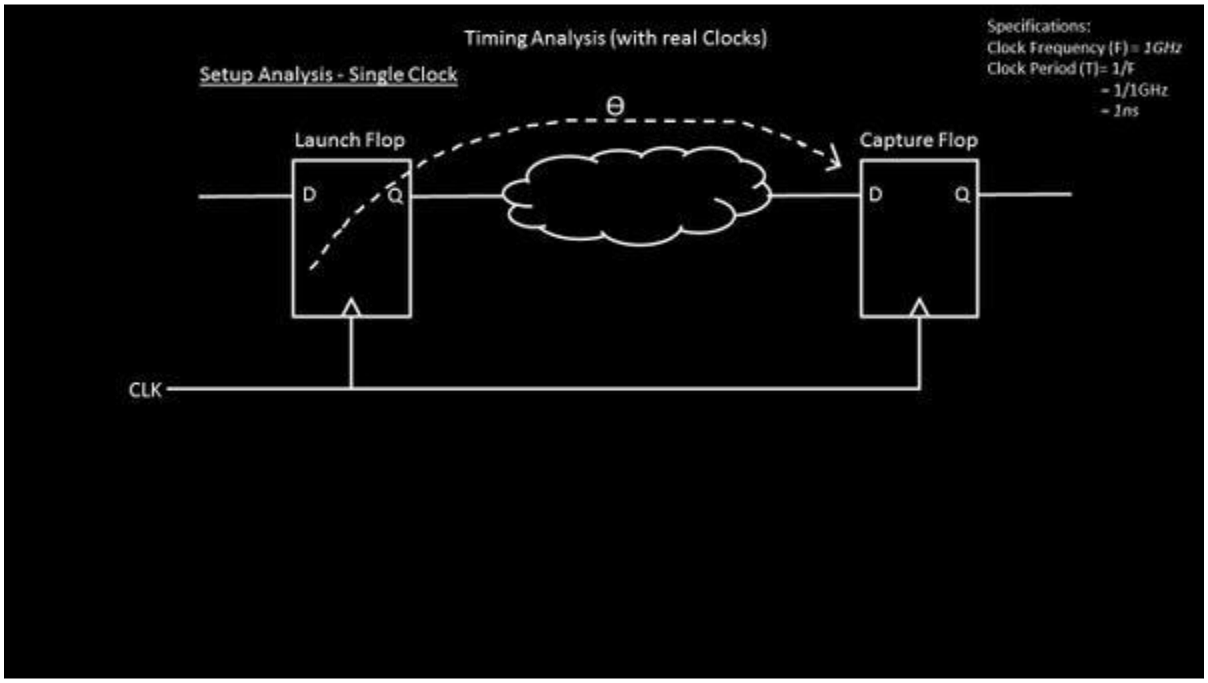
First things first – Timing graph

Kunal Ghosh

This is like the **‘Batman’ of static timing analysis**. We have heard stories about it but never seen it. The reasons, this comes more from an algorithm point of view.

