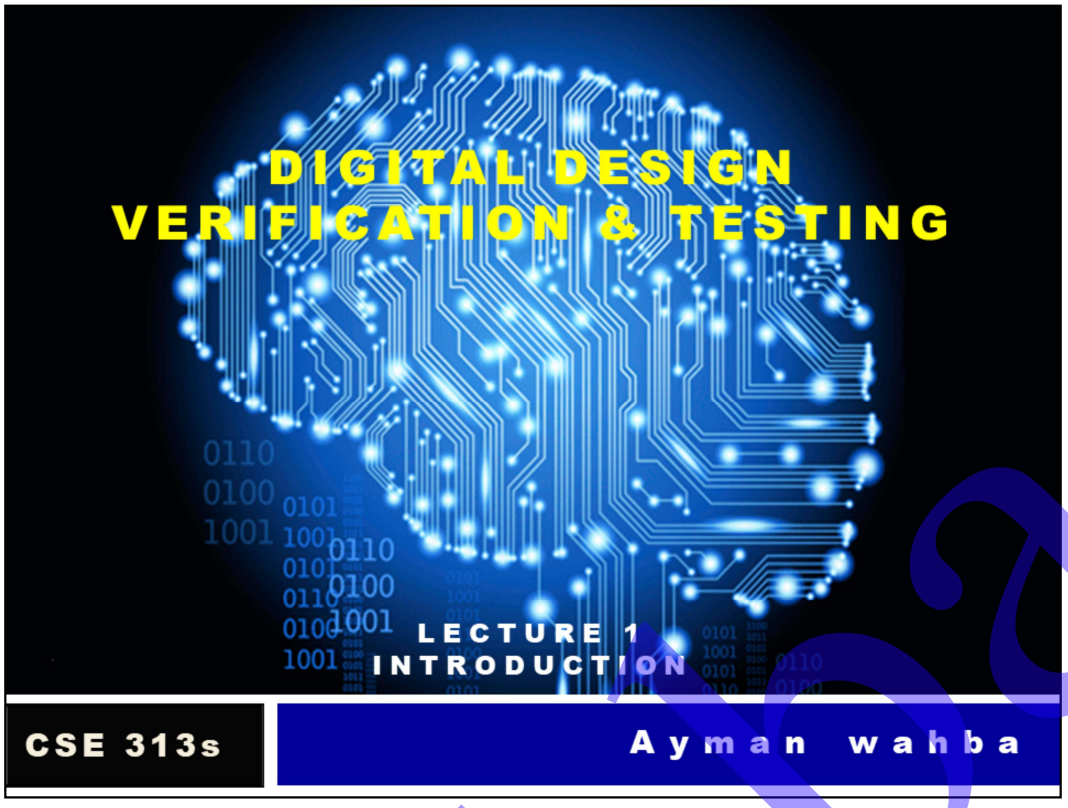

DIGITAL DESIGN
VERIFICATION & TESTING
LECTURE 1
INTRODUCTION
CSE 313s
Ayman wahba

Wahba

Course Information

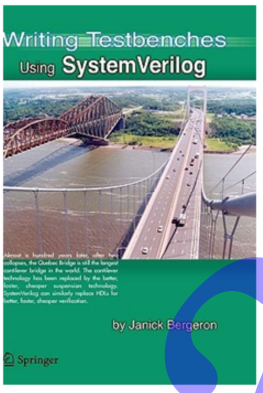
2

- **Instructor**
  - Ayman M. Wahba, Ph.D. ayman.wahba@eng.asu.edu.eg
- **Lecture Time and Location**
  - Tuesday 10:00 – 12:00 Weekly
- **Office Hours**
  - Available through Microsoft Teams all days of the week
  - Available at my office on Saturday 9:00 – 12:00

Wahba

Textbook

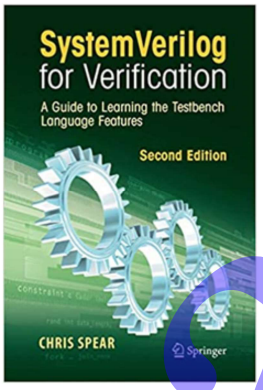
- Janick Bergeron, Writing Testbenches using System Verilog, Springer, 2006.



Wanba

Textbook

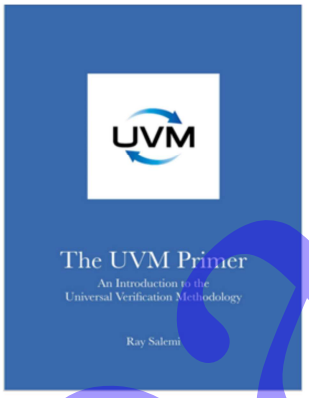
- Chris Spear, System Verilog for Verification: A Guide to Learning the Testbench Language Features, 2nd Edition, Springer, 2008.



