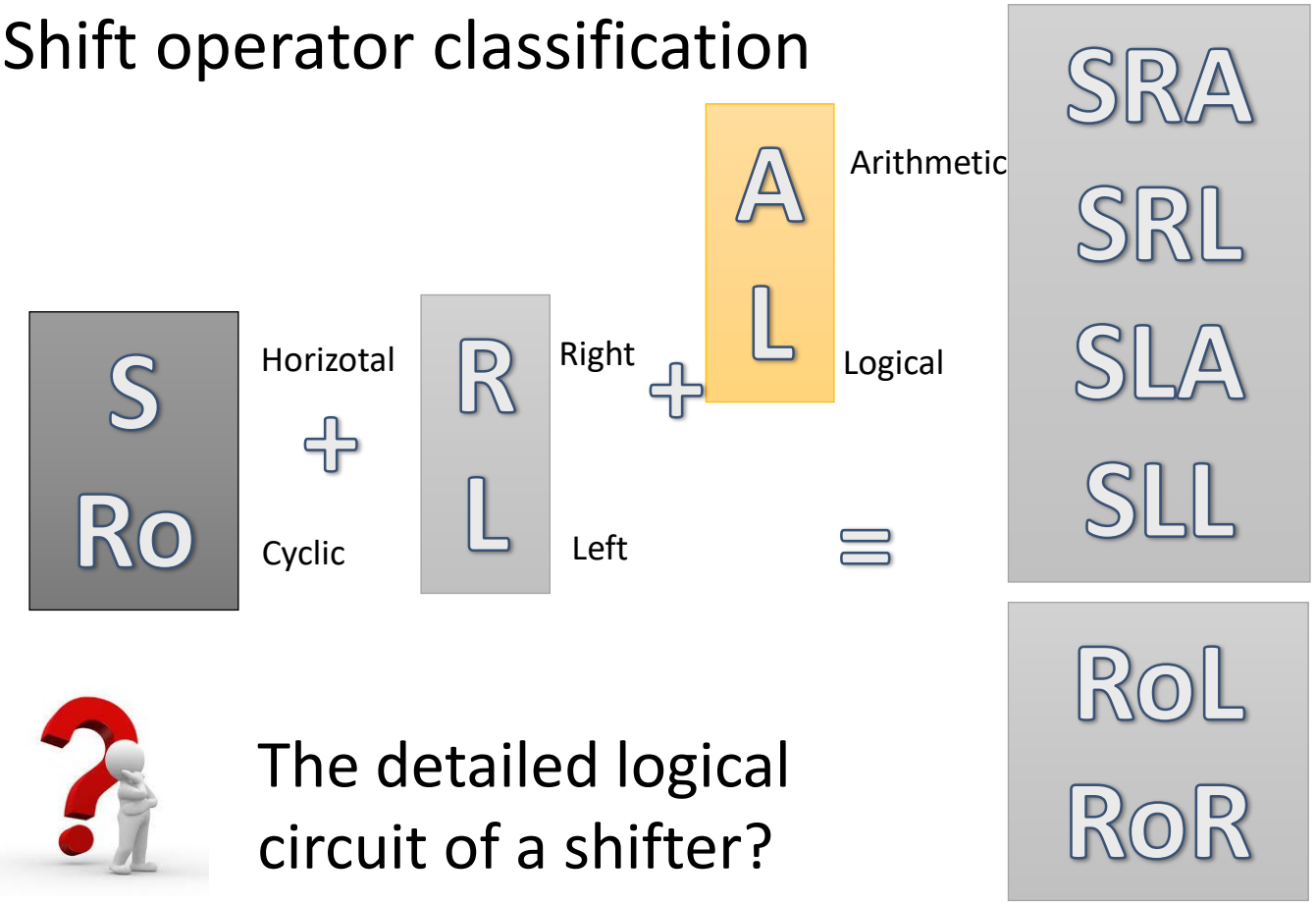
VHDL: OPERATORS

- operators
  - Conditional
  - Logical
  - Relational
  - Shift
  - Add, concatenation
  - Sign
  - Multiplying
  - Miscellaneous
- Logical and concatenation are enough to describe logic functions

Logical	**not, and, or, nand, nor, xor**
Relational	=, /=, <, <=, >, >=
Shift operator	sll, srl, sla, sra, rol, ror
Add, concatenation	+, -, &
Sign	+, -
Multiplying, division	*, /, mod, rem
Miscellaneous	**, abs

VHDL: OPERATORS

- Shift operator classification

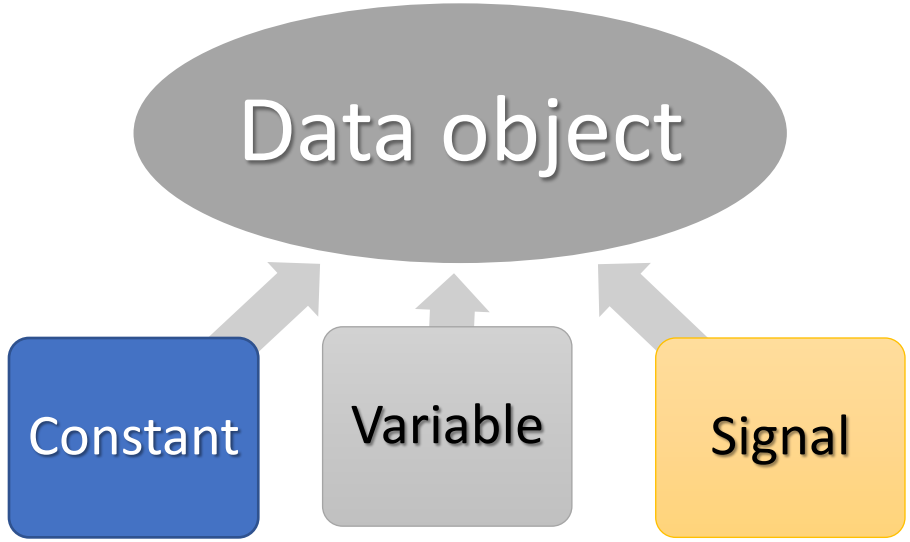


The detailed logical circuit of a shifter?

