

Partitioning-based System Design Methodologies



Alfredo BENSO
Politecnico di Torino (Italy)
UCSD, CA

Alfredo.benso@polito.it
www.testgroup.polito.it

Goal

- This lecture presents some Partitioning-based System Design Methodologies.

Basic design strategy

Whenever the system to be implemented is complex enough not to be easily implemented by a single unit (i.e., a process), it's convenient to partition it in *functional units*.

11.4

3

Side effects

Partitioning the system in several units improves design re-usability.

11.4

4

Partitioning hints

It's usually convenient to introduce, at least:

- an *Input Unit*, charged of communicating with the external world and of transforming input signals in an internal “*suitable*” form
- an *Output Unit*, charged of transforming internal signals in the form required by the external world
- a *Main body*, charged of performing the target operations.

11.4

5

Note

Partitioning is a typical example of the top-down design methodology, through a *hierarchical decomposition*, early introduced in lecture 2.1

11.4

6

***Example of
hierarchical
decomposition***

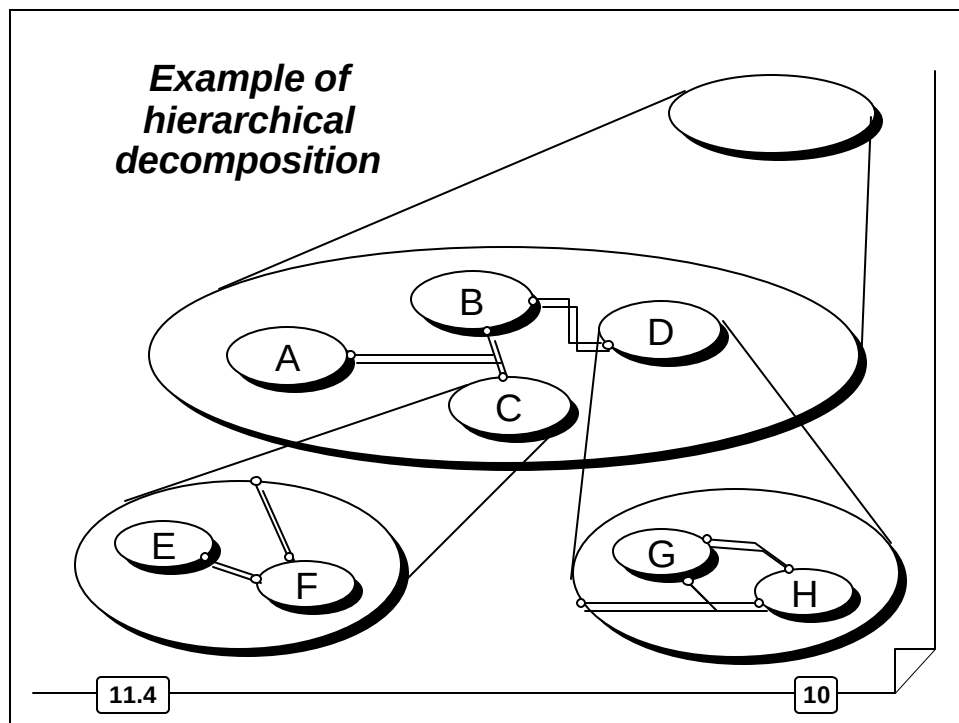
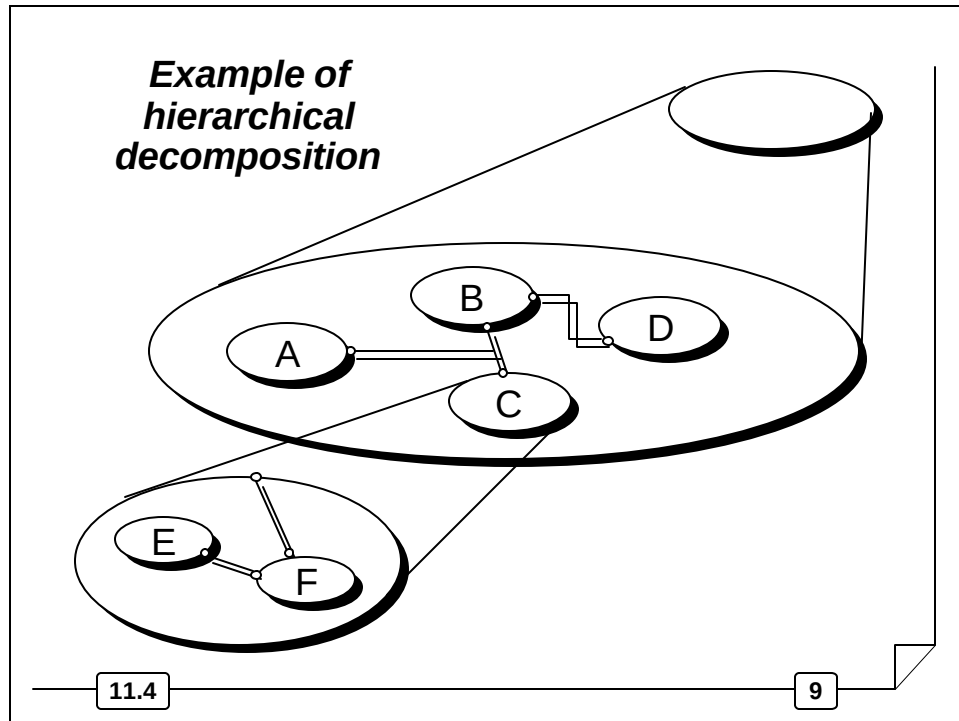
11.4