Wahba

Textbooks

- Ray Salemi, The UVM Primer: An Introduction to the Universal Verification Methodology, Boston Light Press, 2013.



Wanba

Assessment Criteria

- Two quizzes & two assignments 10 %
  - Actually 4 assignments, only 2 are marked.
- Practical Activities 10%
- Final Exam 60%
  - You must attend at least 75% of the tutorials/labs to enter the exam
- Mid-Term Exams 20%
  - The mid-term exam is held after the 7th week.

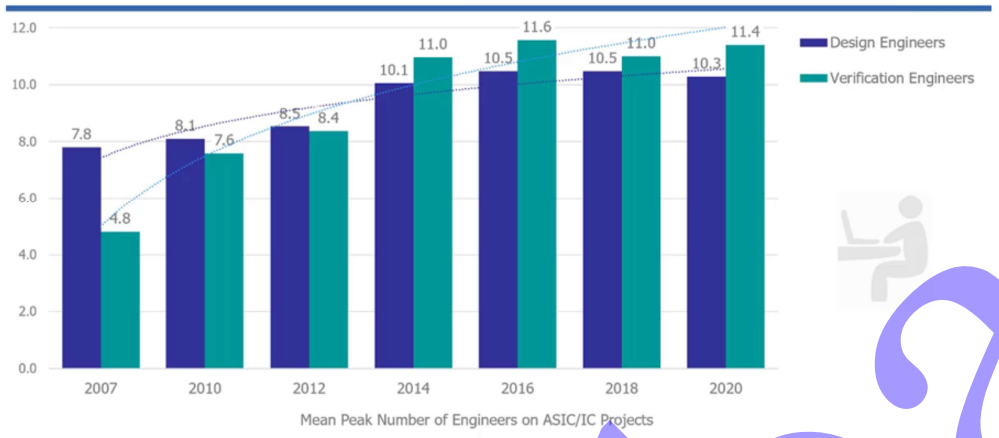
Total 100

Wahba

- Verification is not a testbench, nor is it a series of testbenches.
- Verification is a process used to demonstrate that the intent of a design is preserved in its implementation.
- We will see the differences between various verification approaches as well as the difference between testing and verification.
- If you survey hardware design groups, you will learn that between 60% and 80% of their effort is dedicated to verification.
- This may seem unusually large, but included in "verification" all debugging and correctness checking activities, not just writing and running testbenches.
- Every time a hardware designer pulls up a waveform viewer, he or she performs a verification task.
- With today's ASIC and FPGA sizes and geometries, the challenge is to get the right design, working as intended, at the right time.

Wanba

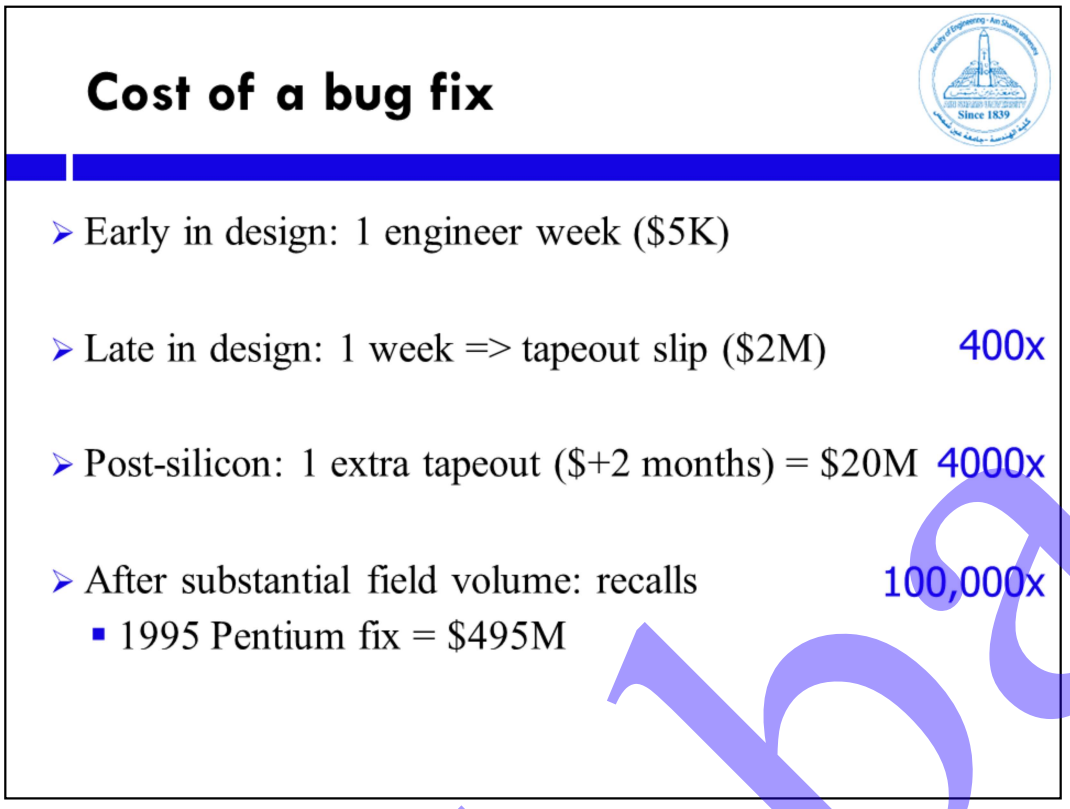
Design vs. Verification Engineers



- CAGR (Compound Annual Growth Rate) for verification is 5.5%
- CAGR (Compound Annual Growth Rate) for design is 1.5%

