Lecture 3: Timing Analysis – Part 2

Thanks to S. Devadas and K. Keutzer for several slides

RTL Synthesis Flow

- **RTL Synthesis**
  - HDL
    - HDL Simulation/Verification
  - netlist
    - logic optimization
    - physical design
      - layout

- **Graph / Rectangles**
  - Timing Analysis

- **Library/module generators**
  - Boolean circuit/network
  - FSM, Verilog, VHDL

- **HDL Simulation/Verification**
  - Boolean circuit/network

K. Keutzer
Timing Analysis / Verification

Verifying a property about **system timing**

Arises in many settings:
- Integrated circuits
- Embedded software
- Distributed embedded systems
- Biological systems
- ...  

Illustrates many concepts of this course
- Graph algorithms
- Optimization
- SAT solving
- Numerical simulation

Graph vs. Circuit

Delay of a combinational circuit depends on
- Circuit topology (graph model)
- Delay model
- Boolean behavior

We have only considered circuit topology so far.

Longest/shortest path found on the graph can be very pessimistic:
- Paths can be FALSE
- Delay values are BOUNDS
False Paths (consider Transition Mode)

A path is false if it cannot be responsible for the delay of a circuit.

Graph model implies path of length 6.
False vs. True Paths

TRUE path = one that can be responsible for the delay of a circuit

Need techniques to find whether a path is TRUE or FALSE

Incremental k-longest path algorithms are useful here. WHY?

The Fixed Delay Model: Constant delay for each gate (or wire)

Transition delay is 0, for both input transitions.
Paradoxical Behavior with the Transition Model?

Transition delay is 0, for both input transitions. Consider the “faster” circuit.

Problems with Fixed Delay + Transition Model

1. Transition model can be tricky to reason about
2. Fixed gate delays are unrealistic, due to manufacturing process variations

More realistic delay model: Lower and upper bounds

Perform timing analysis for a whole family of circuits that share the same lower/upper bounds
Fixed Delays $\rightarrow$ Bounded Delays

Want algorithms that report the **critical path delay** of the **slowest circuit** in the circuit family.

Delay of 6 for the above circuit for transition model (longest path that can propagate a transition).

Floating-Mode Delay Model

Input transition $\rightarrow$ Single input vector condition

Pessimistic, but easier to compute
Floating-Mode Delay Model

Assume an input pair \(<v_1, v_2>\) has been applied, but we only look at \(v_2\) -- i.e. node values are unknown until set by \(v_2\) (pessimistic because we assume any \(v_1\) can be adversarially selected, to reason about long paths)

Assume the 1 at the input of the AND arrives before the 0 (even if in reality it arrives later and the gate output stays at 0 throughout, and no path is sensitized).

Roadmap for rest of lecture

Consider conditions under which paths are TRUE or FALSE under the floating-mode delay model with fixed delays
+ under floating-mode model, fixed and bounded delays yield same worst-case circuit delay (for same upper bounds)
+ worst-case delay under floating-mode model is upper bound on that under the transition model

Reference (posted on bSpace):
S. Devadas, K. Keutzer, S. Malik:
Controlling and Non-Controlling Values

A **controlling value** at a gate input is the value that determines the output value of that gate irrespective of the other input value. (the output value is called a **controlled value**)

A **controlling** value for an AND gate is 0 and for an OR gate is 1. (The controlled values are 0 and 1 resp.)

A **non-controlling** value for an AND gate is 1 and for an OR gate is 0.

What about NAND and NOR gates?

---

Static Sensitization

**Definition:** A path is statically sensitized by a vector \( V \), if along each gate on the path, if the gate output is a controlled value, the input corresponding to the path is the only input with a controlling value.

Input vector 100X statically sensitizes red path
Static Sensitization

Static sensitization is **sufficient** for a path to be responsible for the delay of a circuit

**WHY?**

Input vector 100X statically sensitizes red path

---

Is this path statically sensitzible?

**Definition:** A path is statically sensitized by a vector V, if along each gate on the path, if the gate output is a controlling value, the input corresponding to the path is the only input with a controlling value.
Is this path statically sensitizable?

No, red path is NOT statically sensitizable (work this out)

More on Static Sensitization

Are paths a,d,f,g and b,d,f,g statically sensitizable? Are they true paths?
Static Sensitization is too strong

A true path (one that is responsible for delay of a circuit) need not be statically sensitizable

Paths a,d,f,g and b,d,f,g are NOT statically sensitizable. But they are TRUE paths.

Static Co-sensitization

Definition: A path is statically co-sensitized by a vector V, if the input corresponding to the path presents a controlling value at each gate along the path whose output is a controlled value.

Not necessarily the ONLY controlling value
Static Co-sensitization

Definition: A path is statically co-sensitized by a vector \( V \), if the input corresponding to the path presents a controlling value at each gate along the path whose output is a controlling value.

Not necessarily the ONLY controlling value

Paths \( a,d,f,g \) and \( b,d,f,g \) are statically co-sensitizable

Static Co-sensitization and Delay

Static co-sensitization is \textbf{necessary} for a path to be responsible for the delay of a circuit. (WHY?)

