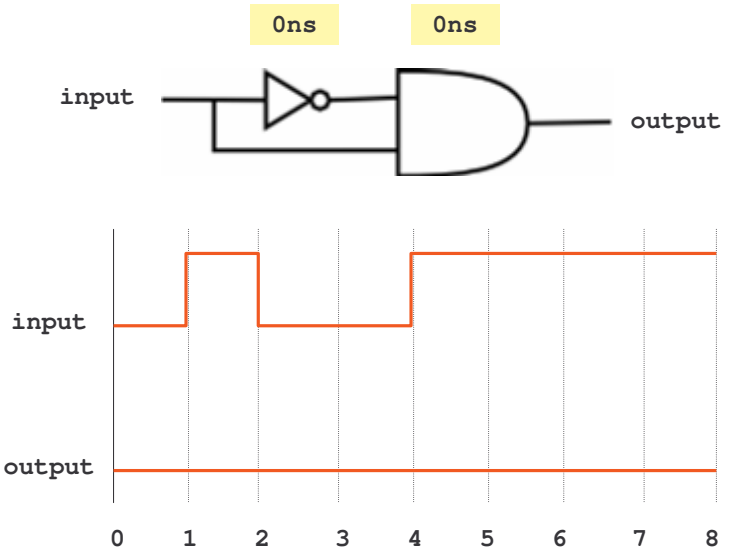
2110211 โครงสร้างข้อมูลเบื้องต้น

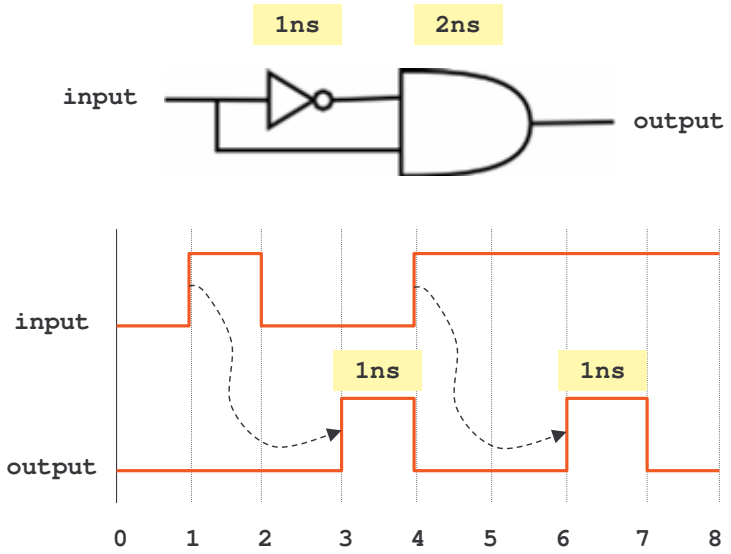
Logic Simulator

สมชาย ประสิทธิ์จูตระกูล

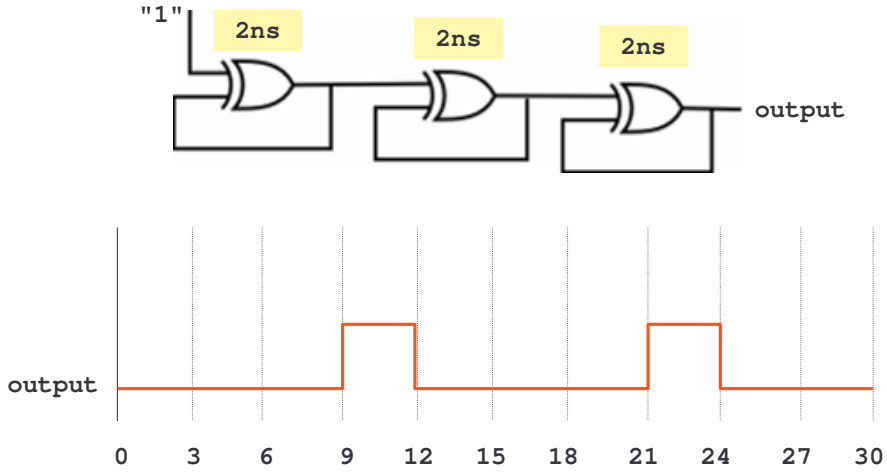
Logic Simulation : ง่ายเมื่อไม่มี gate delay