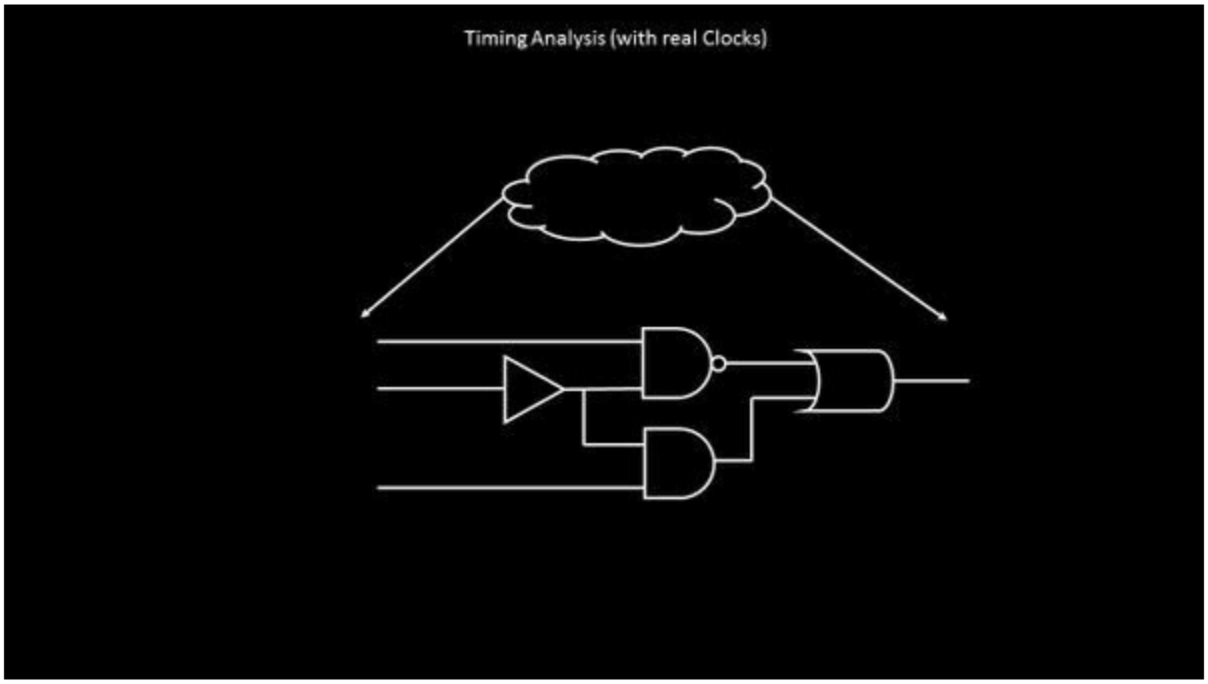
How about looking a level higher and a level deeper into below timing path?