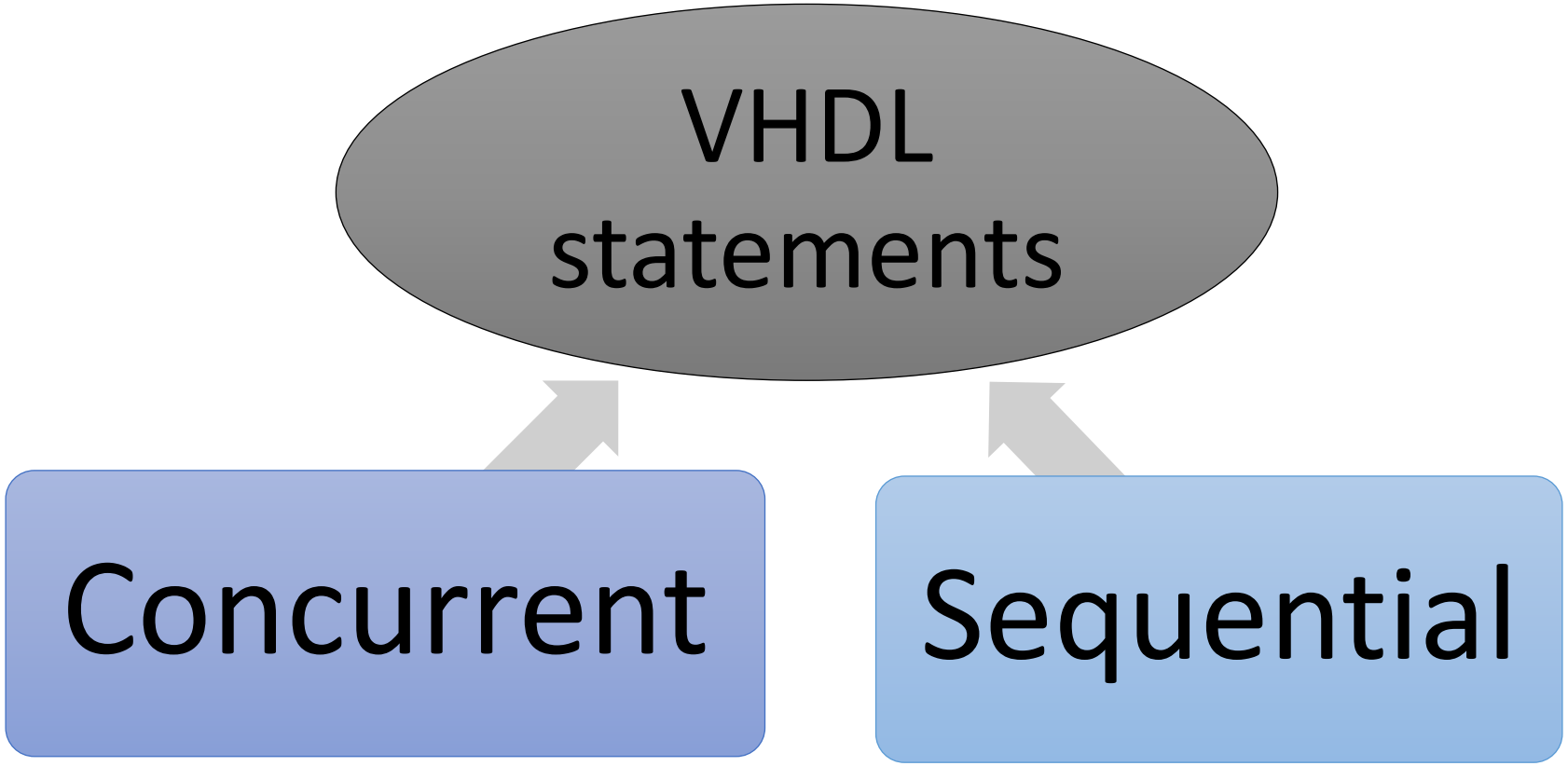
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.Numeric_STD.all;
----------------------------------------
----------------------------------------
Begin
   sh128i <= unsigned(shift_srl_in);
   shva   <= unsigned(shift_srl_value);
   sa     <= to_integer(shva);
   sh128o <= sh128i srl sa;
   shift_srl_out <= std_logic_vector(sh128o);
end shift_operator;
----------------------------------------
```

VHDL: DATA OBJECTS

- **Constant:** the data object for storing permanent value in a design. Constant is declared in package, entity, architecture, subprogram and process.
- **Variable:** the data object for storing intermediate values during a process or subprogram. Variable is declared in
- **Signal** the data object for connecting or synchronizing different blocks or process in the design



VHDL: STATEMENTS



VHDL: SEQUENTIAL STATEMENTS

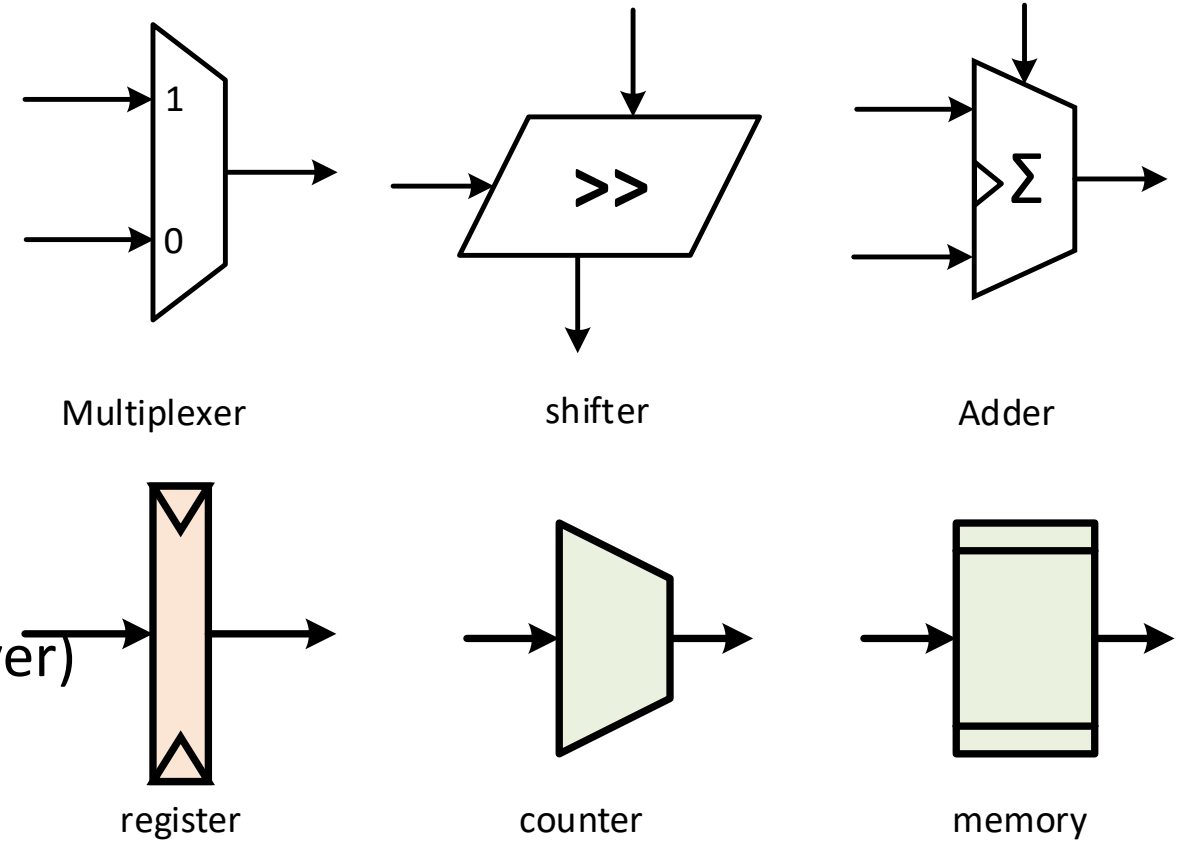
- Sequential statement is executed in the order.