Logic Simulation : ยุ่งเมื่อมี gate delay



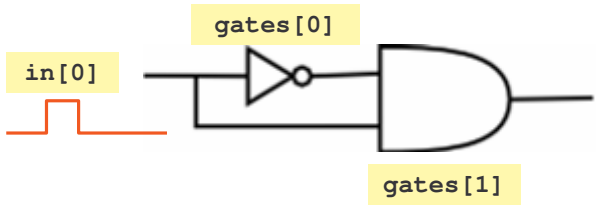
อีกตัวอย่าง



มาเขียน Logic Simulator (แบบง่าย)

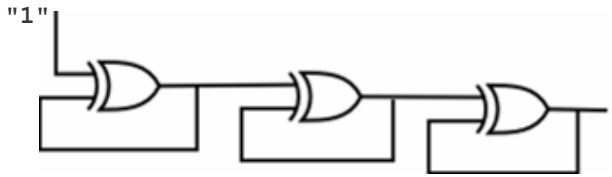
- การสร้างวงจร
- classes ที่ใช้โปรแกรม
  - Gate : AND, OR, NOT, NAND, NOR, XOR, INPUT
  - Event
  - LogicSim
- การใช้ List, PriorityQueue

ตัวอย่างการสร้างวงจร



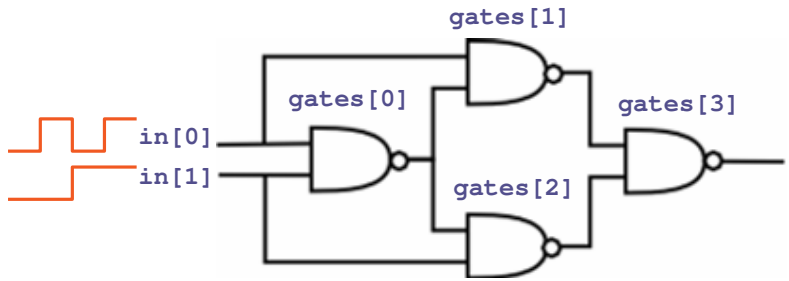

```
InPort[] in = InPort.createInputs(1,
              new boolean[][] {false,true,false}});
Gate[] gates = new Gate[2];
gates[0] = new NOT("not", in[0]);
gates[1] = new AND("and", in[0], gates[0]);
```


ตัวอย่างการสร้างวงจร



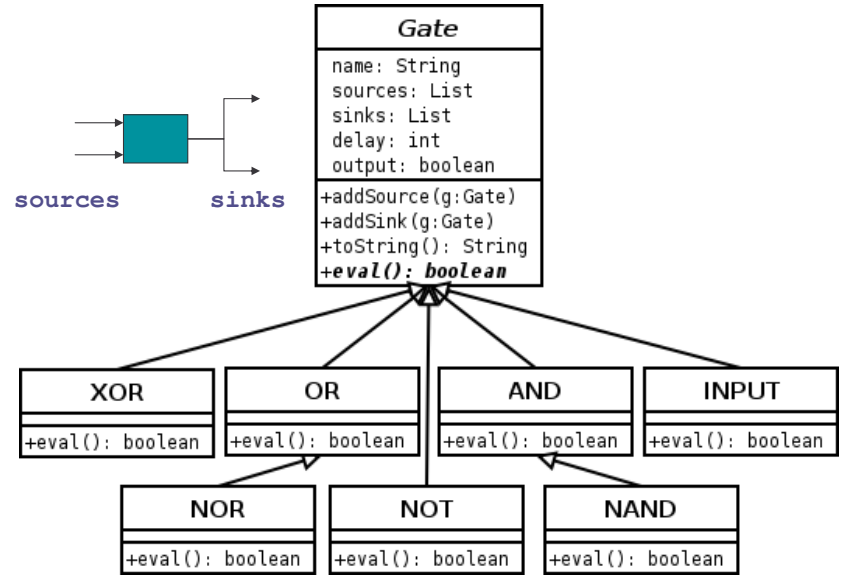

```
InPorts[] in = InPorts.createInputs(1,
               new boolean[][] {{true}});
Gate[] gates = new Gate[3];
gates[0] = new XOR("x0", in[0]);
gates[0].addSource(gates[0]);
gates[1] = new XOR("x1", gates[0]);
gates[1].addSource(gates[1]);
gates[2] = new XOR("x2", gates[1]);
gates[2].addSource(gates[2]);
```


