

Lecture
6.1

Combinational design: the basic approach



Paolo PRINETTO
Politecnico di Torino (Italy)
University of Illinois at Chicago, IL (USA)

Paolo.Prinetto@polito.it
Prinetto@uic.edu
www.testgroup.polito.it

Goal

- This lecture presents the basic approach to manual synthesis of Combinational networks.

6.1

2

Prerequisites

- Modules 4 and 5, and Lecture 2.1

6.1

3

Homework

- No particular homework is foreseen

6.1

4

Further readings

- No particular suggestion

6.1

5

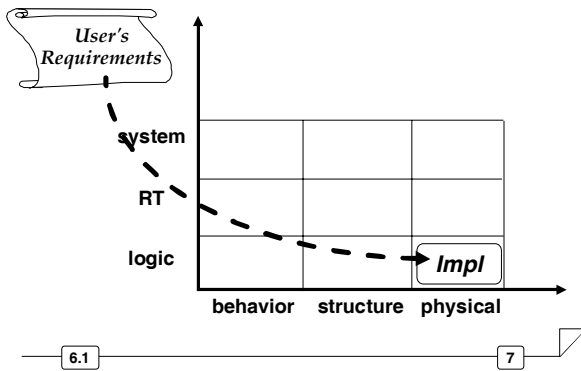
Outline

- Manual synthesis approaches
- Manual synthesis steps

6.1

6

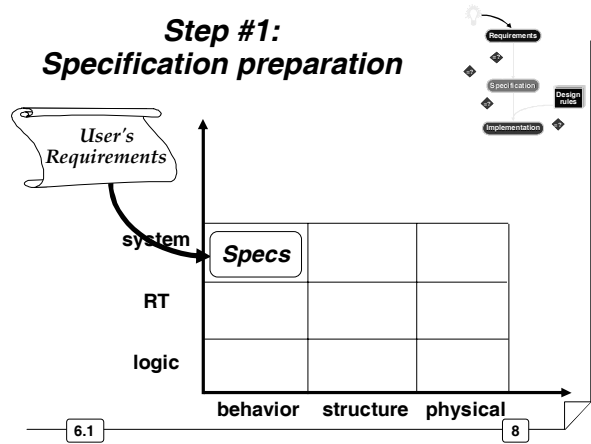
Logic level design



6.1

7

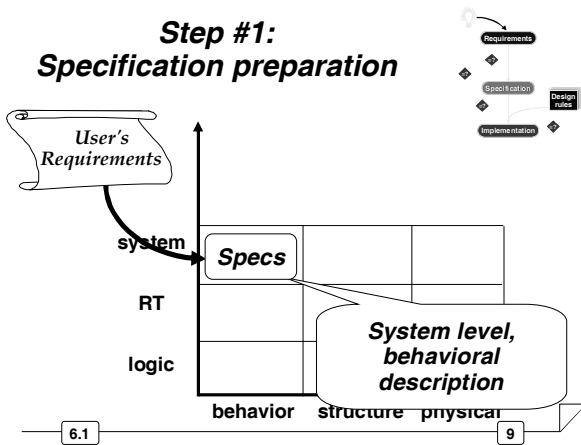
Step #1: Specification preparation



6.1

8

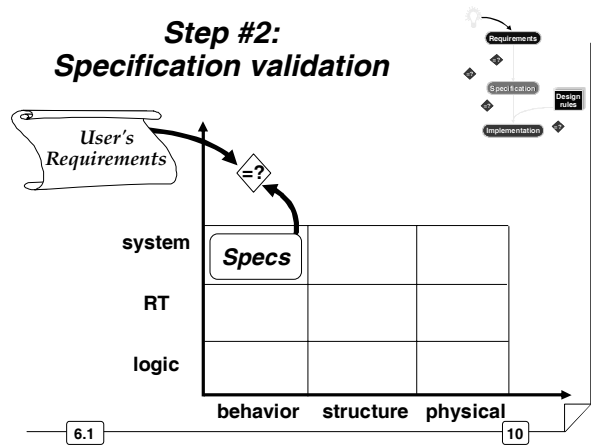
Step #1: Specification preparation



6.1

9

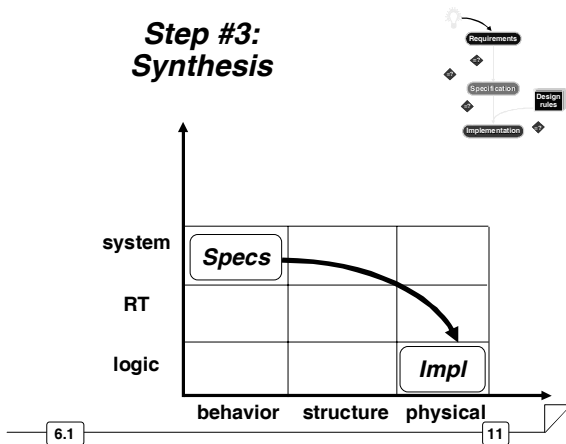
Step #2: Specification validation



6.1

10

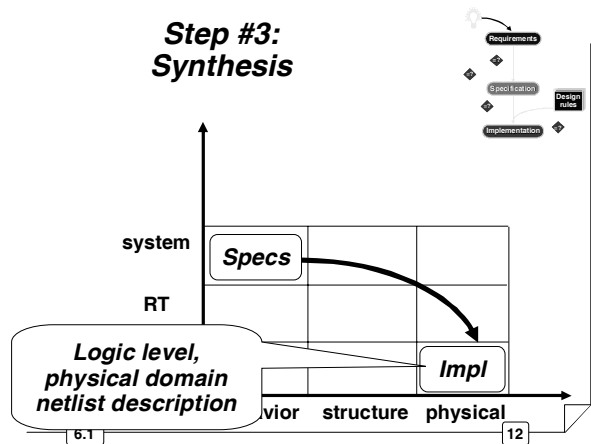
Step #3: Synthesis



6.1

11

Step #3: Synthesis



6.1

12

Manual synthesis approaches

Several approaches are possible, not necessarily strictly each other orthogonal.

They could be classified as follows:

- *Purely manual*
- *Partially automated*
- *Partitioning based.*

6.1

13

Purely manual approach

- Karnaugh map based
- When performing a manual synthesis it is:
 - rather easy to minimize the maximum delay (by designing 2-logic-level circuits, only)
 - extremely hard to minimize the area, trading off with the maximum delay.

6.1

14

Purely manual approach (cont'd)

- Applicable when dealing with very few PIs, only (PIs ≤ 6)
- It will be presented in the sequel of this lecture.