- Can be used only in sub-program or process
- Can be used to described
  - Sequential logic (register, counter, memory, state machine)
  - Combinational logic (not recommended)
  - Design verification
- Branch (IF, CASE)
- Process control (WAIT, ASSERT, REPORT)
- LOOP
- Sequential signal assignments

VHDL: CONCURRENT STATEMENTS

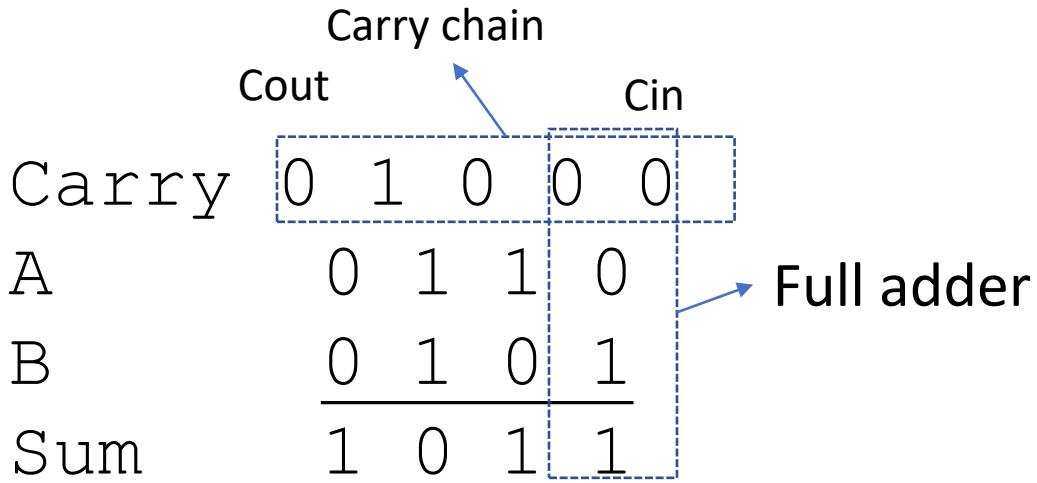
- Sequential statement is executed concurrently (or in parallel, not depended on the position of the statement)
- Must be used directly in the architecture body
- Can be used to described
  - Combinational logic (not recommended)
  - Structure design
- Process
- Component installation
- generate
- concurrent signal assignments

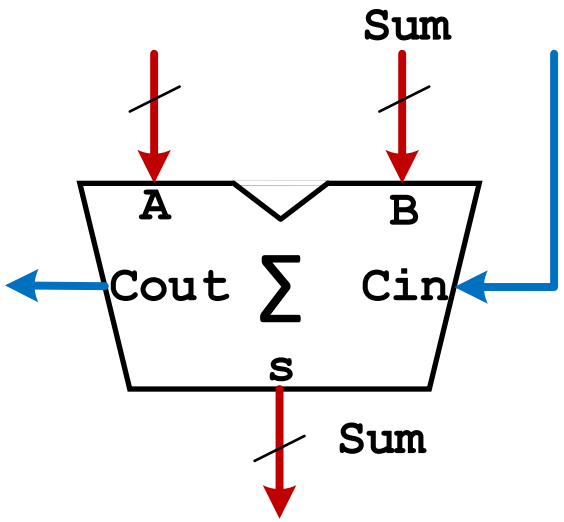
VHDL: PRACTICAL EXAMPLES

- Combinational logic
  - Simple adder design
  - Barrel shifter design
- Sequential logic
  - Register
  - counter
- Advanced examples
  - Finite state machine (UART receiver)
  - Loop example
  - FOR generate and IF generate