ตัวอย่างการสร้างวงจร



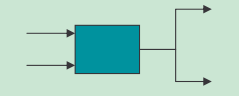

```
InPort[] in = InPort.createInputs(9,
              new boolean[][] {{false,true,false,true,},
                               {false,false,true,true}});
Gate[] gates = new Gate[4];
gates[0] = new NAND("n0", in[0], in[1]);
gates[1] = new NAND("n1", in[0], gates[0]);
gates[2] = new NAND("n2", in[1], gates[0]);
gates[3] = new NAND("n3", gates[1], gates[2]);
```

class Gate



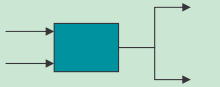
class Gate

```
public abstract class Gate {
  String name = "";
  List sources = new ArrayList();
  List sinks = new ArrayList();
  int delay = 0;
  boolean output = false;
  public abstract boolean eval();
  public Gate(String name) {
    this(name, new Gate[0]);
  }
  public Gate(String name, Gate...gates) {
    this.name = name;
    for (Gate g : gates) {
      sources.add(g);
      g.addSink(this);
    }
  }
  ...
```



class Gate

```
  public final void addSource(Gate g) {
    sources.add(g);
    g.addSink(this);
  }
  public final void addSink(Gate g){
    sinks.add(g);
  }
  public String toString() {
    String c = getClass().getName();
    c = c.substring(c.lastIndexOf(".")+1);
    String s = c + ":" + name + "(";
    String p = "";
    for(int k=0; k<inputs.size(); k++) {
      p = p + ((Gate)inputs.get(k)).name + ",";
    }
    if (p != "") p = p.substring(0,p.length()-1);
    s += p + ") = " + (output?"T":"F");
    return s;
  }
}
```