7

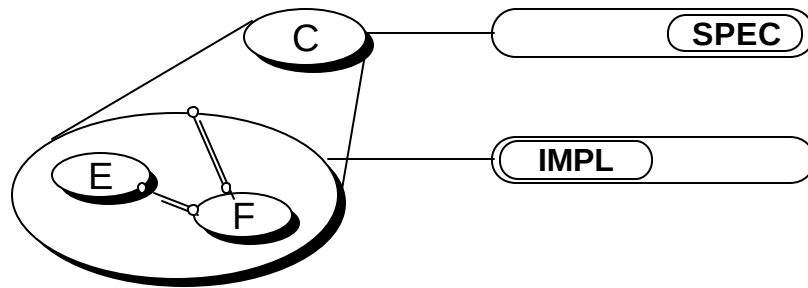
***Example of
hierarchical
decomposition***

11.4

8



Link with the hierarchy



11.4

11

Stopping condition

Partitioning stops when the “leaf” sub-systems are simpler enough to be “easily” implemented.

11.4

12

Basic assumptions

In the following, unless differently stated explicitly, we shall assume that:

- all the units run concurrently, clocked by a same master clock signal
- all the memory elements of all the units are clocked by the same clock edge (e.g., the rising one).

11.4

13

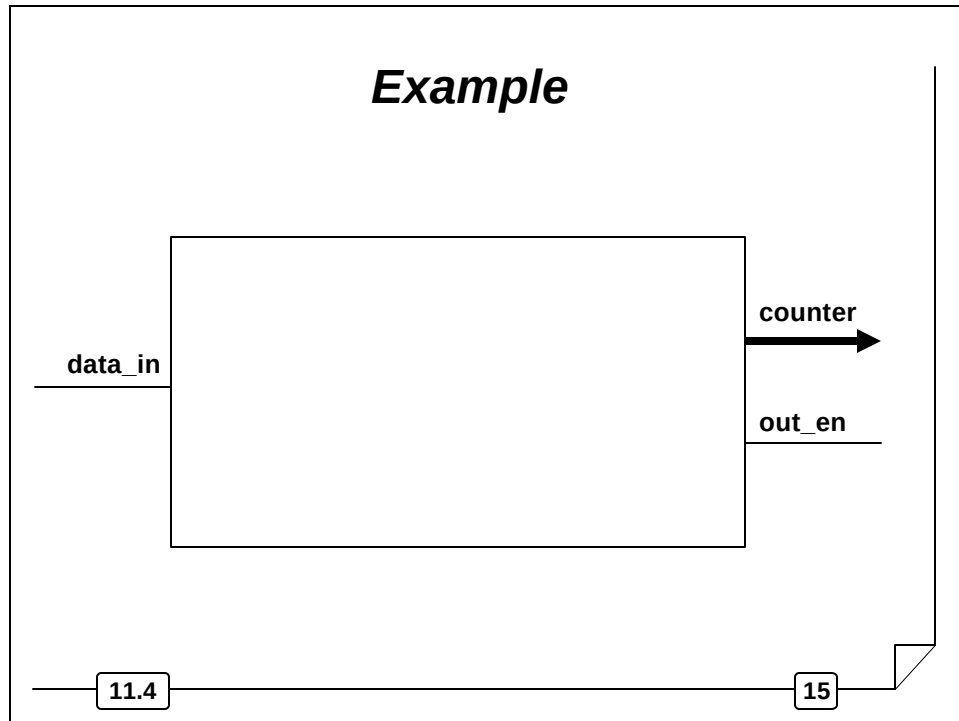
Design steps

1

Identify the Input / Output signals of the target system, in terms of their names, type and width.