Remind them what CAGR stands for

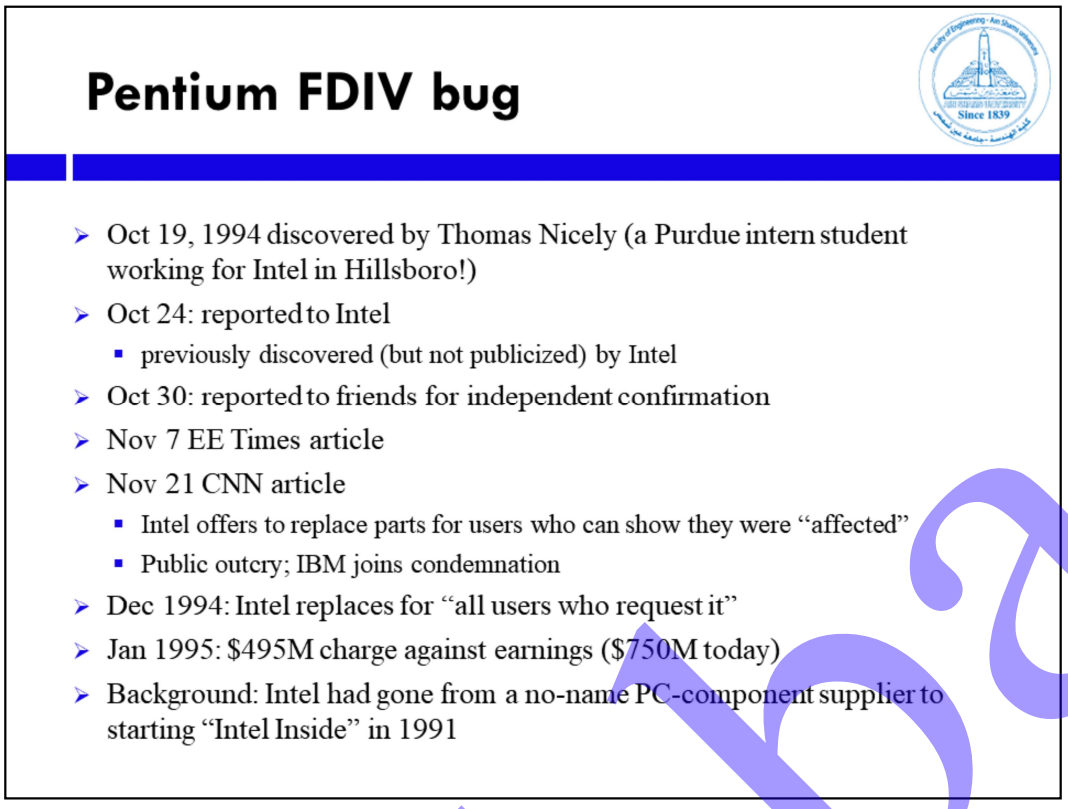
Wahba



Assume you're selling a GPU; 1M units/year, $100/unit. so $100M/year or $2M/week

Hard to compute the cost of a tapeout (depends on your volume, etc and how many layers it involves). Scott's numbers are $1M/layer, or $40M for a full-layer tapeout and $20M for metal-only. I'll use $50M since Scott's numbers are slightly old.

Wabba



The original bug find was by Tom Kraljevic, a Purdue intern student working for Intel in Hillsboro! But Intel never made it public (or so says Wikipedia).

Yes, the half-billion dollar charge was from, according to Wikipedia, "only a small fraction of users" requesting replacement CPUs.

Did they mishandle the issue? Perhaps. They were learning how big, famous companies must behave!

The motto – if you're successful and you make money – then everyone wants a piece of you.