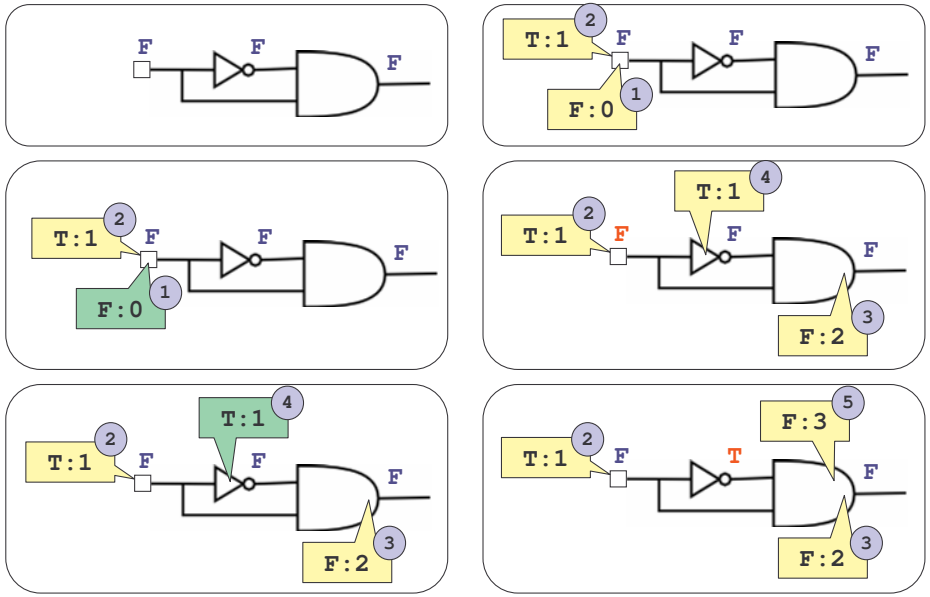


class AND, NAND

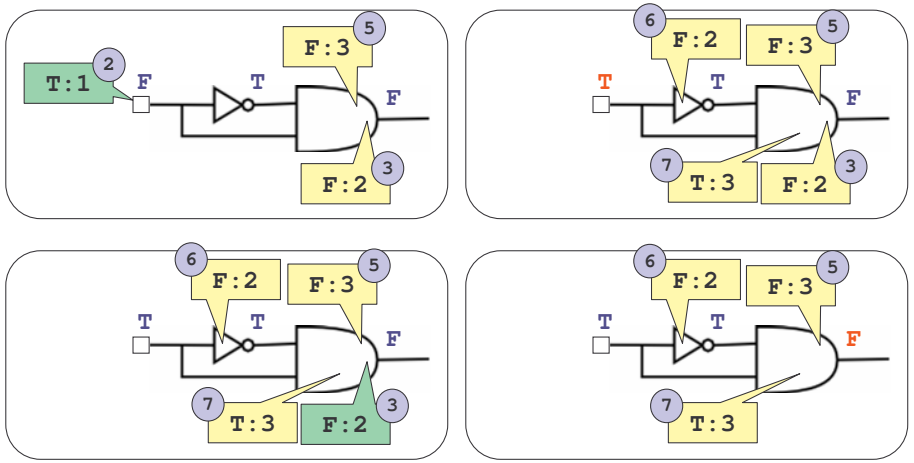
```
public class AND extends Gate {
  public AND(String name, Gate... gates) {
    super(name, gates);
    delay = 2;
  }
  public boolean eval() {
    boolean result = true;
    for(int k=0; k<sources.size(); k++) {
      result &= ((Gate)sources.get(k)).output;
    }
    return result;
  }
}
```

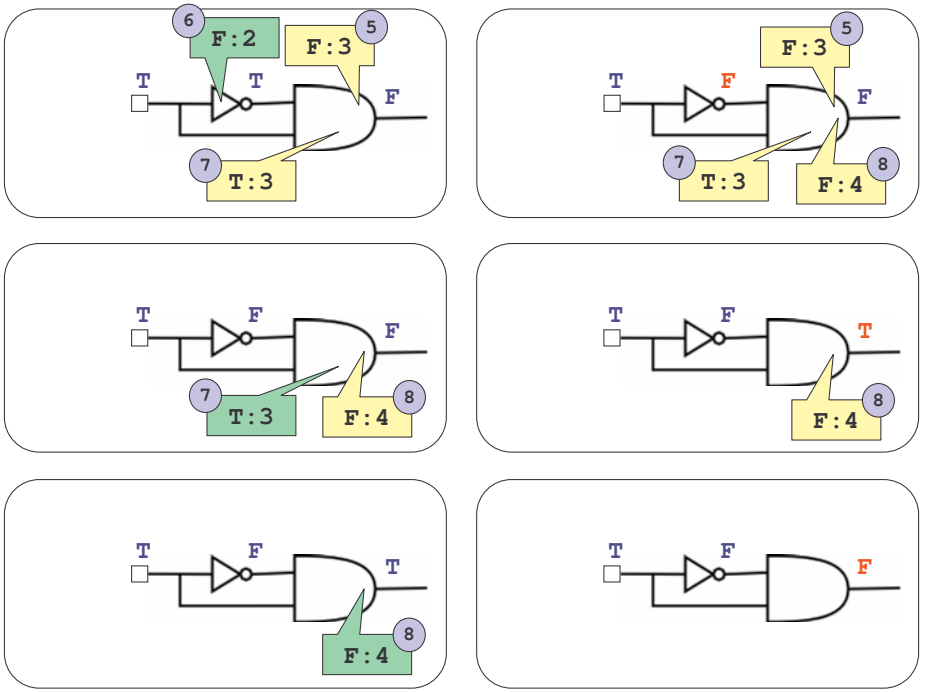
```
public class NAND extends AND {
  public NAND(String name, Gate... gates) {
    super(name, gates);
    delay = 2;
  }
  public boolean eval() {
    return !super.eval();
  }
}
```