11.4

14



***Elements definition:
I/O***

<i>Label</i>	<i>Type</i>		<i>Range</i>
clk	in	std_logic	single bit
reset	in	std_logic	single bit
data_in	in	std_logic	single bit
out_en	out	std_logic	single bit
counter	out	integer	0 to 15

Elements definition:

To improve re-usability, never use numbers; always resort to constants.

<i>Label</i>	<i>Type</i>		<i>Range</i>
clk	in	std_logic	single bit
reset	in	std_logic	single bit
data_in	in	std_logic	single bit
out_en	out	std_logic	single bit
counter	out	integer	0 to 15

***Elements definition:
constants***

<i>Label</i>	<i>Type</i>		<i>Range</i>
OUT_WIDTH	constant	integer	16

Elements definition: I/O

<i>Label</i>	<i>Type</i>		<i>Range</i>
clk	in	std_logic	single bit
reset	in	std_logic	single bit
data_in	in	std_logic	single bit
out_en	out	std_logic	single bit
counter	out	integer	0 to OUT_WIDTH-1

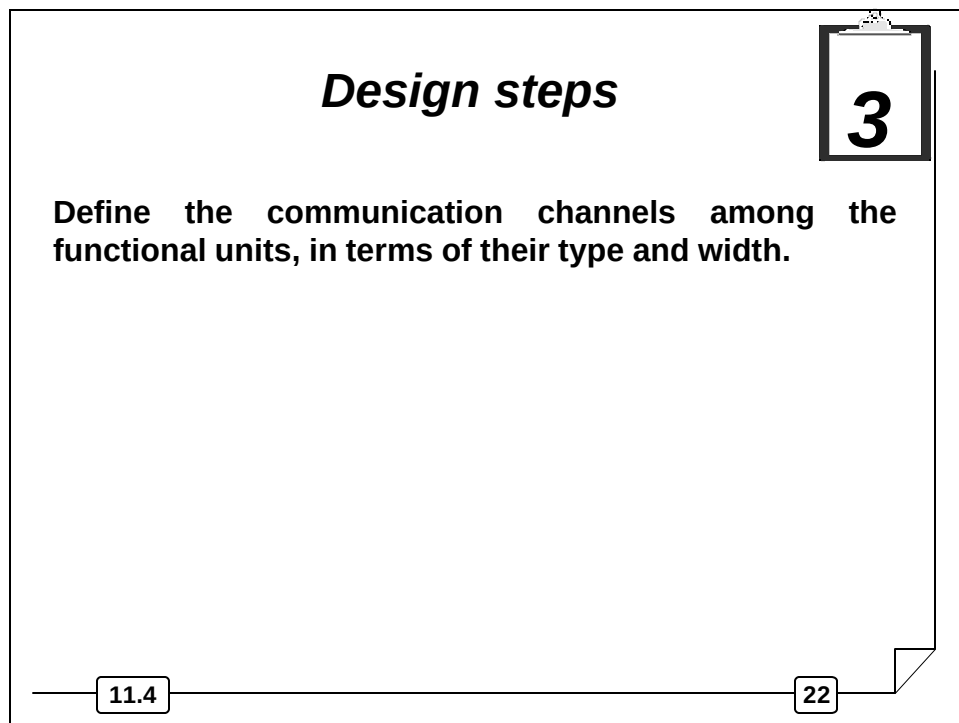
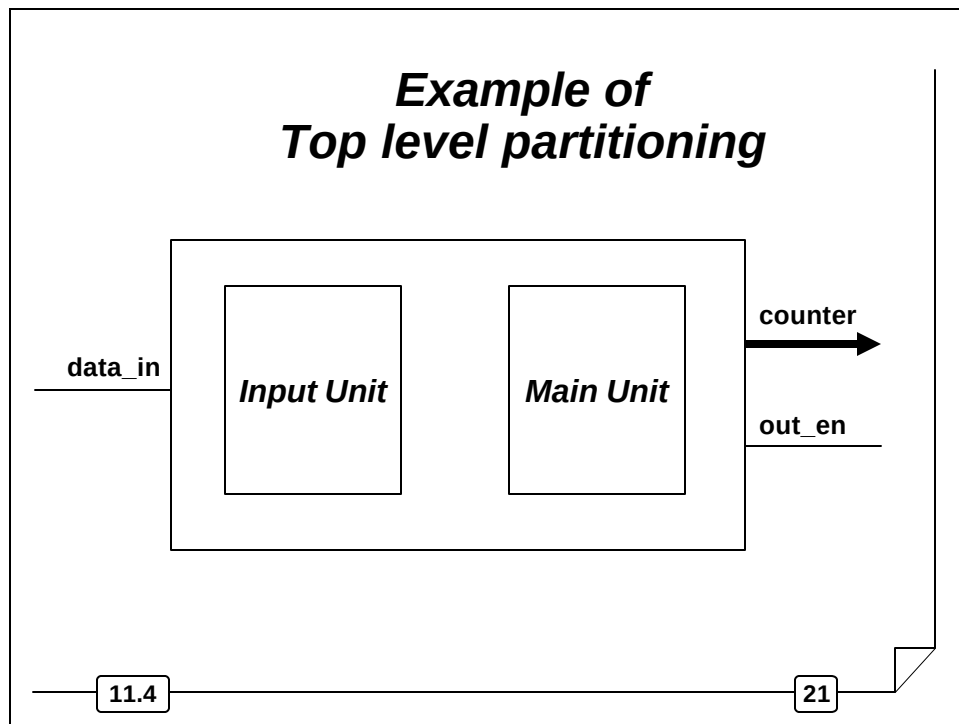
Design steps