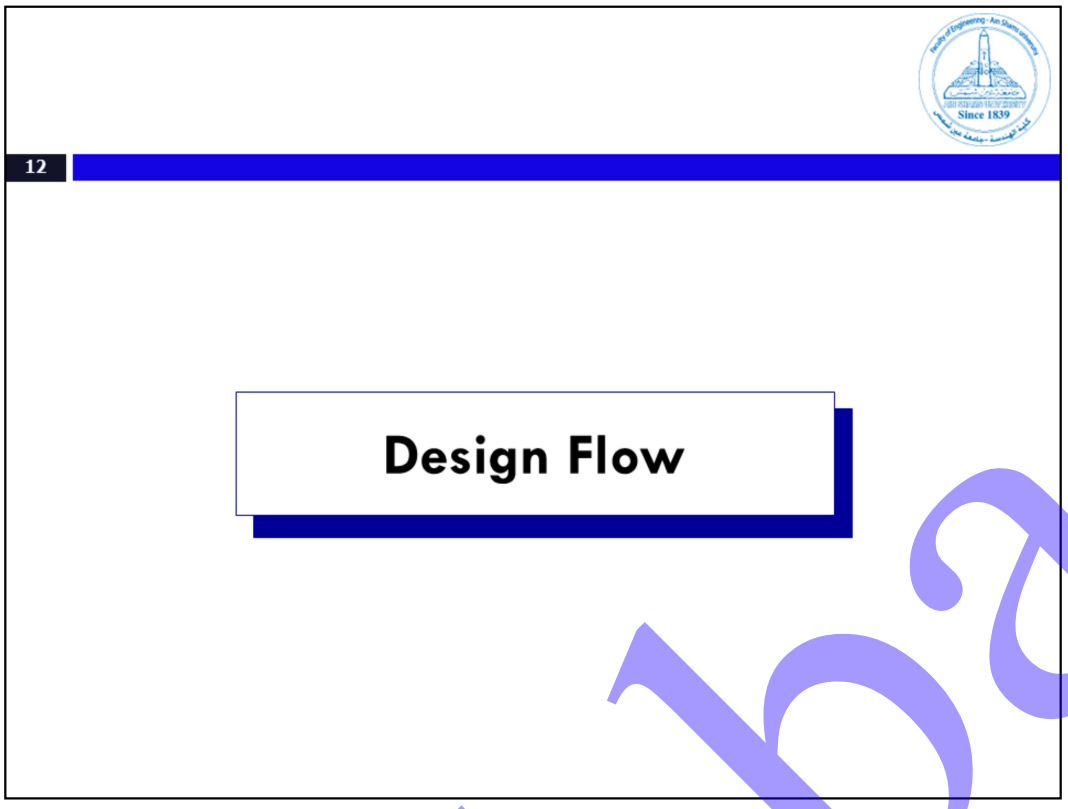
Wahba

Evolution of verification

- The "good old days"
  - designs were relatively simple
  - everyone "verified" their own work
  - and it usually worked fine!
- But then designs got more complex
  - Billions of transistors, numerous clock domains, embedded CPUs, …
  - Too many tests to write
  - Specialized infrastructure and skills are needed
  - And so… specialists are needed!
- Realization: verifying your own design is a bad idea
  - We humans are really bad at finding our own flaws

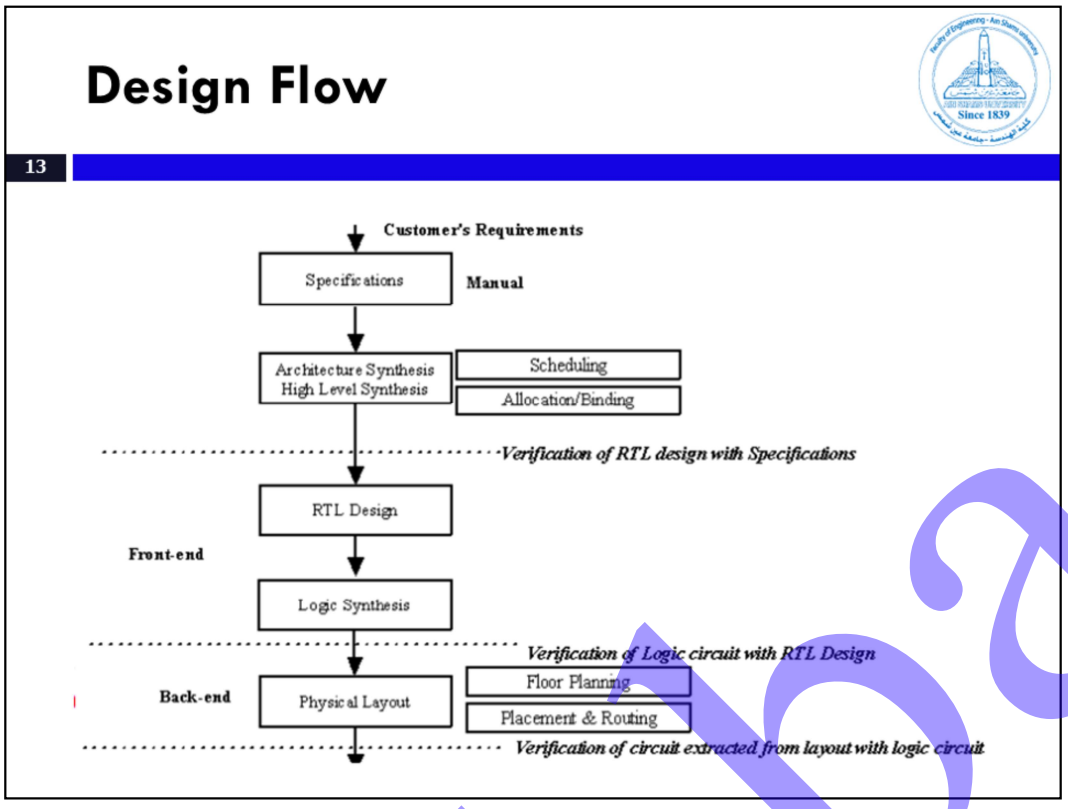
Nvidia Ampere = over 50 billion transistors