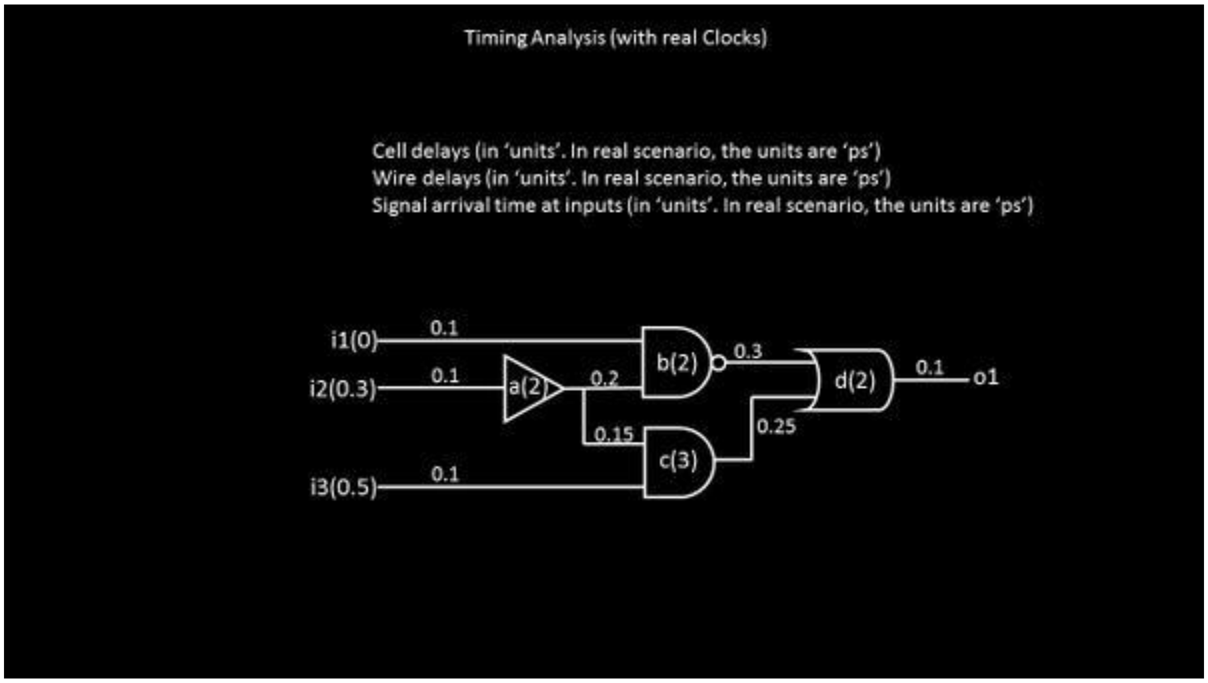

Timing Analysis (with real Clocks)
Setup Analysis - Single Clock
Specifications:
Clock Frequency (F) = 1GHz
Clock Period (T)= 1/F
= 1/1GHz
= 1ns
θ
Launch Flop
Capture Flop
D
Q
D
Q
CLK

I will take an example of combinational logic and will convert or represent the combinational logic as **Directed Acyclic Graph** (DAG) as below. Now DAG is something related more to software, and that’s not our intention here. So, I will keep that aside for some time now and focus on **actual arrival time** (AAT) and **required arrival time** (RAT).

I am sure you have heard of arrival time and required time. AAT and RAT are of the same class but of a different level and commonly used to denote node times. Stay with me and I will come to that in a moment

First, let’s add cell delays, wire delays and input delays to the combinational logic.

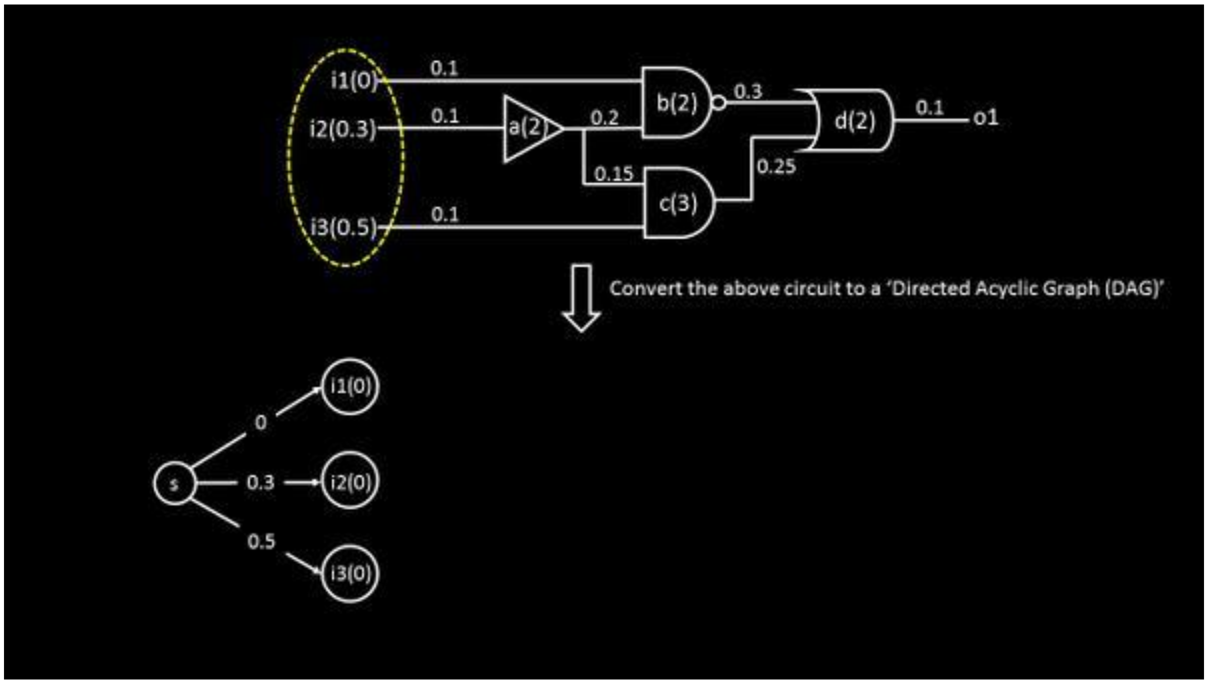

Timing Analysis (with real Clocks)