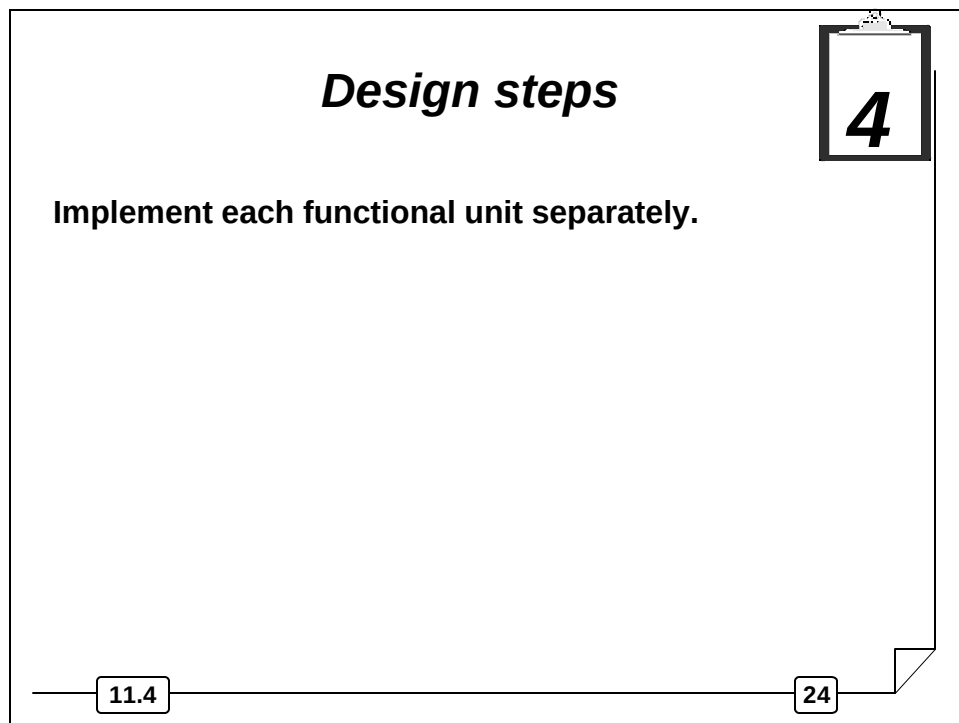
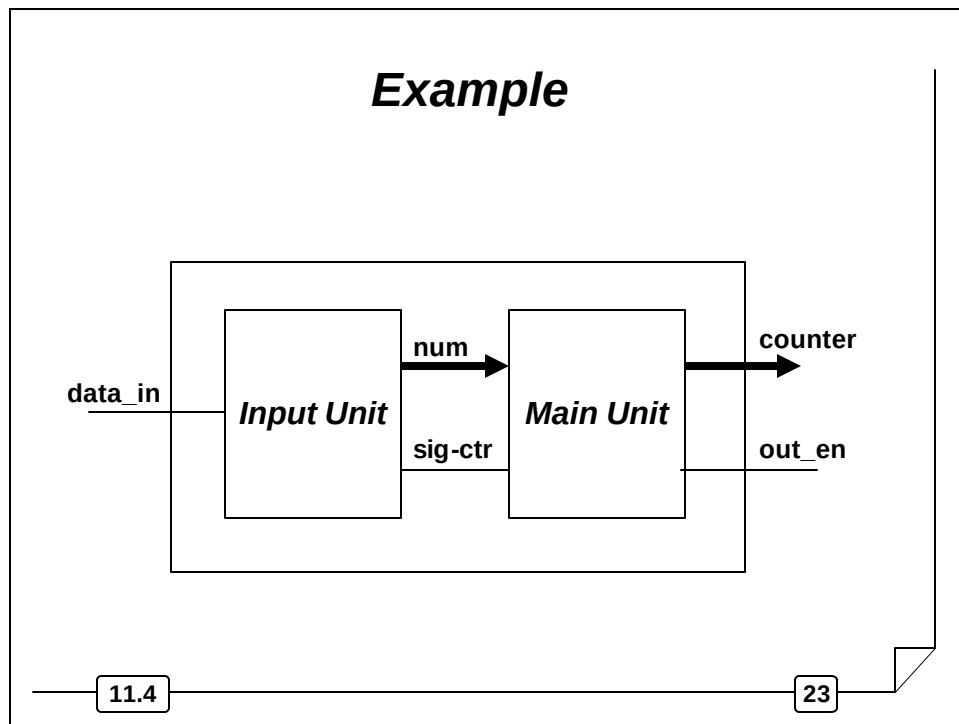
2