6.1

15

Partially automated approach

- Some of the sub-steps can be accomplished resorting to tools freely downloadable from the web
- Applicable when the behavior of each PO has been described by a Boolean Function expressed as a set of cubes

6.1

16

Partially automated approach

- Some of the sub-steps can be accomplished resorting to tools freely downloadable from the web
- Applicable when the behavior of each PO has been described by a Boolean Function expressed as a set of cubes
- No significant limitation w.r.t. the # of PIs.
- It will be presented in lecture 6.4.

6.1

17

Partitioning-bases approach

- The system is first partitioned in functional blocks
- Each functional block is then implemented resorting to one of the above mentioned approaches
- The system is eventually designed simply assembling the functional blocks

6.1

18

Partitioning-bases approach

- The system is first partitioned in functional blocks
- Each functional block is then implemented resorting to one of the above mentioned approaches
- The system is eventually designed simply assembling the functional blocks
- No significant limitation w.r.t. the # of PIs
- It will be presented in lecture 6.5

6.1

19

Partitioning-bases approach (cont'd)

- Historically this method was thoroughly applied in the 70's and 80's when digital systems were mainly implemented by PCBs and the chips available were mostly the RT level basic blocks presented in lecture 5.2 and 5.3.

6.1

20

Outline

- Manual synthesis approaches
⇒ *Manual synthesis steps*

6.1

21

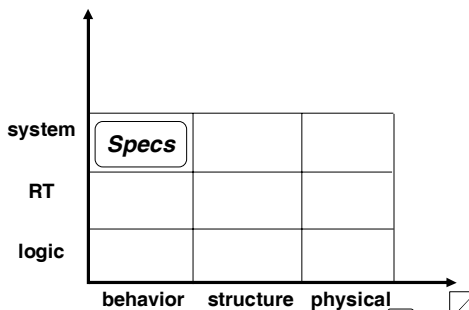
Manual synthesis steps

Manual synthesis is usually performed in several sub-steps.