Timing Analysis (with real Clocks)

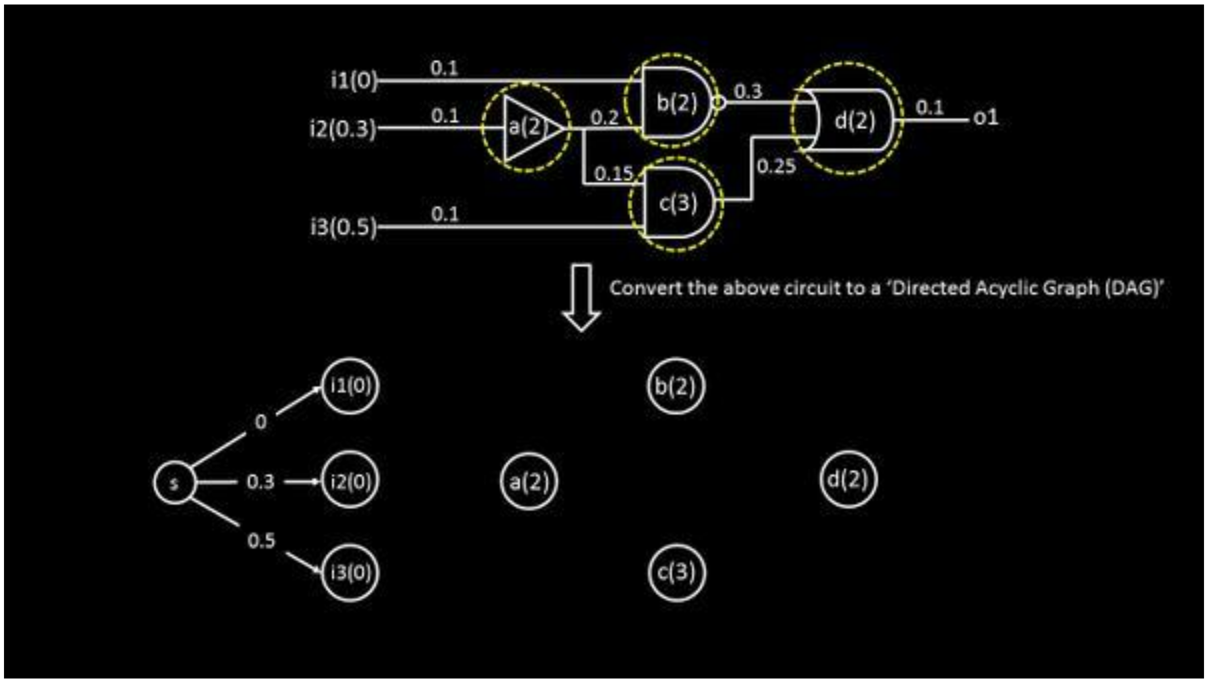
Cell delays (in 'units'. In real scenario, the units are 'ps')
Wire delays (in 'units'. In real scenario, the units are 'ps')
Signal arrival time at inputs (in 'units'. In real scenario, the units are 'ps')