Identify the functional units the system has to be partitioned in (*Top-level partitioning*)

11.4

20





Unit implementation

Describe the behavior of each unit as a process.

Two cases can usually occur:

- The unit is rather “*simple*”, i.e., its behavior can be described as a straight piece of behavioral VHDL, without resorting to the introduction of “states”
- The unit is more “*complex*”, i.e., its behavior can be described only by resorting to the introduction of “states” the process evolves through.

11.4

25

An example

A ROM memory (1K x 16) stores 1024 data of 16 bits each, represented according to the “2’s complement” notation.

On a serial transmission line, 16-bits numbers are transmitted synchronously w.r.t. a clock signal clk, one bit per clock cycle, according to the following protocol:

- Numbers are transmitted LSB first
- Each new number has a 4 bit header : “1111”
- 6,000+ clock cycles occur between two consecutive transmitted numbers.

11.4

26

An example *(cont'd)*

A system is to be designed such that, whenever a new number N is received via the input line, determines the overall number T of data D, stored in the ROM, are such that

$$|D| = N.$$

At the operation completion, T has to be visualized, on a proper set of 7-segment displays, till getting the next number N.

11.4

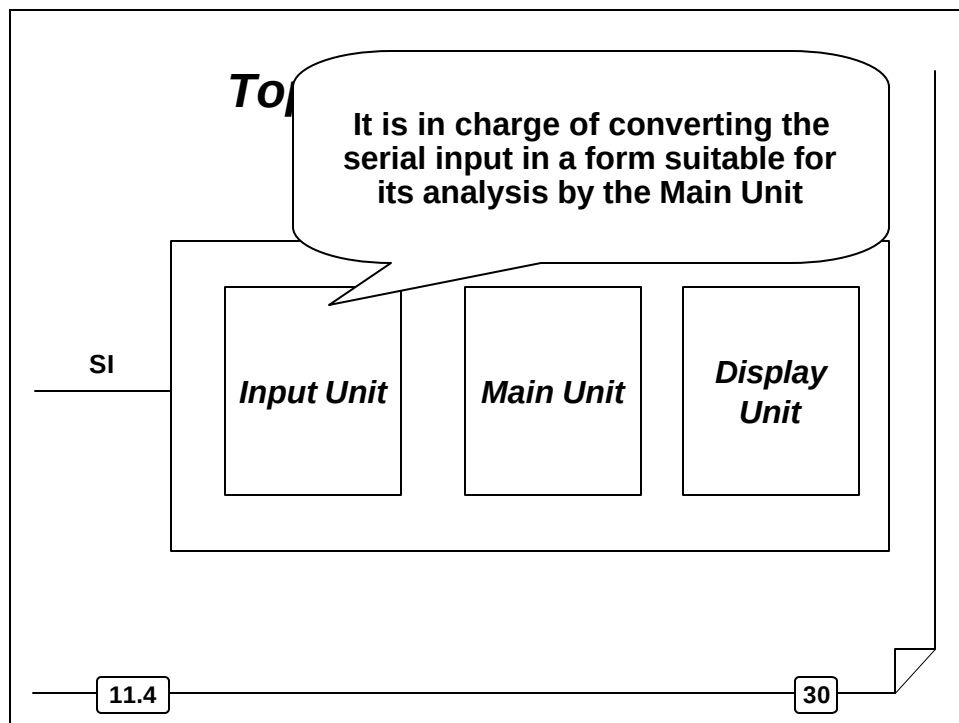
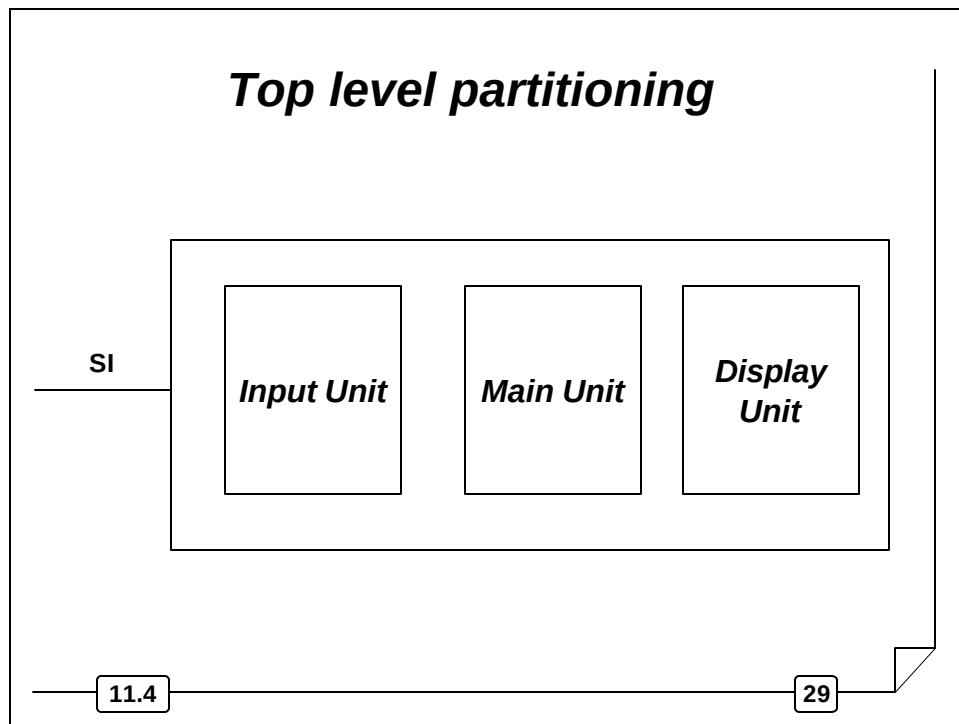
27