6.1

22

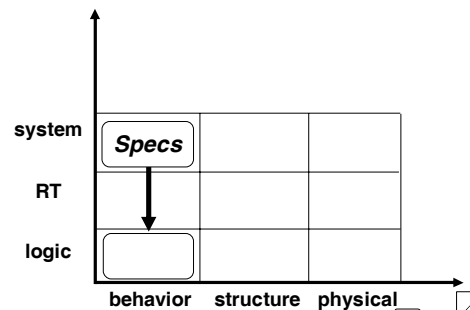
Step #3.1: Logic level refinements



6.1

23

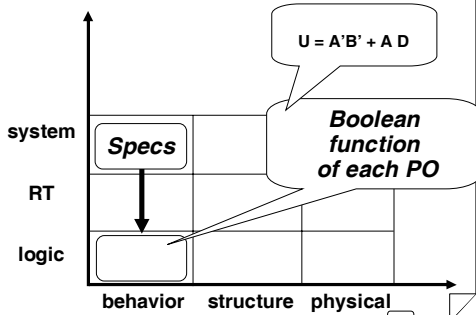
Step #3.1: Logic level refinements



6.1

24

**Step #3.1:
Logic level refinements**

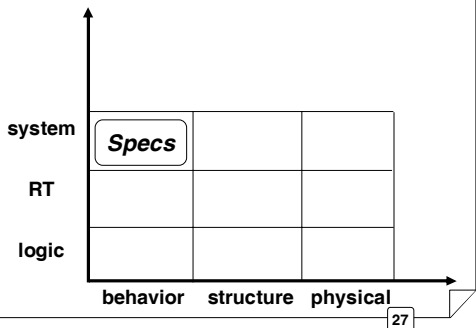


**Step #3.1:
Logic level refinements**

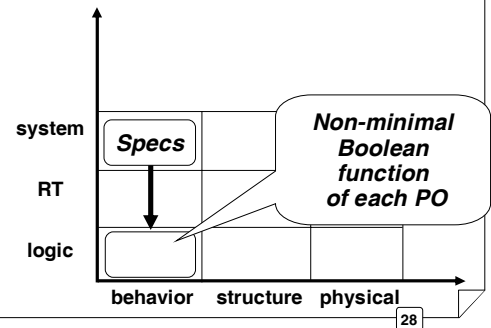
It's usually performed in 2 sub-sub-phases:

6.1 26

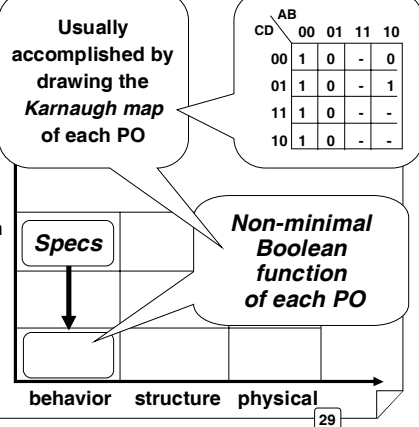
**Step #3.1.1:
1st refinements**



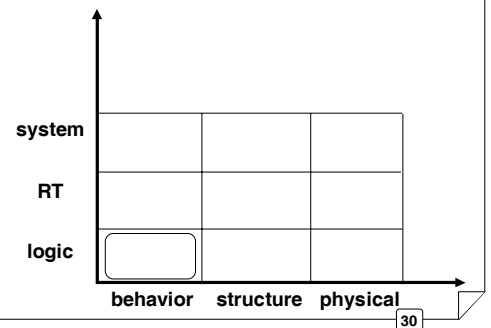
**Step #3.1.1:
1st refinements**



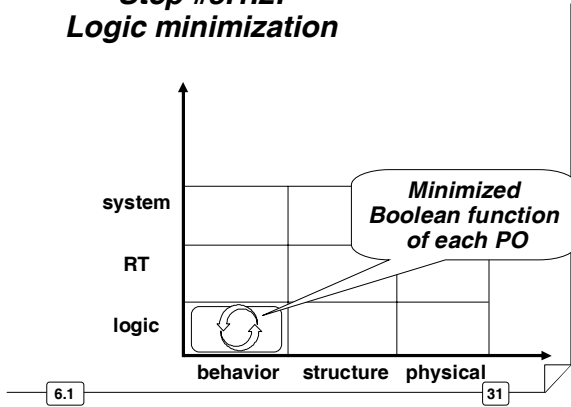
**Step #3.1.1:
1st refinements**



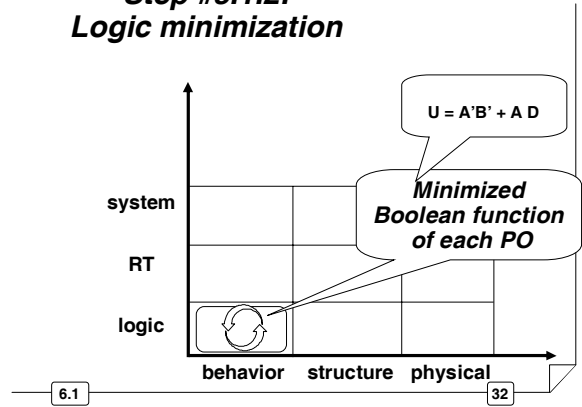
**Step #3.1.2:
Logic minimization**



**Step #3.1.2:
Logic minimization**



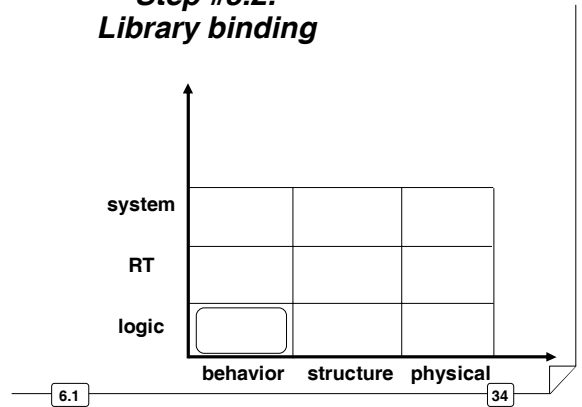
**Step #3.1.2:
Logic minimization**



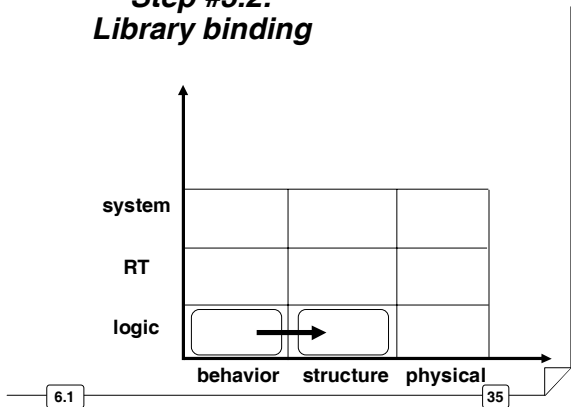
Note

A detailed analysis of manual logic refinements will be presented in lecture 6.2 whereas some algorithmic approaches will be introduced in lecture 6.3

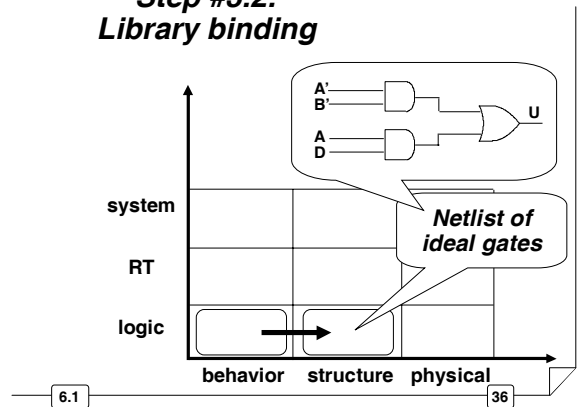
**Step #3.2:
Library binding**



**Step #3.2:
Library binding**



**Step #3.2:
Library binding**



**Step #3.2:
Library binding**

The library binding of sp expressions is trivial:

<i>logic operation</i>	<i>logic gate</i>
sum	or
product	and
complement	not

6.1

37

Example

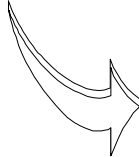
$$U = A'B' + AD$$

6.1

38

Example

$$U = A'B' + AD$$



6.1

39

Example

$$U = A'B' + AD$$

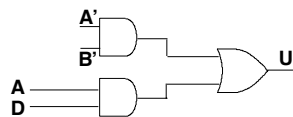
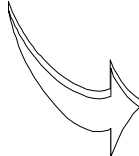


6.1

40

Example

$$U = A'B' + AD$$

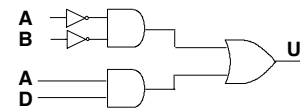


6.1

41

Example

$$U = A'B' + AD$$



6.1

42

Multiple outputs

When dealing with multiple outputs, shared implicants are naturally mapped in shared gates.