Event-Driven Simulation



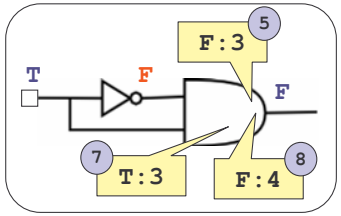
Event-Driven Simulation





Event-Driven Simulation

- ต้องมี Event objects ซึ่งเก็บ
  - gate ที่ input ได้รับการตั้งค่า และจะมี output ใหม่
  - ค่าของ output ใหม่
  - เวลาที่ค่าใหม่จะไปปรากฏที่ output ของ gate
  - หมายเลข event
- มี priority queue เก็บ Event objects
  - จัด priority ตาม [เวลา, ชื่อ gate, event ID]



```
class Event implements Comparable {
  private static int idCounter = 0;

  final int time;
  final Gate gate;
  final int uniqueID;
  final boolean output;

  Event(Gate gate, int time) {
    this.gate = gate;
    this.time = time;
    uniqueID = idCounter++;
    output = gate.eval();
  }
  public int compareTo(Object o) {
    Event that = (Event) o;
    int cmp = this.time - ((Event) o).time;
    if (cmp != 0) return cmp;
    cmp =  this.gate.name.compareTo(that.gate.name);
    if (cmp != 0) return cmp;
    return this.uniqueID - that.uniqueID;
  }
  ...
```

ใช้สาม `fields` นี้รวมกันเพื่อกำหนด `priority` ของ `event`

Logic Simulator

```
public static void simulate(InPort[] in, Gate[] gates,
                            Gate observed, int duration) {
  PriorityQueue events = new BinaryHeap();
  initializeInputEvents(events, in);
  while ( !events.isEmpty() ) {
    Event e = (Event) events.dequeue();
    if ( e.time > duration ) break;
    while(!events.isEmpty() && e.equals(events.peek())) {
      e = (Event) events.dequeue();
    }
    if ( observed == null || e.gate == observed )
      System.out.println(e);
    e.gate.output = e.output;
    List sinks = e.gate.sinks;
    for(int i = 0; i < sinks.size(); i++) {
      Gate gate = (Gate) sinks.get(i);
      events.enqueue(new Event(gate, e.time + gate.delay));
    }
  }
}
```



```
public class InPort extends Gate {
  private int k = 0;
  boolean[] stream;
  int duration;
  public InPort(String name, int duration, boolean[] stream) {
    super(name, new Gate[0]);
    this.stream = stream;
    this.duration = duration;
    delay = 0;
  }
  public boolean eval() {
    if (k >= stream.length) return output;
    return stream[k++];
  }
  public static InPort[] createInputs(int duration,
                                      boolean[][] streams) {
    InPort[] in = new InPort[streams.length];
    for (int i = 0; i < streams.length; i++) {
      in[i] = new INPUT("IN" + i, duration, streams[i]);
    }
    return in;
  }
}
```



```
public class LogicSimulator {
 public static void main(String[] args) {
    InPort[] in = InPort.createInputs(1,
        new boolean[][] {{false,true,false}});
    Gate[] gates = new Gate[2];
    gates[0] = new NOT("not", in[0]);
    gates[1] = new AND("and", in[0], gates[0]);
    simulate(in, gates, gates[1], 40);
 }
 public static void simulate(InPort[] in, Gate[] gates,
                             Gate observed, int duration) {
    ...
 }
 private static void initializeInputEvents(
                     PriorityQueue events, InPort[] in) {
    for(int i=0; i<in.length; i++) {
      for(int j=0; j<in[0].stream.length; j++) {
        events.enqueue(new Event(in[i], j*in[i].duration));
      }
    }
 }
}
```

การบ้าน

- ทำความเข้าใจคลาสต่าง ๆ และหลักการทำงานของ logic simulator
- สูดหายใจลึก ๆ
- ใช้เวลาสักสามวัน เขียน logic simulator เองโดยไม่ต้องดู sheet นี้
- เพิ่มคลาส NOR, OR, NOT, XOR ด้วย