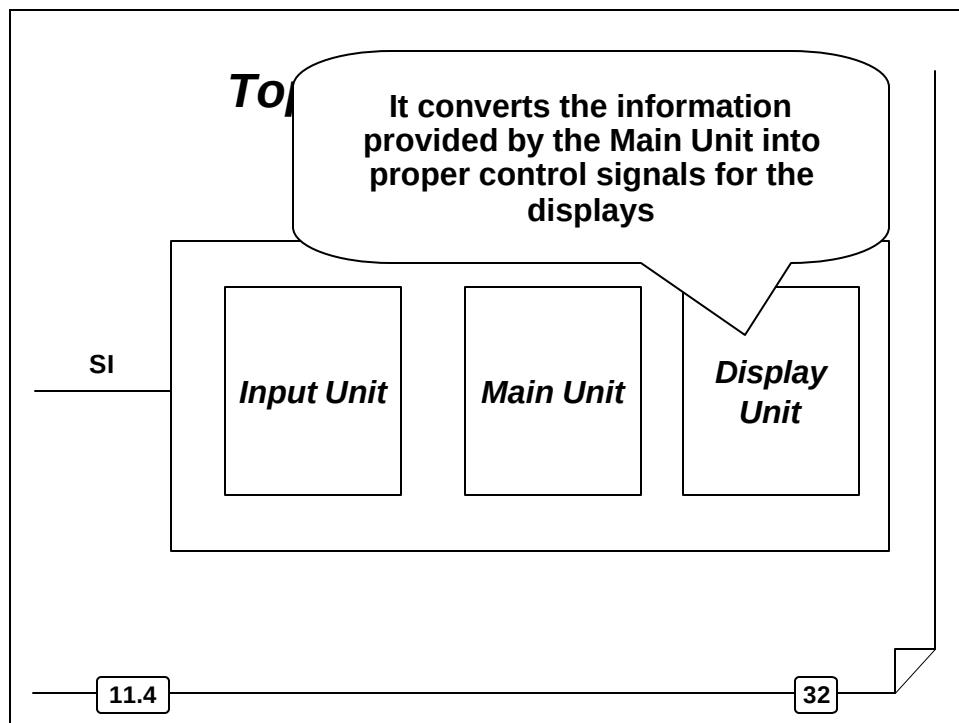
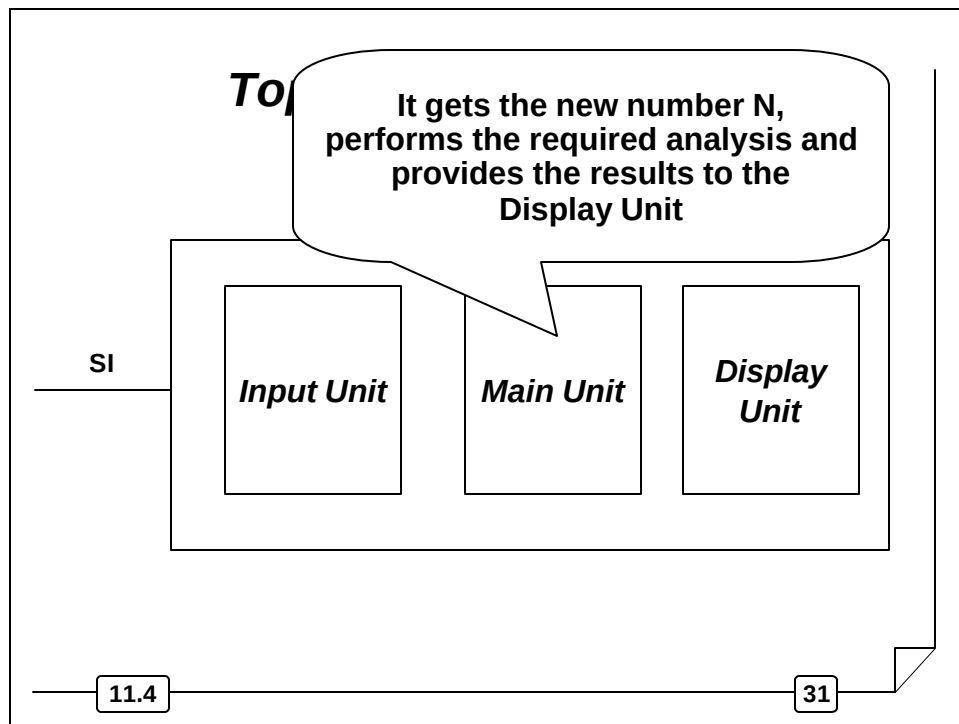
Top level view