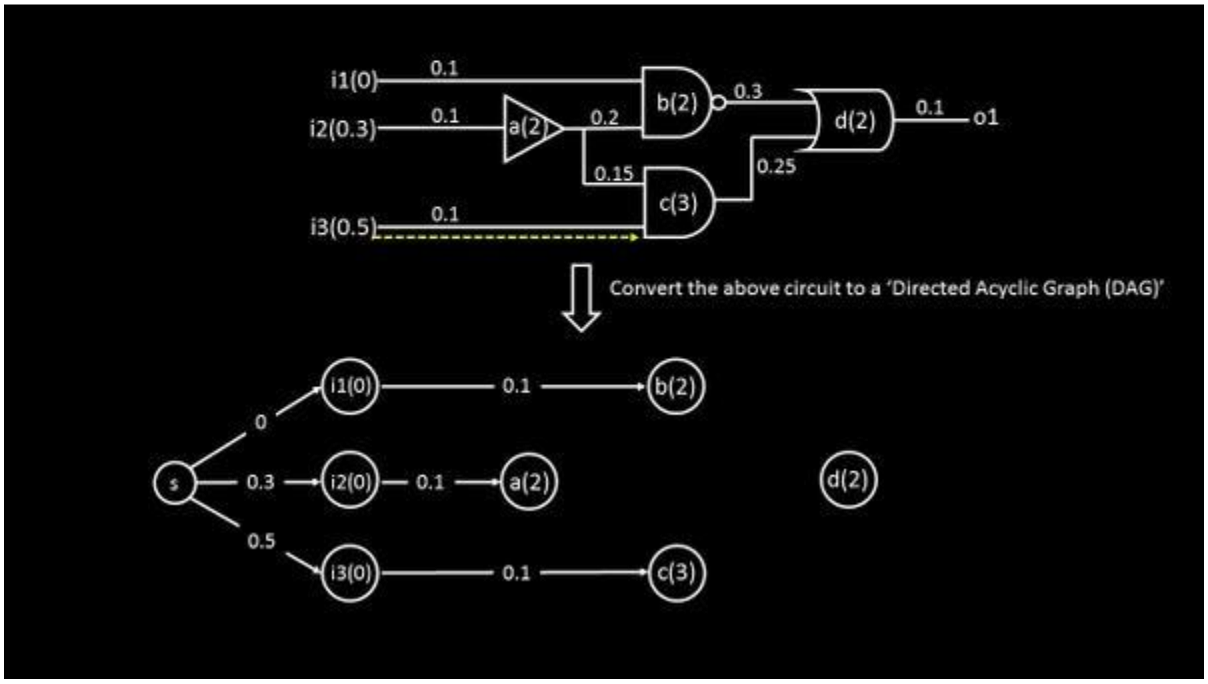
In the above image, a (2) says cell 'a' has a delay of 2 units, wire delay from (say) i2 to input of cell 'a' is 0.1 units and i2(0.3) represents the time (after launch clock edge) when signal transition occurs. In this case, signal transition occurs at 0.3 time units after launch clock edge.

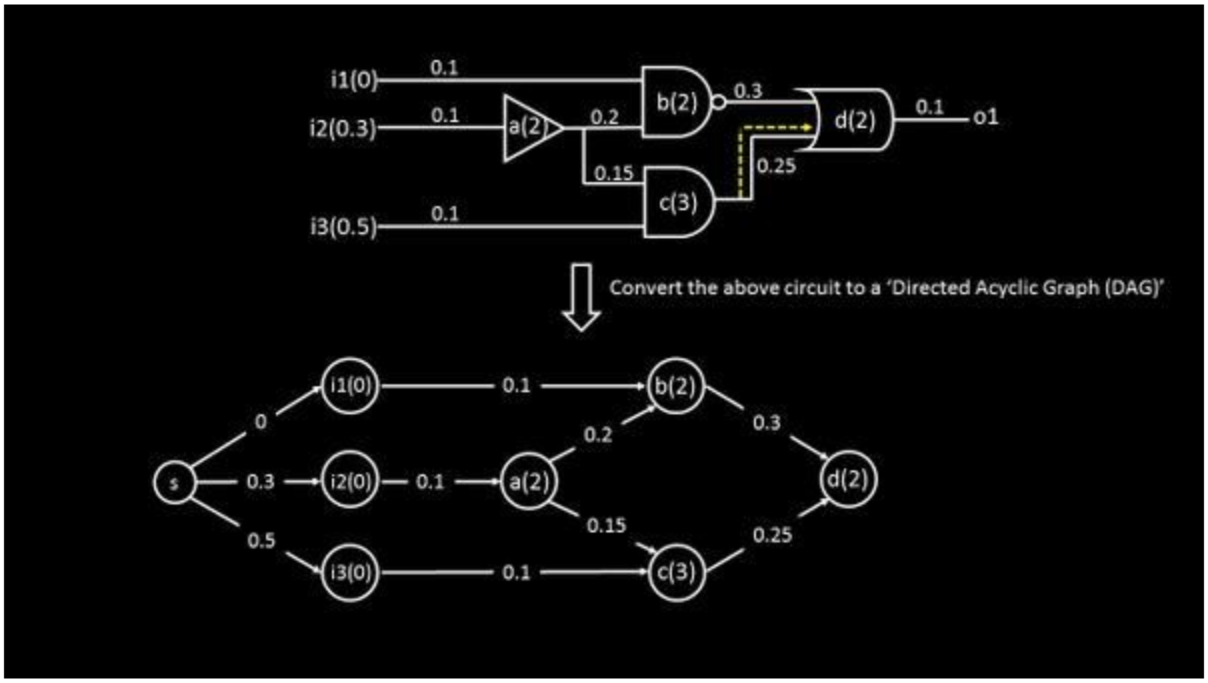
Now converting the above to DAG takes some amount of work. We use a source node for each input pin like below