Wahba



- Ver

Wanba



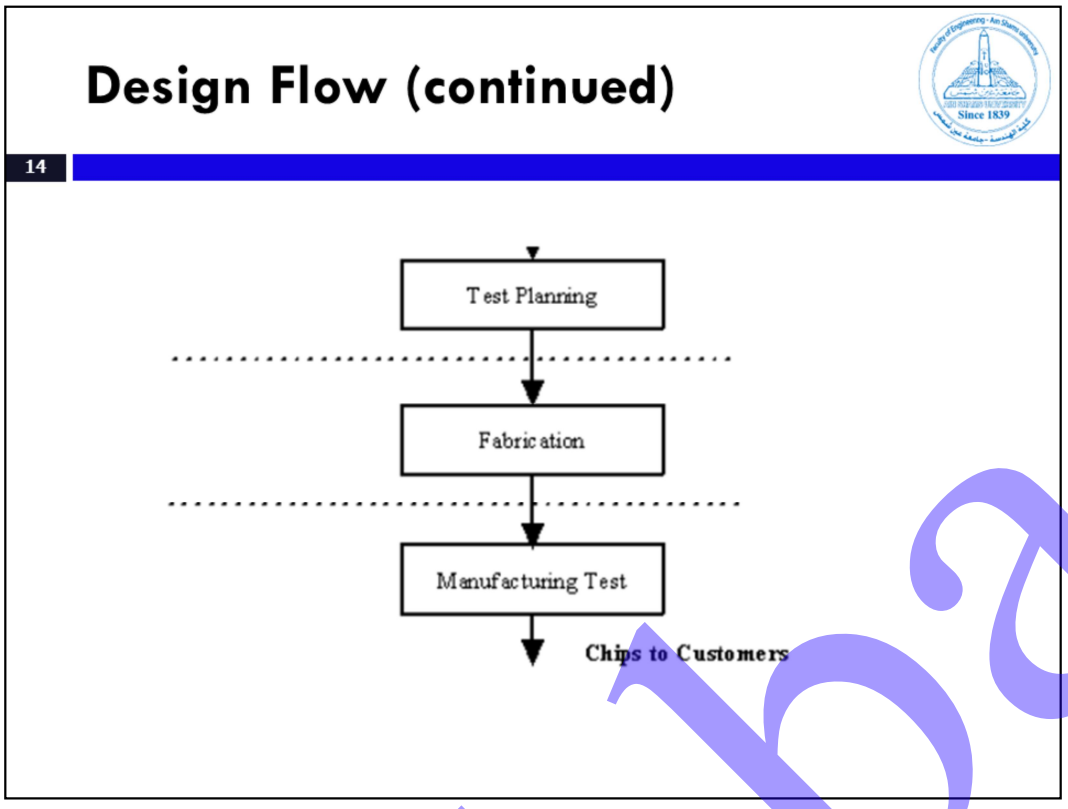
- This is the first lecture of the course on Design verification and test of Digital VLSI circuits.
- What is a circuit? It is a device or an object that transforms electrical signals. You give one form of signals and you get other forms of signals.
- For digital circuits, our inputs and outputs are different voltage levels. We don't have any kind of continuous-time current or voltages. If you have continuous-time voltage or current, then you have an analog circuit.
- In the case of digital circuits we have discrete values (normally 2 values, that we call 0 and 1). 0 is near 0 volts, and 1 is near 5 volts (in some technologies, it is around 3.3 volts).
- For the design, you are given a specification, for example a half adder, so you know that sum = a xor b, and carry = a.b
- What do we mean by *verification* ? You give some inputs to your circuit, and then check whether it performs as desired.
- So, you can give the possible combinations of inputs 00, 01, 10, 11, for the circuit to be verified.
- What does *test* mean? You designed the circuit, and then you fabricated it into hardware, (VLSI: Very Large Scale Integration, i.e. millions of transistors). Once the circuit is fabricated, you give inputs to the hardware and see if everything is operating fine or not. The errors here are due to fabrication not the design.
- *So, verification is done on the design, while test is done on the circuit after*

Wahba

fabrication.