Is it sufficient?
Static Co-sensitization and Delay

Static co-sensitization is **necessary** for a path to be responsible for the delay of a circuit

But NOT sufficient

Path of length 6 is statically co-sensitized
Delay of circuit is 5 (as observed earlier)

Summary

Static sensitization (SS) sufficient for true path, but not necessary

Static co-sensitization (SC) necessary for true path, but not sufficient
Timing Analysis for Embedded Software

Latency from reading sensor values to writing actuator commands is determined by execution time of `compute()`

```c
while(1) {
    read_sensors();
    compute();
    write_actuators();
}
```

This code is known to be terminating:
- loops with finite bounds
- no unbounded recursion

Typically:
- No interrupts/threads

Example of ComputationalTask

`altitude_control_task()` from implementation of software controller of “Paparazzi UAV”

```c
main.c:
... while(1) {
    ... periodic_task(...);
    ... }

switch(...) {
    case 0:
        altitude_control_task(...);
    ... }
```

```c
void altitude_control_task(void) {
    if (pprz_mode == PPRZ_MODE_AUTO) {
        if (vertical_mode == VERTICAL_MODE_AUTO_ALT) {
            /* inlined below: function altitude_pid_run(); */
            float err = estimator_z - desired_altitude;
            desired_climb = ppid_climb + altitude_gain * err;
            if (desired_climb < -CLIMB_MAX) {
                desired_climb = -CLIMB_MAX;
            }
            if (desired_climb > CLIMB_MAX) {
                desired_climb = CLIMB_MAX;
            }
        }
    } ...
```
Path Space in Program

```c
void altitude_control_task(void) {
    if (pprz_mode == PPRZ_MODE_AUTO2
         || pprz_mode == PPRZ_MODE_HOME) {
        if (vertical_mode == VERTICAL_MODE_AUTO_ALT) {
            /* inline below: function altitude_pid_run(); */
            float err = estimator_z - desired_altitude;
            desired_climb = prev_climb + altitude_pgain * err;
            if (desired_climb < -CLIMB_MAX)
                desired_climb = -CLIMB_MAX;
            if (desired_climb > CLIMB_MAX)
                desired_climb = CLIMB_MAX;
        }
    }
}
```

Must find:
Longest path in the control-flow graph (CFG)

Are All Program Paths Executable (Feasible) ?

```c
void altitude_control_task(void) {
    if (pprz_mode == PPRZ_MODE_AUTO2
         || pprz_mode == PPRZ_MODE_HOME) {
        if (vertical_mode == VERTICAL_MODE_AUTO_ALT) {
            /* inline below: function altitude_pid_run(); */
            float err = estimator_z - desired_altitude;
            desired_climb = prev_climb + altitude_pgain * err;
            if (desired_climb < -CLIMB_MAX)
                desired_climb = -CLIMB_MAX;
            if (desired_climb > CLIMB_MAX)
                desired_climb = CLIMB_MAX;
        }
    }
}
```

Determining if a program path is feasible (assuming finite-precision bit-vector types) is a SAT problem.
Graph Problems → Mathematical Programming

Some optimization problems on graphs can be conveniently represented as mathematical programs (e.g. linear programs)

Linear Optimization Problems

Linear Program

max  \( c_1 x_1 + c_2 x_2 + \ldots + c_m x_m \)

subject to
\[
a_{i1} x_1 + a_{i2} x_2 + \ldots + a_{im} x_m \leq b_i \quad i=1,2,\ldots,n
\]
Linear Optimization Problems

Linear Program

\[ \text{max } c_1 x_1 + c_2 x_2 + \ldots + c_m x_m \]

subject to
\[ A x \leq b \]

- \( A \) is a \( n \times m \) matrix
- \( x \) is a \( m \times 1 \) vector (of variables)
- \( b \) is a \( n \times 1 \) vector

Longest Path as a Linear Optimization Problem

Linear Program for Longest path in a DAG

\( x_i = 1 \) iff ith edge in the DAG is on the path, 0 o.w.

\[ \text{max } w_1 x_1 + w_2 x_2 + \ldots + w_m x_m \]

subject to
\[ E x = (1, 0, 0, \ldots, 0, -1)^T \]
\[ 0 \leq x_j \leq 1 \quad j = 1, 2, \ldots, m \]

- \( E \) is the incidence matrix of the graph
- The * constraints encode the condition that a single path is taken from source to sink
Impact of Variability on Timing Analysis

Delays of gates and wires are a function of manufacturing process and environment conditions

Process parameters:
- Oxide thickness
- Dopant concentration
- Transistor width
- Interconnect height and width
- ...

Environment parameters:
- Temperature
- Supply voltage variation
- ...

Longest Path under Variability is a Non-Linear Optimization Problem

Linear Program for Longest path in a DAG

\[ x_i = 1 \text{ iff } \text{ith edge in the DAG is on the path}, \ 0 \ o.w. \]

\[ p_k \text{ is the value of the kth process/environmental parameter} \]

\[ (k=1,\ldots,K) \]

\[
\begin{align*}
\text{max} & \quad w_1 x_1 + w_2 x_2 + \ldots + w_m x_m \\
\text{where} & \quad w_i = \sum_k \alpha_k + \beta_{jk} p_k
\end{align*}
\]

subject to

\[
E x = (1, 0, 0, \ldots, 0, -1)^T \quad \text{........................ (\text{*})}
\]

\[ 0 \leq x_j \leq 1 \quad j = 1,2,\ldots,m \quad L_k \leq p_k \leq U_k, \ k=1,\ldots,K \]

\( E \) is the incidence matrix of the graph

The * constraints encode the condition that a single path is taken from source to sink