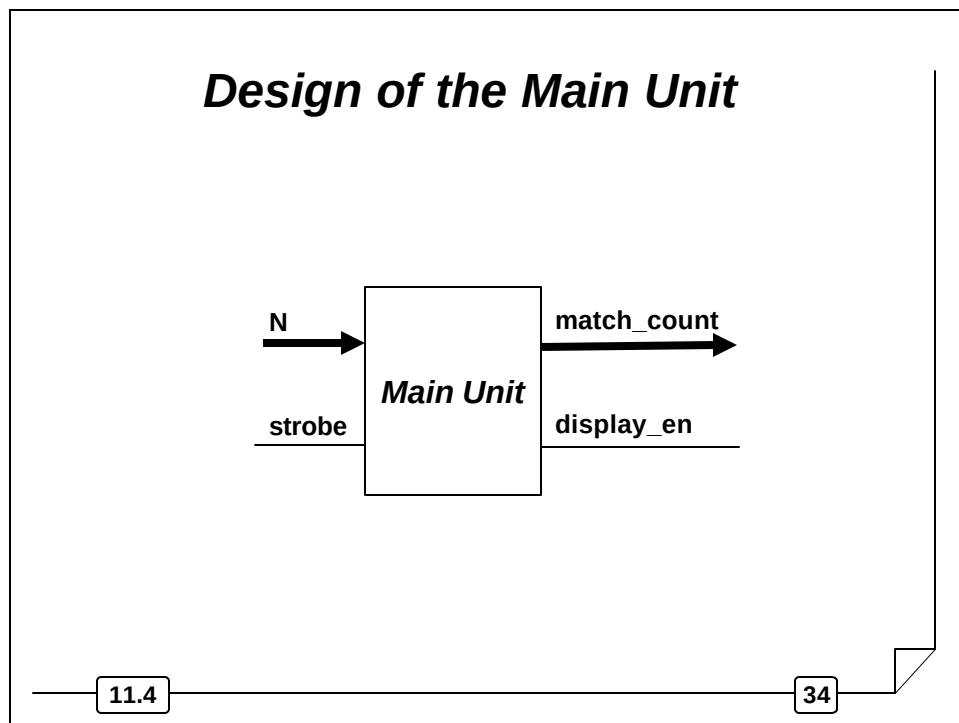
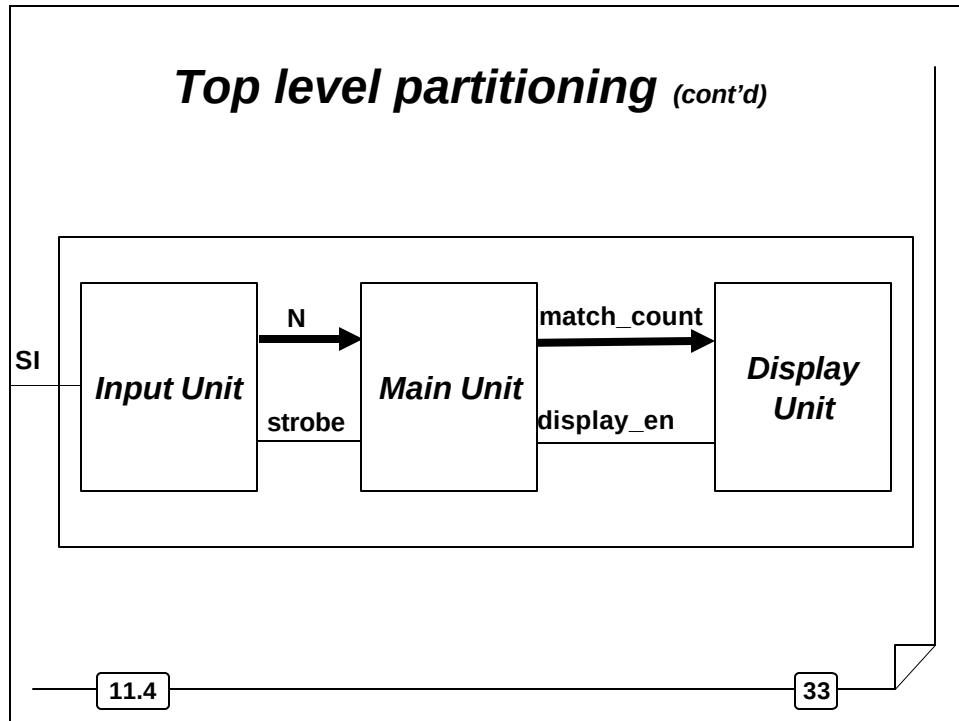
SI

11.4

28







Elements definition: constants

<i>Label</i>	<i>Type</i>		<i>Range</i>
ROM_WIDTH	constant	integer	16
ROM_SIZE	constant	integer	1024

Elements definition: I/O

<i>Label</i>	<i>Type</i>		<i>Range</i>
clk	in	std_logic	single bit
reset	in	std_logic	single bit
N	in	integer	0 to (2**ROM_WIDTH)-1
strobe	in	std_logic	single bit
display_en	out	std_logic	single bit
match_count	out	integer	0 to ROM_SIZE

Elements definition: internal signals

<i>Label</i>	<i>Type</i>		<i>Range</i>
acc	signal	integer	$-2^{**}(\text{ROM_WIDTH} - 1)$ to $2^{**}(\text{ROM_WIDTH} - 1) - 1$
address	signal	integer	0 to ROM_SIZE - 1

Intermediate STG of the Main Unit

match_count <= 0
acc <= 0
address <= 0
display_en <= '0'

init

11.4

38