  - VHDL testbench



VHDL: EXAMPLE # 1



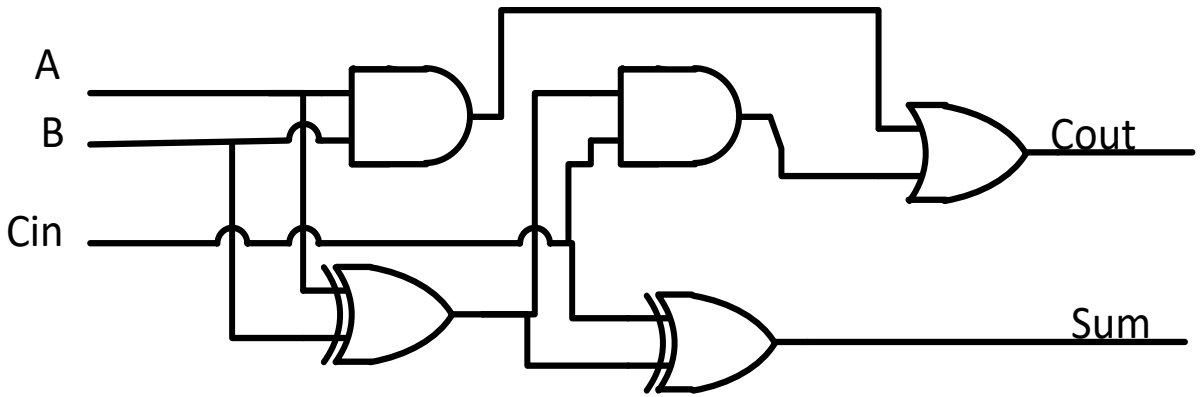


- Add operation
  - Naturally is a sequential operation : result of the next bit position (left) is dependent on the previous (right) position
  - The speed of carry chain defines the design latency
  - The 4-bit adder composes of 4 serial-connected full adders.

VHDL: EXAMPLE # 1

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

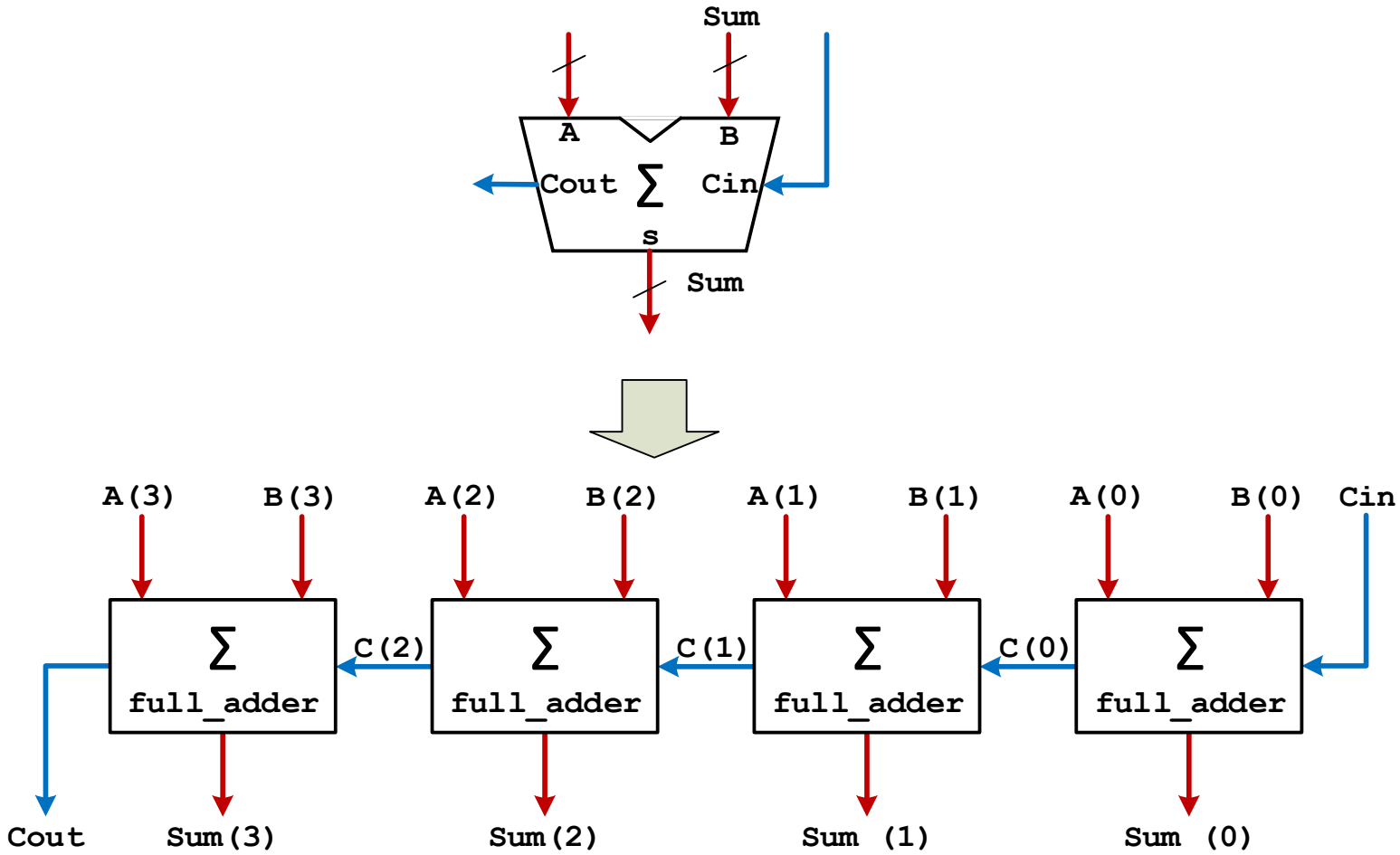
Full adder truth table




```
S    <=  A xor B xor Cin;
Cout <= (A and B) or (Cin and (a or b));
```

VHDL: EXAMPLE # 1

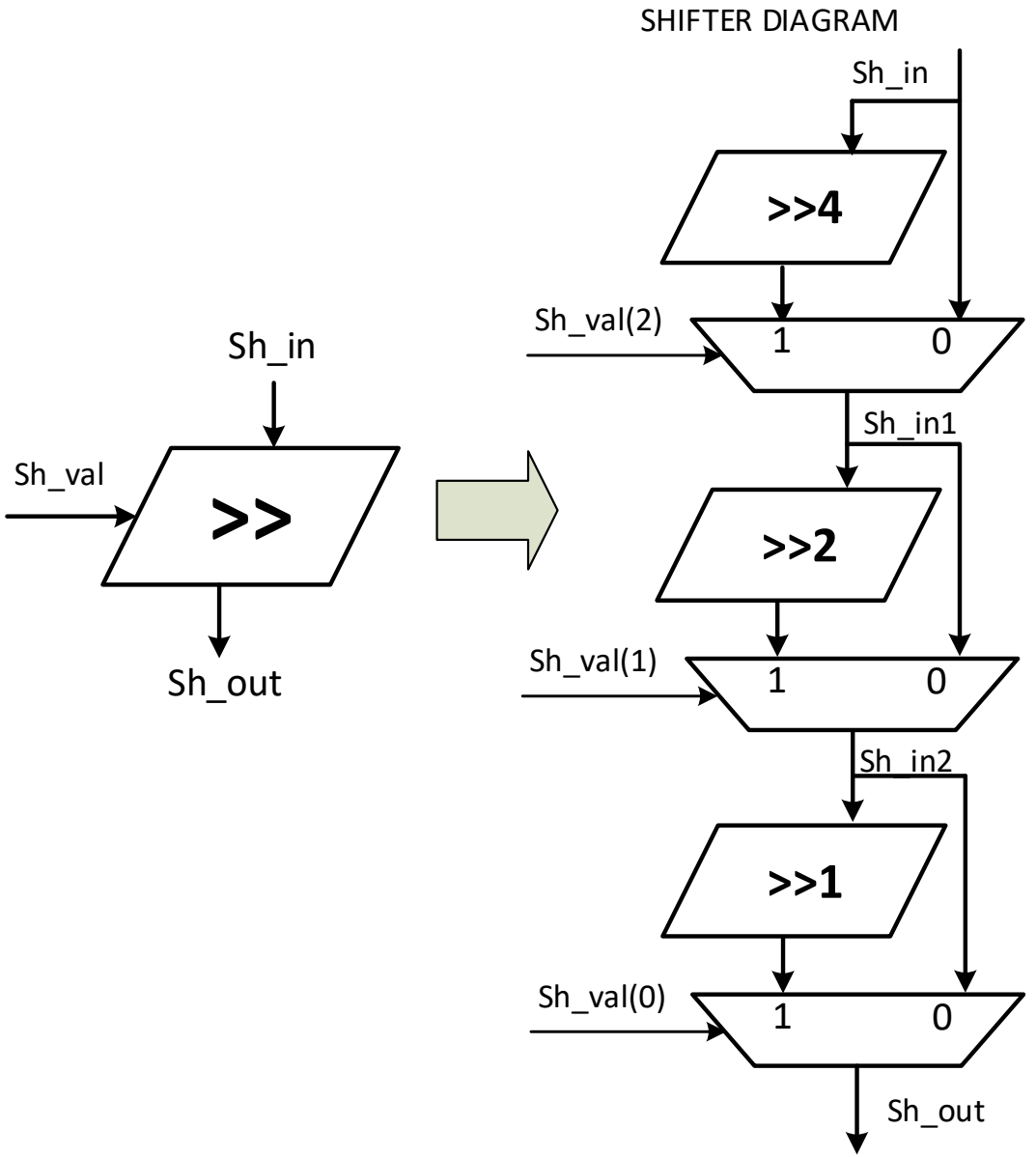
- 4-bit adder
  - Serial adder
  - Composes of 4 full-adders
  - Can be extended to N-bit adder



VHDL: EXAMPLE # 2

- 32-bit shifter
  - Using concatenation (&) for fixed amount of shifted bits
  - Variable shifted bit: need a smarter structure
    - Shifter by fixed amount using & operator
    - Multiplexer to select shifted or original value
    - Shift value is calculated as