Wabba

Design Flow (continued)



Wahba

1. Specification

Step1: Specification Design

In a typical VLSI flow, we start with system specifications, which is nothing but technical representation of design intent. To explain the flow, the following example will be used through this section.

Example:

Specification: out1=a+b; out2=c+d; where a,b,c,d are single bit inputs and out1,out2 are two bit outputs (sum and carry).

Wahba

2. High Level Synthesis

Step 2: High level Synthesis

High-level synthesis (HLS) algorithms are used to convert specifications into Register Transfer Level (RTL) circuits.

•HLS, sometimes referred to as architectural synthesis is an automated design procedure that interprets an algorithmic description of the design intent and creates hardware at RTL that implements that behavior.

•The input to a HLS tool is design intent written in some high level hardware definition language like SystemC, System Verilog etc.

•The HLS tool first schedules the computations (required to meet the specifications) at different control steps.

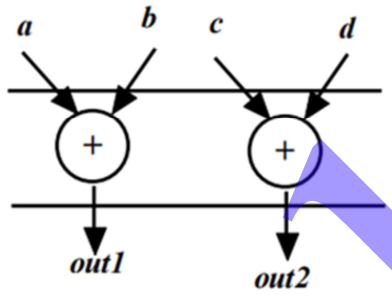
•Following that, depending on availability of hardware units and time constraints, the scheduled computations (comprising instructions and variables) are allocated and binded to the hardware units like adders, multipliers, multiplexors, registers, wires etc.

Wahba

HLS Example

HLS Example:

In the example there are two operations (addition of single bit numbers) and none of them depend on each other. So both the operations can be scheduled in a single control step. However, if there are dependencies e.g., out1=a+b; out2=out1+d; then "out1=a+b;" is scheduled in 1st control step whereas "out2=out1+d;" is scheduled in 2nd control step.

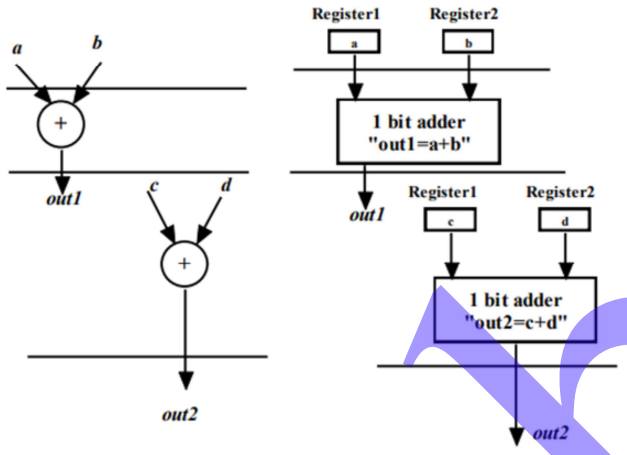


Wahba

HLS Example (continued)

•Depending on availability of hardware resources and time constraints the scheduled operators and variables are allocated and binded to hardware units.

Let there be one adder and two registers in the library.



Wahba

HLS Example (continued)

There is one adder and two registers in the library. So the two operations (addition) of the example, even if scheduled in one control step, cannot be allocated to the single adder. Similarly, the four variables cannot be allocated to two registers.

In the running example with the given resource constraints, the two operations can be done in two control steps:

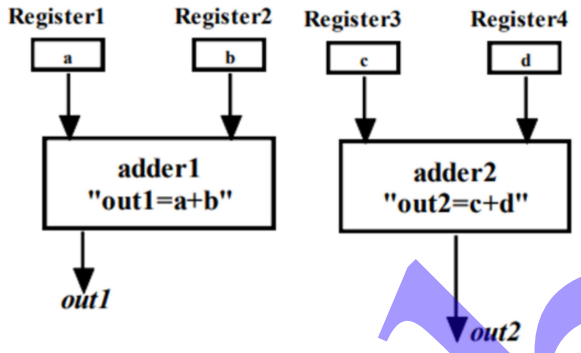
Step 1- variable a is allocated to Register1, variable b is allocated to Register2 and operation "out1=Register1+Register2;" is allocated to adder;

Step 2- variable c is allocated to Register1, variable d is allocated to Register2 and operation "out2=Register1+Register2;" is allocated to adder.

Wahba

HLS Example (continued)

However, if there are two adders and four registers in the library then both the operations can be carried out in one control step.



Wahba

HLS Example (continued)

•Finally, based on allocation and binding, the control unit is to be designed (at high level).

•If the allocation/binding is according to (2 adders + 4 registers), the control is trivial.