6.1

43

Example

- $X = b'd + bcd$
- $Y = bd' + bcd$

6.1

44

Example

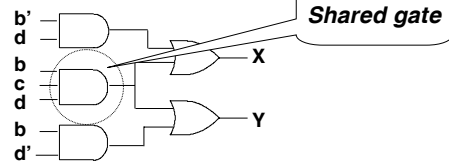
- $X = b'd + bcd$
 - $Y = bd' + bcd$
- Shared implicant**

6.1

45

Example

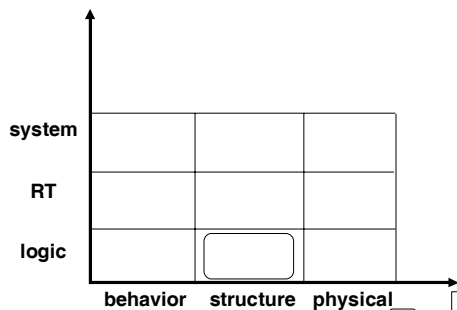
- $X = b'd + bcd$
 - $Y = bd' + bcd$
- Shared implicant**



6.1

46

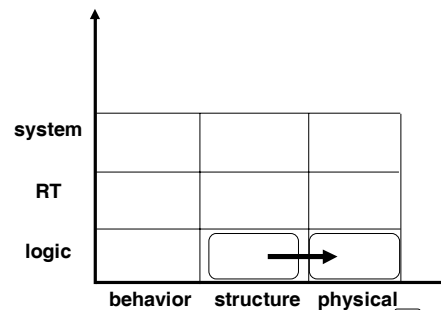
Step #3.3: Technology mapping



6.1

47

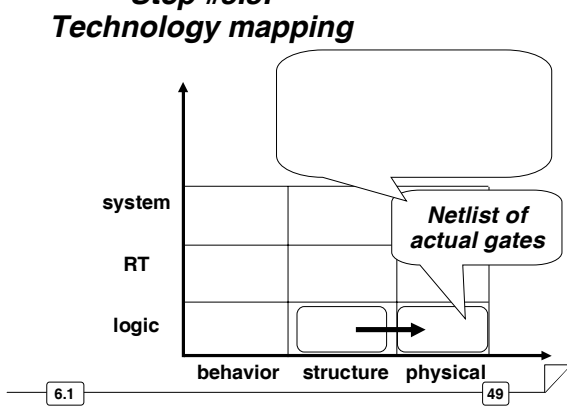
Step #3.3: Technology mapping



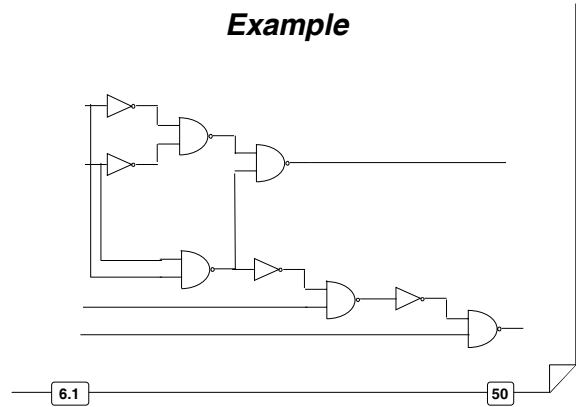
6.1

48

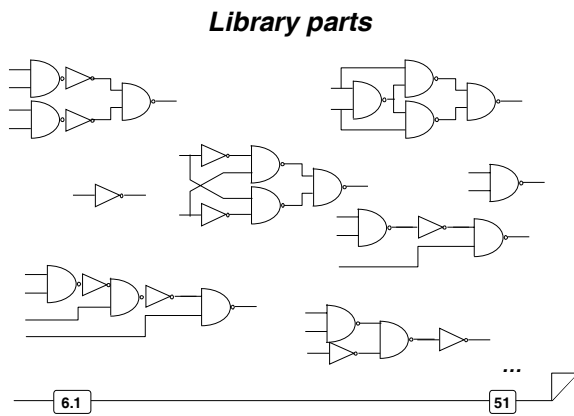
**Step #3.3:
Technology mapping**



Example



Library parts



Solution