$$SH_{val} = SH_{val}(2)2^2 + SH_{val}(1)2^1 + SH_{val}(0)2^0$$



EXAMPLE

Sh_in = 11011011 ,
Sh_val=101 (5)

Sh_val(2) = 1, SHIFT 4 bit

Sh_in1 = 00001101

Sh_val(2) = 0, No SHIFT

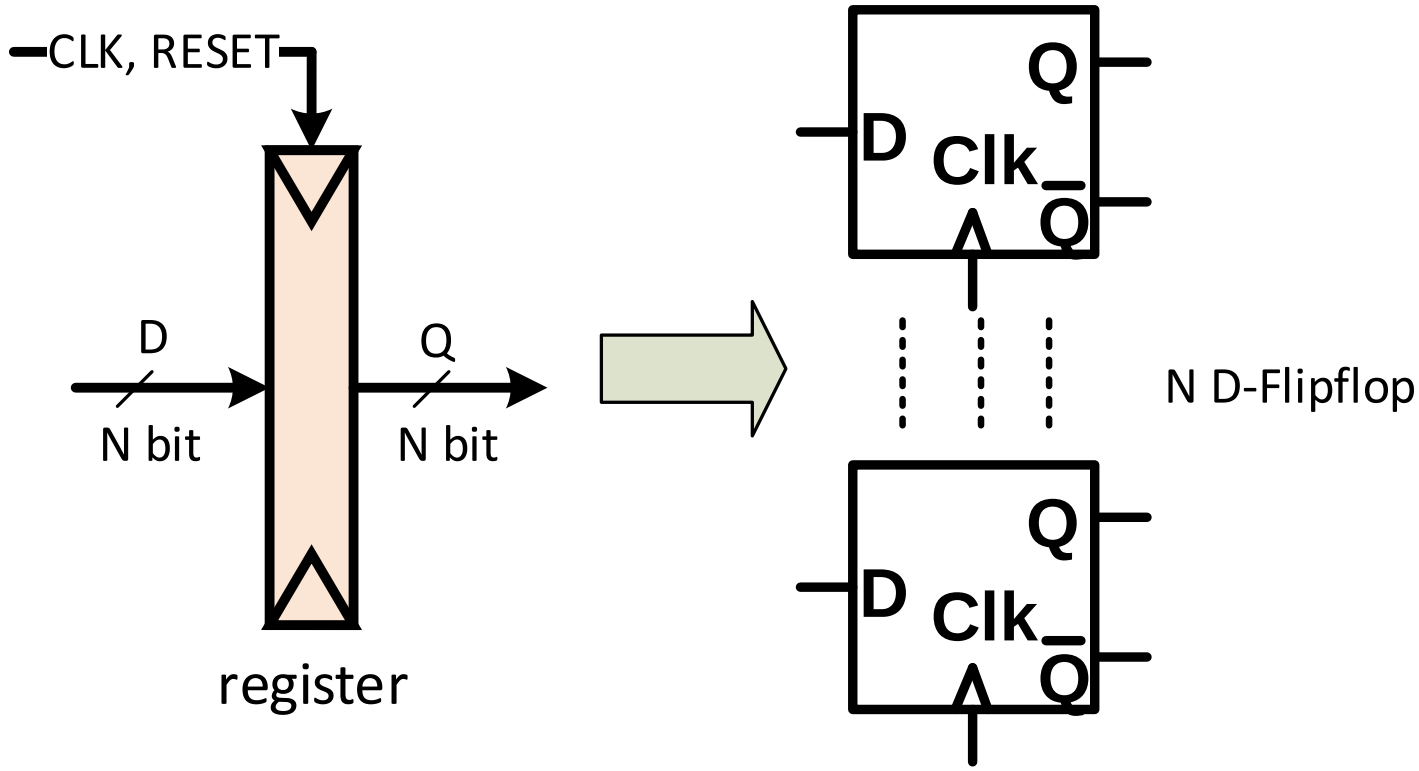
Sh_in1 = 00001101

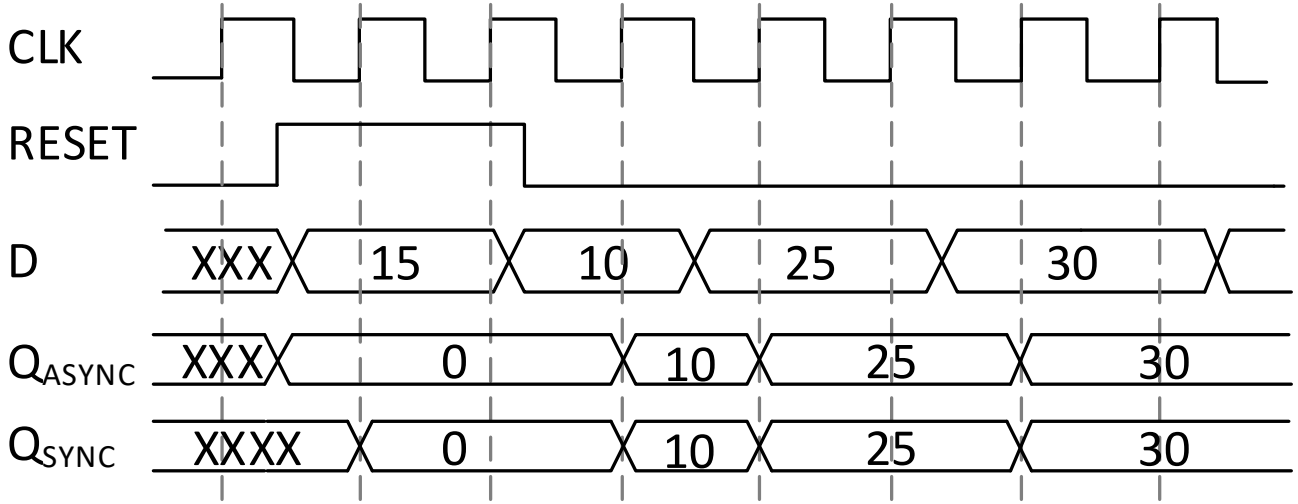
Sh_val(0) = 1, SHIFT 1 bit

Sh_out = 00000110

VHDL: EXAMPLE # 3

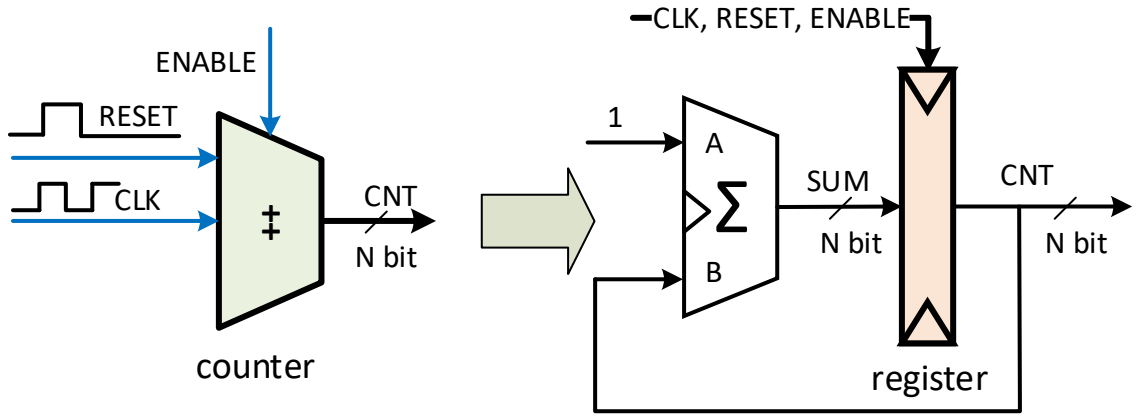
- 32-register
  - Fundamental sequential building block
  - Asynchronous reset vs synchronous
  - Equivalent N D-FF connected in parallel

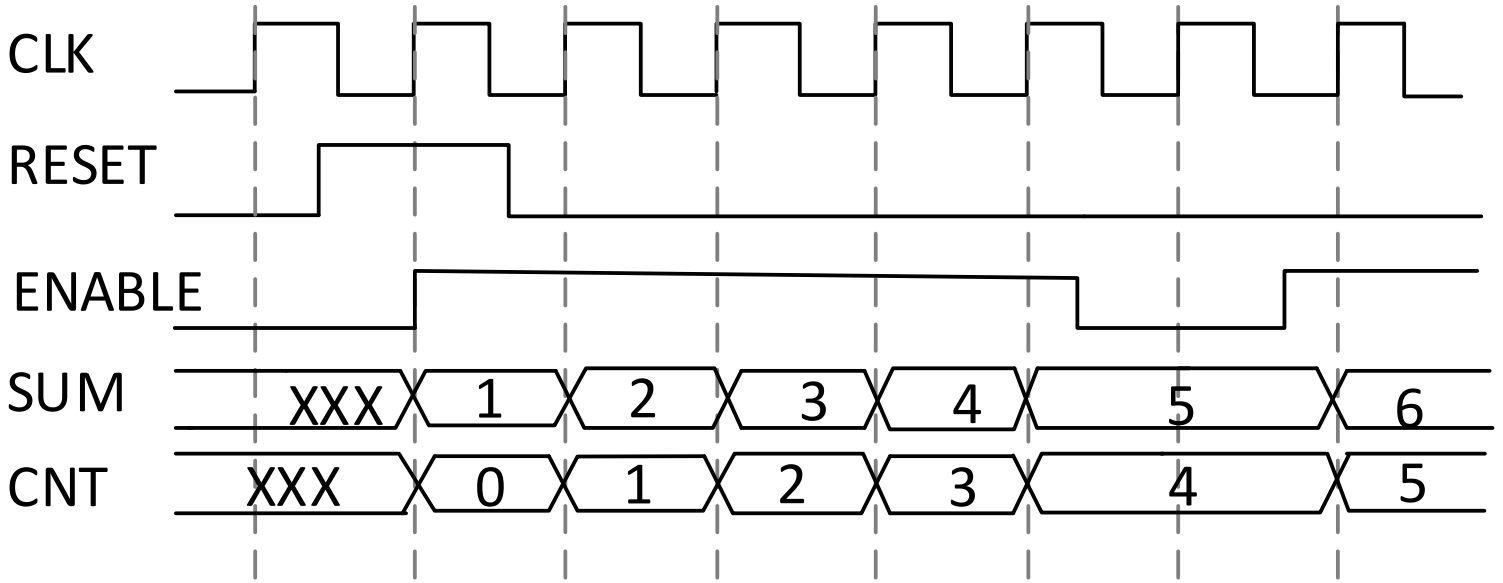




VHDL: EXAMPLE # 4

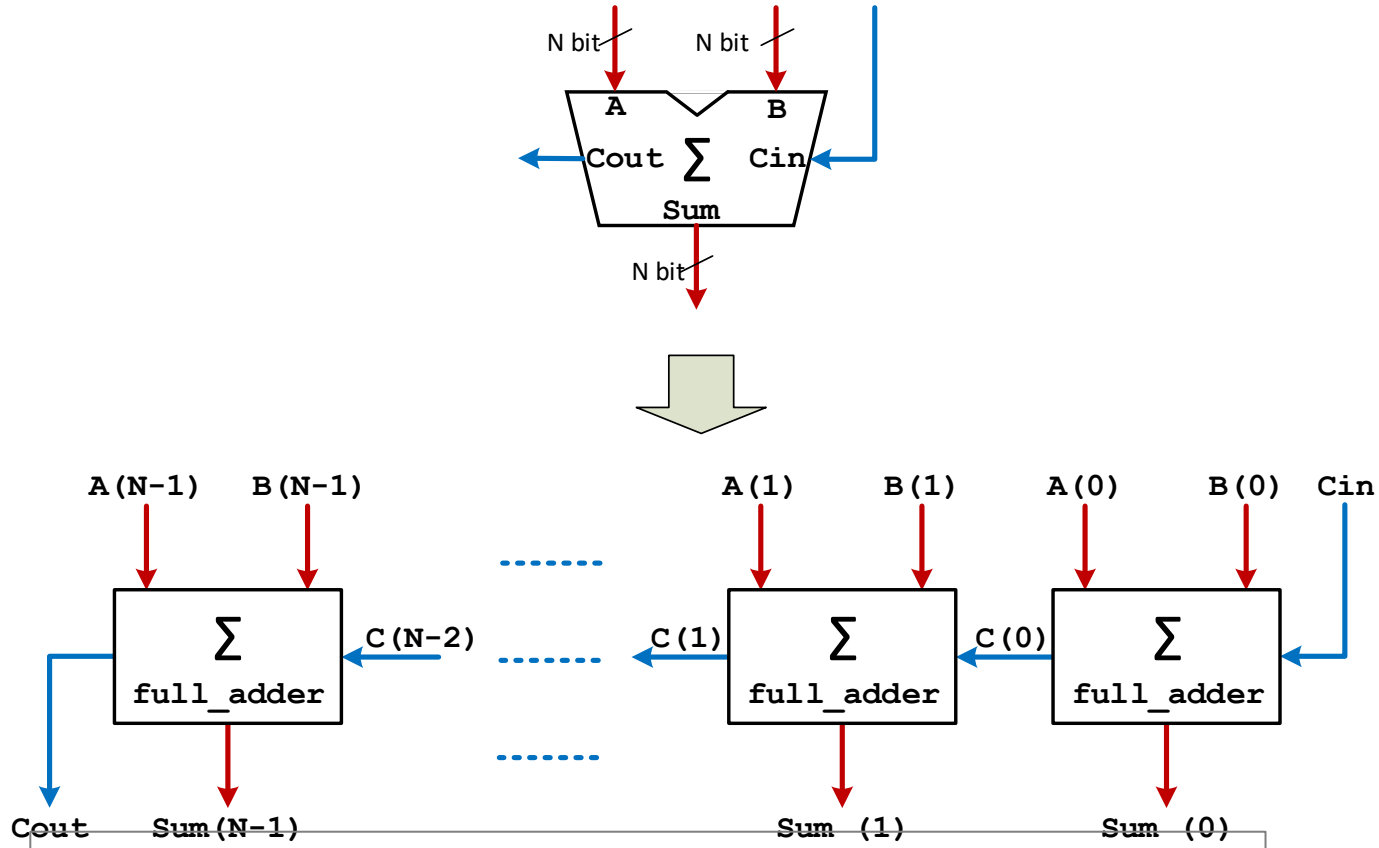
- Binary Counter
  - Adder (incrementor) + register
  - CLK+ENABLE: Counter pulse





VHDL: EXAMPLE # 5

- N-bit Adder
  - Using operator “+”
  - Using for-generate to instantiate N-full_adder components



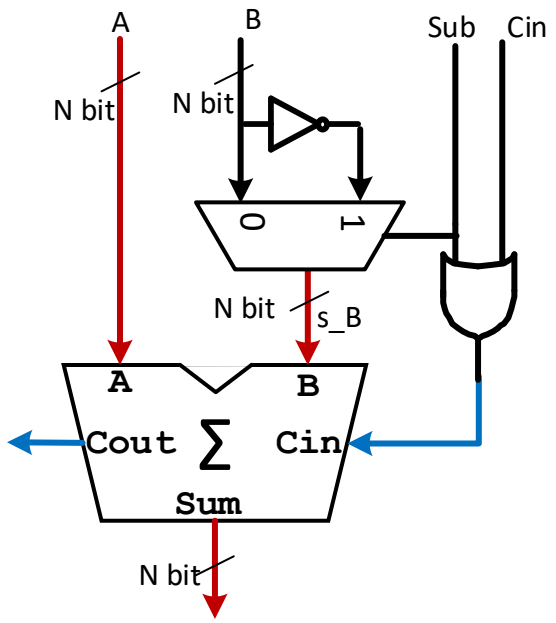

```
A1   <= '0' & A;
B1   <= '0' & B;
Sum1 <= A1 + B1 + Cin;
Cout <= Sum1(N);
Sum  <= Sum1(N-1 downto 0);
```