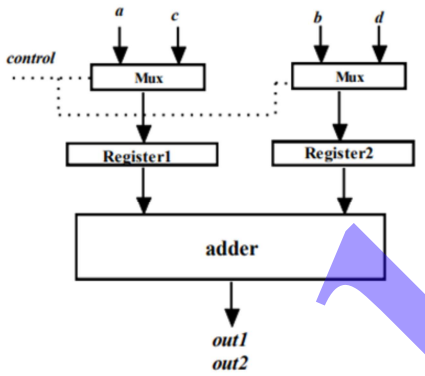
•However, if the allocation is according to (1 adder + 2 registers), then the control circuit needs to provide signals that can do multiplexing between *a* and *c*, *b* and *d;*

- •In 1st control step, *a* should be fed to Register1 and *b* should be fed to Register2,
- •In 2nd control step, *c* should be fed to Register1 and *d* should be fed to Register2.

Wahba

HLS Example (continued)

•Figure below illustrates the block diagram where control modules are added after allocation and binding (1 adder + 2 registers).
•It may be noted that *control* signal is not available as an external pin which can be controlled by the user. "*Control*" is connected to some signal generated by the system, which alternates in every control step thereby making its value 0 in 1st step and 1 in the 2nd.



Wahba

HLS Example (continued)

The HLS tool generates output comprising,
(i) operations-variables allocated-binded to hardware units and
(ii) control modules.

The output of HLS tool is called Register Transfer Level (RTL) circuit because data flow, data operations and control flow are captured between registers.

After HLS, RTL circuits are transformed into logic gate level implementation; the step is called logic synthesis.

Wahba

Verification (Specs vs. RTL)

Before the staring of logic synthesis, one needs to verify if the RTL is equivalent to the specifications.

In the running example, we can verify by applying all possible input conditions of a,b,c,d (along with control) to the RTL and checking if out1 and out2 are as expected.

However, if the RTL has about hundreds of inputs then exercising all possible inputs is impossible because of the exponential complexity (i.e., if there are n inputs then all possible input combinations are 2^n).

So we need to have formal verification methods which verify equivalence of RTL with input specifications.

Wahba

Verification (continued)

25

Broadly speaking, for formal verification we need to model the RTL circuit and the specifications using some formal modeling techniques and verify that both of them are equivalent.

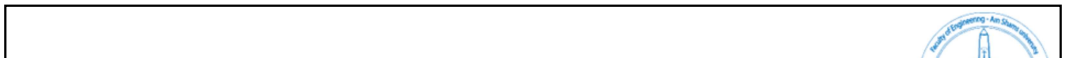
In other words, equivalence is determined without applying inputs.

Control and Data Flow Diagram (CDFG), a formal modeling, to capture the RTL. Finite State Machine (FSM) to model the control logic.

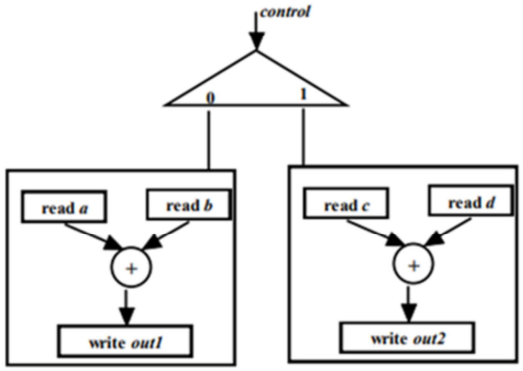
This example being very simple, we can see that both specifications and the model are equivalent. Formal techniques for checking equivalence will be elaborated in "VERIFICATION" section of the course.

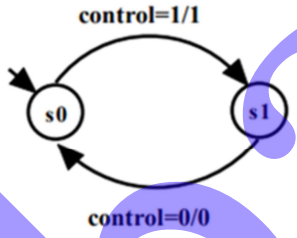
Wahba

Verification (continued)



This example being very simple, we can see that both specifications and the model are equivalent.





Wahba

3. Logic Synthesis

•After the RTL is verified to be equivalent to system specification, logic synthesis is performed by CAD tools.

•In logic synthesis all blocks of the RTL circuit is transformed into logic gates and flip-flops.

- •For the running example all the blocks namely, adder, multiplexers, control logic etc. need to be synthesized to logic gates.

Will illustrate synthesis only for the adder module and for the rest, similar procedure holds.

We first determine the Boolean function of the adder module, in terms of mean terms.

a	*b*	*Out1(sum)*	*Out1(carry)*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Wahba

3. Logic Synthesis (continued)

28

•From the table we have Boolean equations for
Out1(sum)= a ⊕ b, and Out1(carry) = a.b

•After the equations are obtained they need to be minimized so that the circuit can be implemented using minimal number of gates. Karnaugh map, Quine–McCluskey algorithm etc. are some standard techniques to minimize Boolean functions.

