Algorithms Final Project Progress Report

May 24, 2016

Team

Group 24

- 劉典恆 b01901185
- 胡明衛 b01901133
- 許睿中 b01502119
- 張漢維 b01901181

Problem

2016 D. Static Timing Analysis

Find as many sensitizable paths as possible for combinational logic circuit using shortest time.

Progress

Documentation

- [x] Summarize problem requirement. (@dianhenglau)
- [ ] Summarize ideas of solving this problem. (@dianhenglau)
- [ ] Read (and summarize if it is useful) about SAT. (@dianhenglau)
- [ ] Summarize article provided by TA. (@raychunghsu)
- [ ] Add compile requirement. (@dianhenglau)

Classes

- [ ] Define interface of Gate. (@davidhu34)
- [ ] Define interface of Circuit. (@davidhu34)

- [ ] Define interface of hash_map. (@dianhenglau)

Functions

- [x] Write main function. (@dianhenglau)
- [ ] Write a function to parse verilog netlist file into circuit. (@davidhu34)
- [x] Write a function to output sensitizable paths according to requirement. (@davidhu34)
- [ ] Write a function to find all sensitizable paths and their corresponding input vector. (@raychunghsu)
- [ ] Write a function to verify whether a path is sensitizable path. This is for verification. (@raychunghsu)
- [ ] Write string hash functions. (@dianhenglau)

Other

- [ ] Add OpenMP library to support multithread. (@dianhenglau)

Ideas

How to Find All Paths

- Do depth first search from all input pins toward output pins.
- Add the path to path list every time you reach an output pin.

Time Complexity

- Time complexity of doing depth first search on a single input pin is $O(|E|)$ where $|E|$ is number of edges of entire circuit. This is the worst case.
- Time complexity of doing depth first search on all input pins is $O(|PI| * |E|)$ where $|PI|$ is number of input pins.

How to Find All Paths Within Constraint

- Do depth first search from all input pins toward output pins.
- While doing depth first search, calculate arrival time at each node. If arrival time is greater than or equal to time constraint, stop moving down this path.
- Calculate slack everytime you reach output pin. If slack is greater than or equal to slack constraint, do not add this path into path list.

Time Complexity

- Time complexity is same as depth first search, i.e. $O(|PI| * |E|)$.

How to Calculate Arrival Time of All Gates

- From input pins, do breadth first search toward output pins.

Time Complexity

- $O(|E|)$.

How to Calculate Value of All Points

Given an input vector and a set of nodes sorted according to each node’ s arrival time, how to find value of all nodes.

- For each node in the sorted set, get its input nodes’ value, then calculate its value.

Time Complexity

- Since each node need to access its input nodes, the overall time complexity should be $O(|E|)$.

How to Find Sensitizable Paths

Method 1 (Brute Force)

- Find all paths within constraint and collect them in a path list.
- Try all possible permutation of input vectors.
- For each input vector, assume circuit has become stable (ignoring delay), then find value of all points in the circuit.
- With arrival time and value at each point, we can know whether a path is sensitizable. For each path in path list, check whether it is sensitizable.

Time Complexity

- Find all paths within constaint: $O(|PI| * |E|)$.
- Try all possible permutation: $O(2^|PI|)$.
- Find value of all points for an input vector: $O(|E|)$.
- Check all paths in path list: $O(|P| * |p|)$ where $|P|$ is number of paths and $|p|$ is average number of node in a path.
- Overall time complexity is $O(|PI| * |E|) + O(2^|PI| * |E|) + O(|P| * |p|)$, and $O(2^|PI| * |E|)$ will probably dominate.

Note

- This method will find all sensitizable paths.

Method 2 (Less Brute Force)

- Calculate arrival time of all gates.
- Basically the idea is trace from output pins toward input pins. Try every possibility (condition) that make a path become a true path. Check whether our assumption has any contradiction.
- Monitor slack constraint and time constraint while tracing.
- Pseudo code:

TODO: Parallelize it; Monitor constraint; How to check for confliction?

```
sensitizable_paths = vector()
input_vecs = vector()

path = vector()

for po in output_pins
    trace(po.from)

trace(gate)
    path.add(pair(gate, gate.value))

    if gate.value == X
        assert(gate.type == PO)
        path.pop()
        gate.value = 0
        trace(gate)
        gate.value = 1
        trace(gate)
        gate.value = X

    else if gate.type == NAND
        # Try to make gate.from_a become a true path.

        ` start_code_block(basic_logic)
        if gate.from_a.arrival_time < gate.from_b.arrival_time
            if gate.from_a.value == X
                if gate.value == 1
                    gate.from_a.value = 0
                    trace(gate.from_a)
                    gate.from_a.value = X

       else if gate.from_a.arrival_time > gate.from_b.arrival_time
            if gate.from_b.value == X
                gate.from_b.value = 1
                gate.from_a.value = !gate.value
                trace(gate.from_a)
                gate.from_b.value = X
                gate.from_a.value = X

        else # Both of them have same arrival time.
            if gate.from_a.value == X
                if gate.value == 1
                    gate.from_a.value = 0
                    trace(gate.from_a)
                    gate.from_a.value = X

            if gate.from_b.value == X
                gate.from_b.value = 1
                gate.from_a.value = !gate.value
                trace(gate.from_a)
                gate.from_b.value = X
                gate.from_a.value = X
        ` end_code_block(basic_logic)

        # Try to make gate.from_b become a true path.

        ` print(swap("from_a", "from_b", basic_logic))

    else if gate.type == NOR
        # Try to make gate.from_a become a true path.

        ` print(swap("0", "1", basic_logic))
```

```
        # Try to make gate.from_b become a true path.

      ` print(swap("0", "1", swap("from_a", "from_b", basic_logic)))

    else if gate.type == NOT
        if gate.from.value == X
            gate.from.value = !gate.value
            trace(gate.from)
            gate.from.value = X

    else if gate.type == PI and no_conflict()
        sensitizable_paths.add(reverse(path))
        input_vec = vector()
        for pi in input_pins
            input_vec.add(pi.value)
        input_vecs.add(input_vec)

    else
        print("Error: Unknown gate type.\n")
        exit(1)

    path.pop_front()
```

Time Complexity

- Calculate arrival time: $O(|E|)$.
- Tracing: ?

Note

- This method find all sensitizable paths too.
- To adapt parallel execution, every thread must have their own copy of `gate.value`. Other gate attribute can be shared.