```
SumGen: for i in 0 to N-1 generate
    SUM(i) <=  A(i) xor B(i) xor C(i);
end generate;
```

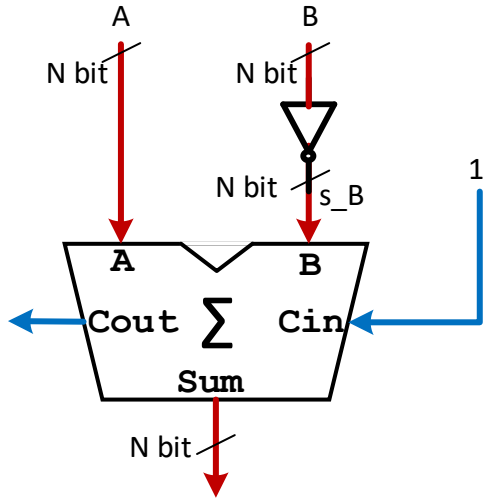
VHDL: EXAMPLE # 6

- Adder or subtractor
  - Can be either subtractor or adder
  - Some component always performs add or sub only -> more optimized
- Add/sub
  - Universal adder: can perform both adding and subtracting with minimum modification of the original adder

A-B = A + 2'complement(B) = A+ not B +1

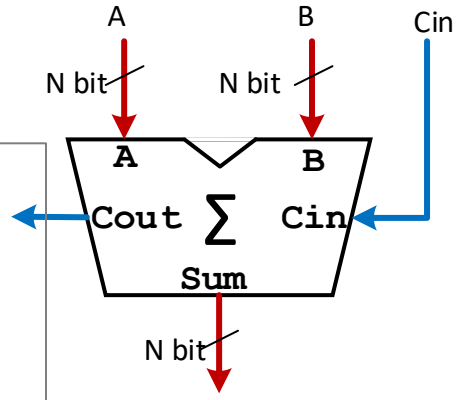


ADDER/SUBTRACTOR



SUBTRACTOR

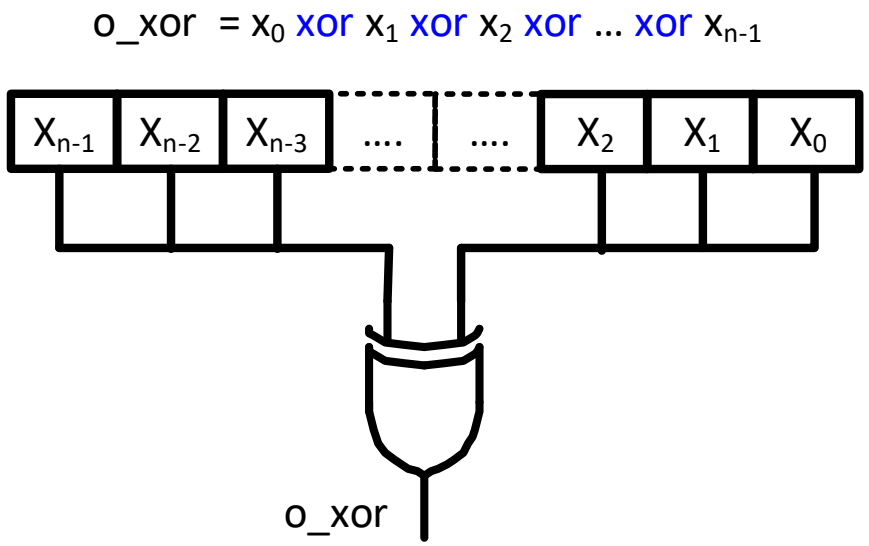
```
B_1: if SUB = TRUE generate
   s_B   <= not B;
   s_Cin <= '1';
end generate;
B_not: if SUB = FALSE generate
   s_B   <= B;
   s_Cin <= Cin;
end generate;
```

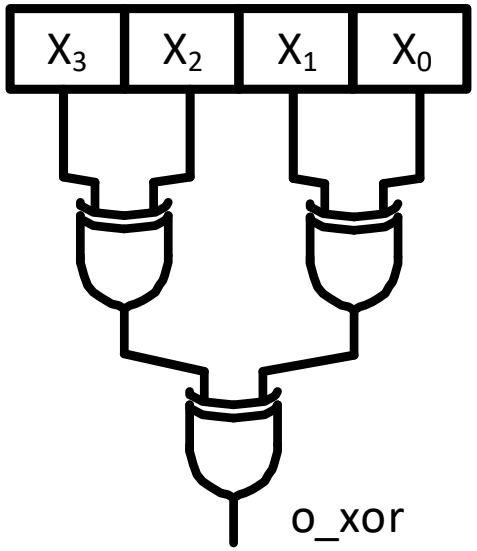


ADDER

VHDL: EXAMPLE # 7

- Reduction XOR function
  - Perform XOR function on all bits of a vector
  - Using generate or loop
  - Variable is used for storing intermediate value only






```
loop_p: process (i_vector)