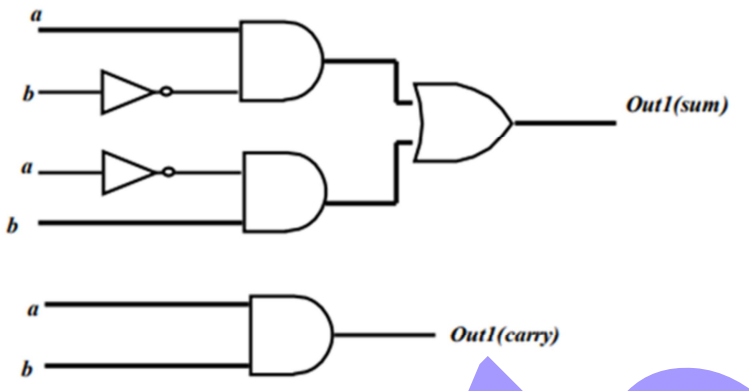
•In this example of the adder, the equations are already minimized and can be directly converted to Boolean gate implementation as shown.

•Karnaugh map and Quine–McCluskey techniques work well if the number of inputs is less. However, in case of practical VLSI circuits the number of inputs are in orders of hundreds, so minimization is carried out using heuristics techniques.

•Again equivalence of logic synthesis output should be established with RTL design.

Wahba

3. Logic Synthesis (continued)



Wahba

4. Backend

•Once the logic level output of the circuit is obtained we move to backend phase of the design process.

In backend we start with a soft version of the silicon die where the chip will be finally fabricated.

- •Broad plan regarding placement of gates, flip-flops etc. (output of logic synthesis) in appropriate places in the soft representation of the chip; this process is called Floorplan.
- •Exact locations in the die (software representation) where the circuit components are placed; this is called Placement.
- •Required interconnections (as given in the logic circuit) among the gates that are placed in exact positions in the die; this process is called routing.

Again equivalence of output of Backend process should be established with logic design.

Wahba

5. Test

•In VLSI designs millions of transistors are packed into a single chip, thereby leading to manufacturing defects. So all chips need to be physically tested by providing input signals from a pattern generator and comparing responses using a logic analyzer.

• As in the case of verification, testing by applying all possible input combinations is prohibitive, due to curse of dimensionality problem.

•The testing problem is more time hungry than verification because all chips need to be tested while only "one" design is to be verified.

•Testing by applying all possible input combinations is called exhaustive functional testing, which is avoided because of prohibitive time requirements.

Wahba

5. Test (continued)

•Testing is therefore done based on "structure" of the circuit and is called structural testing.

•In structural testing we first decide on set of faults that can occur, called Fault Models; stuck-at, bridging etc. are some well known fault models.

•Then we apply only those inputs which are required to validate that faults (as per fault model) are not present.

•Number of patterns required to perform structural testing is exponentially lower than that required for exhaustive functional testing.

• In Test Planning step, given a logic level circuit and fault model, we generate patterns, which when applied to a circuit determines that no fault from the fault model exists in the circuit.

Wahba

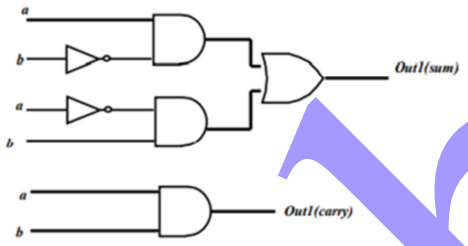
5. Test (continued)

•Test planning for the adder module of the example assuming that fault model is "stuck-at".

- •In "stuck-at" fault model each line of the circuit is assumed to have two types of faults i.e., s-a-1 and s-a-0.
- •So if there are n lines in a circuit then in all there can be 2n stuck-at faults in the circuit.

•In test planning we need to find input patterns which can determine that none of the stuck-at faults are present.

•In the circuit of Figure as there are 12 lines (9 lines in circuit for "sum" and 3 lines in the circuit for "carry"), there can be 24 stuck-at faults.

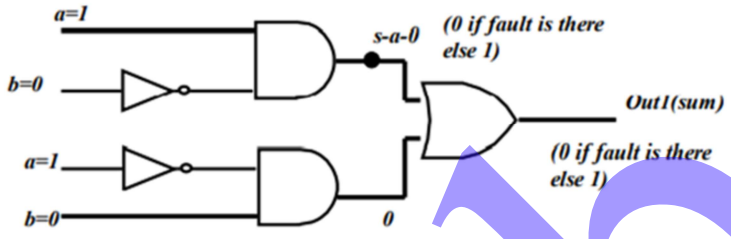


Wahba

5. Test (continued)

•Here we will illustrate for only one fault and the same holds for all the other 23 faults. Let there be a stuck-at-0 fault in the output of one AND gate of the circuit for "sum".

•If a=1 and b=0 is applied as inputs, then "output1(sum)" is 0 if fault is present, 1 otherwise. So a=1 and b=0 can verify the absence of fault by comparing output with 1.



Wahba

6. Fabrication, Test and Marketing

35

Once all steps are completed and verification after each level of transformations are done, the chips are fabricated, physically tested and fault free chips are sent for marketing.

Wahba