variable i: integer := K-1;
variable p: std_logic :='1';
begin
   p := i_vector (0);
   i := 1;
   for i in 1 to  K-1 loop
       p := p xor i_vector(i);
   end loop;
    o_xor <= p;
end process;
```

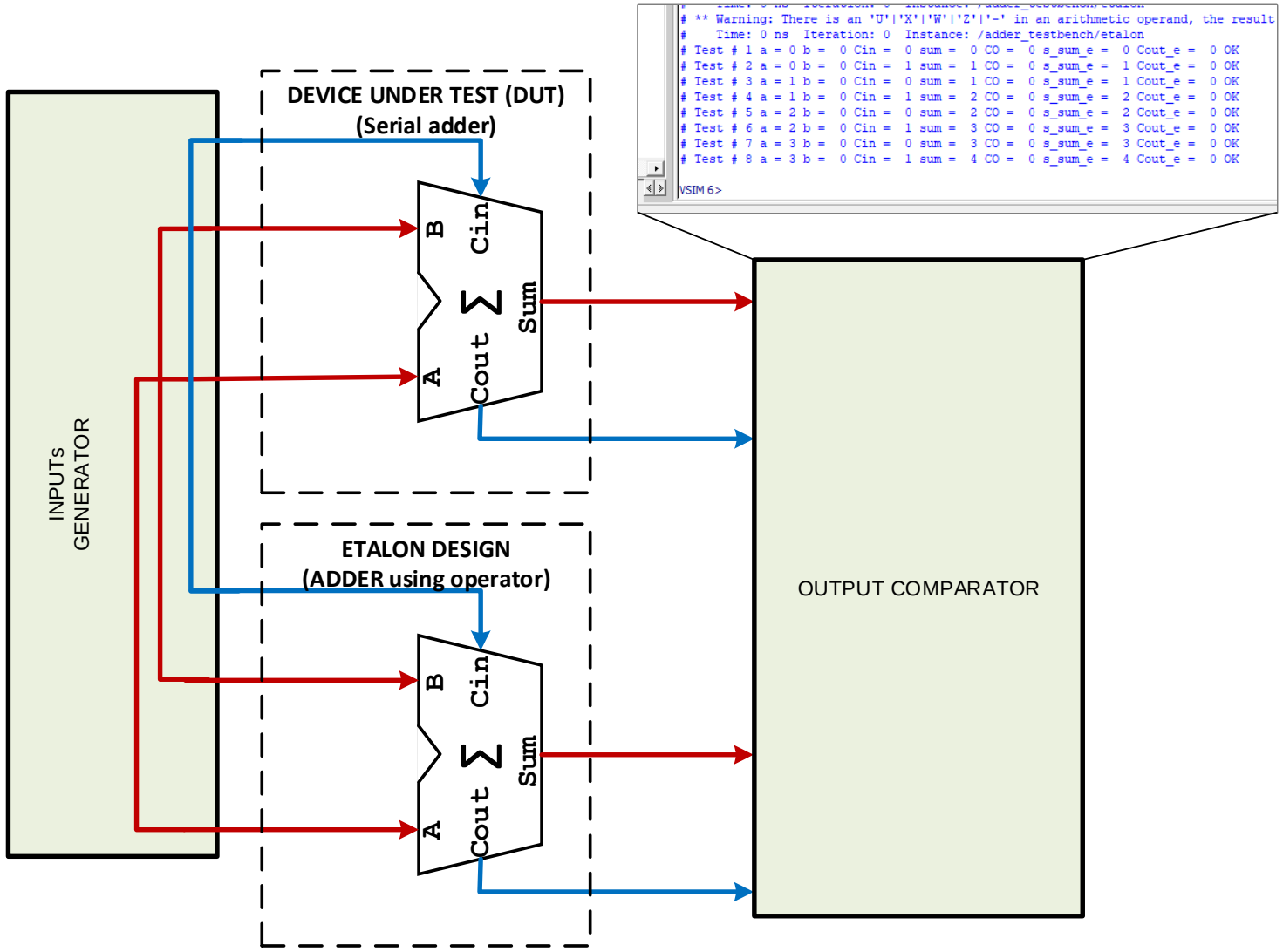
Variable declaration

Sequential statements

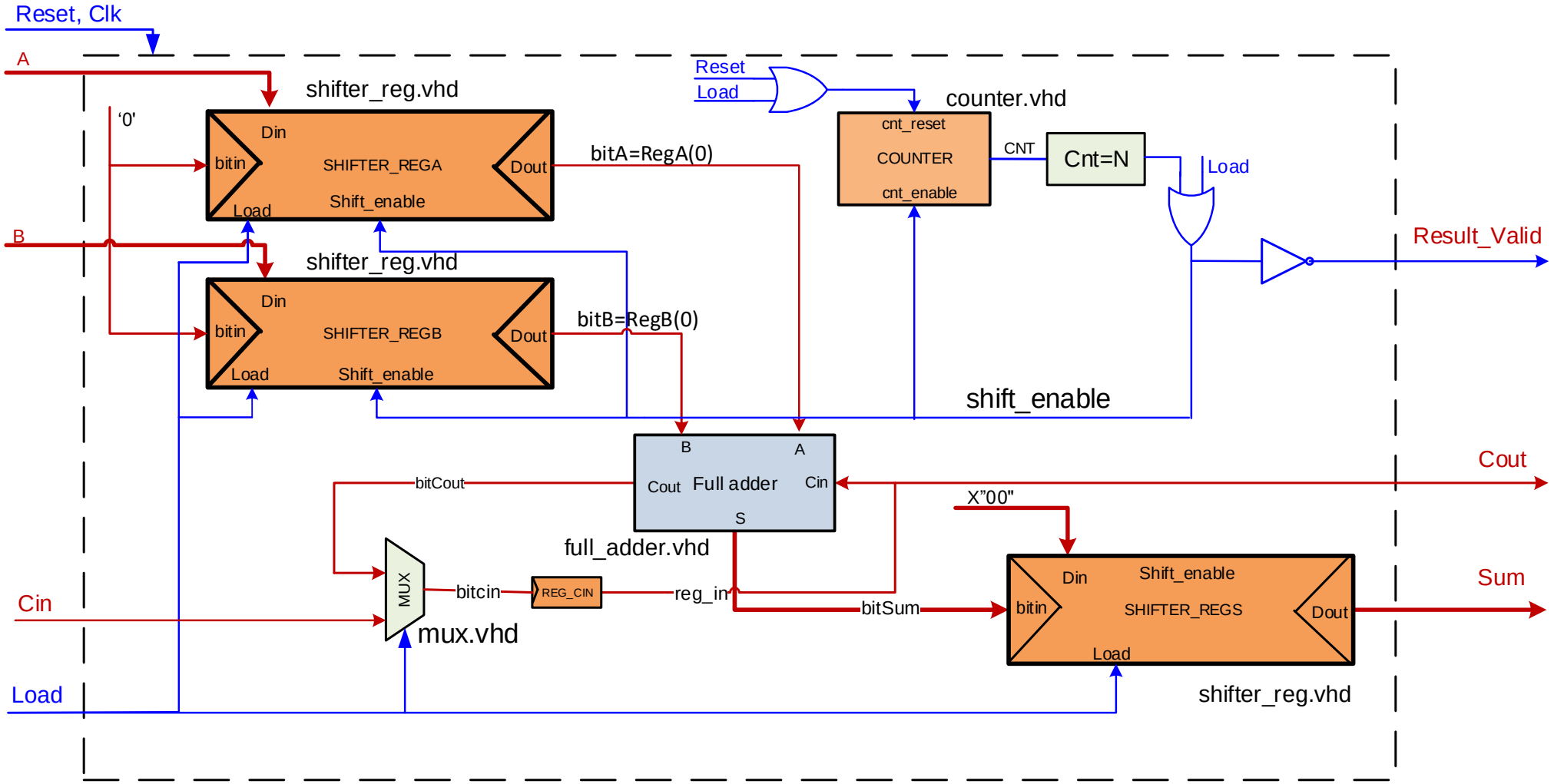
Store the calculated variable to signal/output

VHDL: EXAMPLE # 9

- TESTBENCH
  - Automatically check for large number of the inputs
  - Results is systematically reported in as a text-based file rather in a waveform

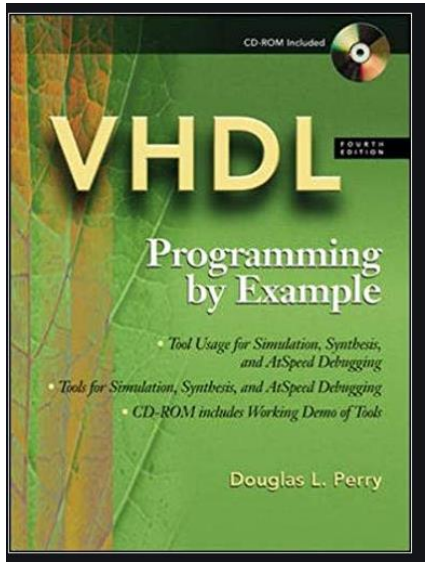


VHDL: EXAMPLE # 8 SERIAL-BIT ADDER



VHDL FURTHER READING TOPIC

- T. U. of Technology, "VHDL Coding Rules Guideline," 2009.
- D. Perry, VHDL Programming by Example, McGraw-Hill, 2002.
- VHDL vs Verilog link



VHDL Coding Rules

Tampere University of Technology
Department of Computer Systems
Version 4.4 – Jan 2009

VHDL: RULES # 1

- Included file header
- VHDL file
  - Name of file and entity are the same (**adder** and **adder.vhd**)
  - One file per one entity
- Entity /port
  - Use only **in, out** modes for ports
  - Use suffixed _in, _out unless for common designs
- Every entity has one testbench

adder.vhd

```
----------------------------------------------
-- simple adder example
----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
----------------------------------------------
entity adder is
generic (N: natural:=32);
port(
    Cin  : in  std_logic;
    A    : in  std_logic_vector(N-1 downto 0);
    B    : in  std_logic_vector(N-1 downto 0);
    SUM  : out std_logic_vector(N-1 downto 0);
    Cout : out std_logic
    );
end adder;
----------------------------------------------
architecture behavioral of adder32 is
signal A_temp, B_tem, Sum_temp : std_logic_vector(N
downto 0);
begin
    A_temp <= '0'& A;
    B_temp <= '0'& B;
    sum_temp <=  a_temp + b_temp + Cin;
    SUM  <= sum_temp(31 downto 0);
    Cout <= sum_temp(32);
end behavioral;
----------------------------------------------
```