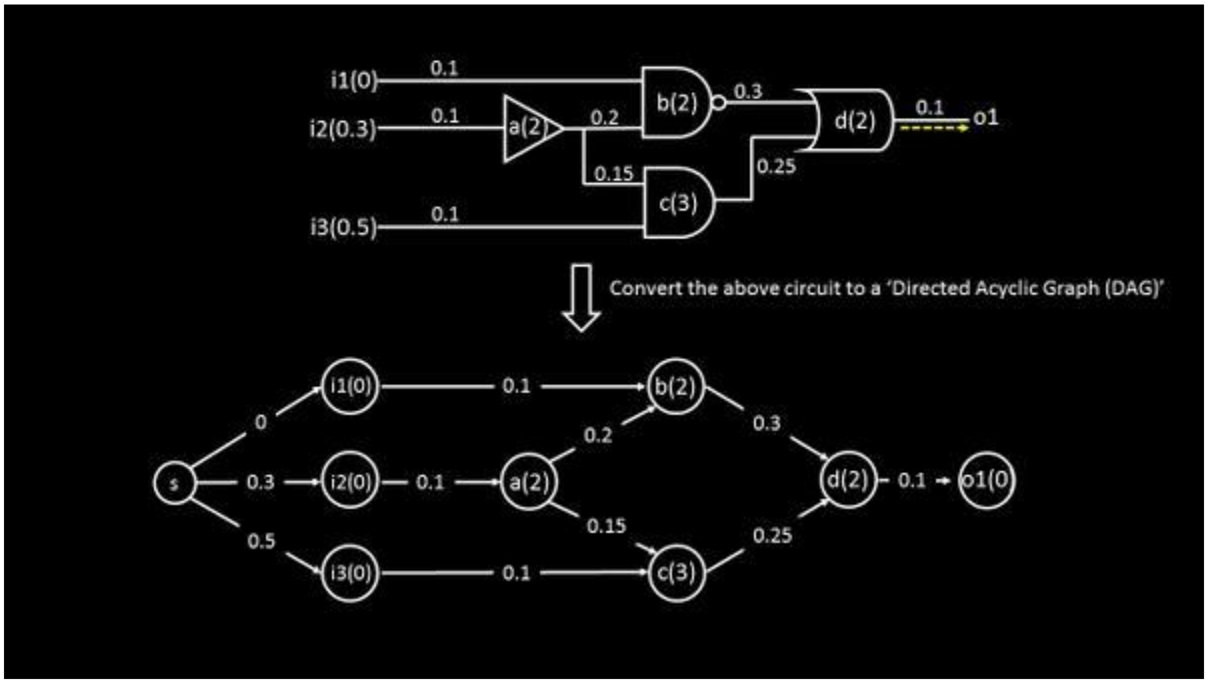
The cells are converted to nodes with their delay and cell names shown inside and **the wire delays are represented as directed arrow**s like below







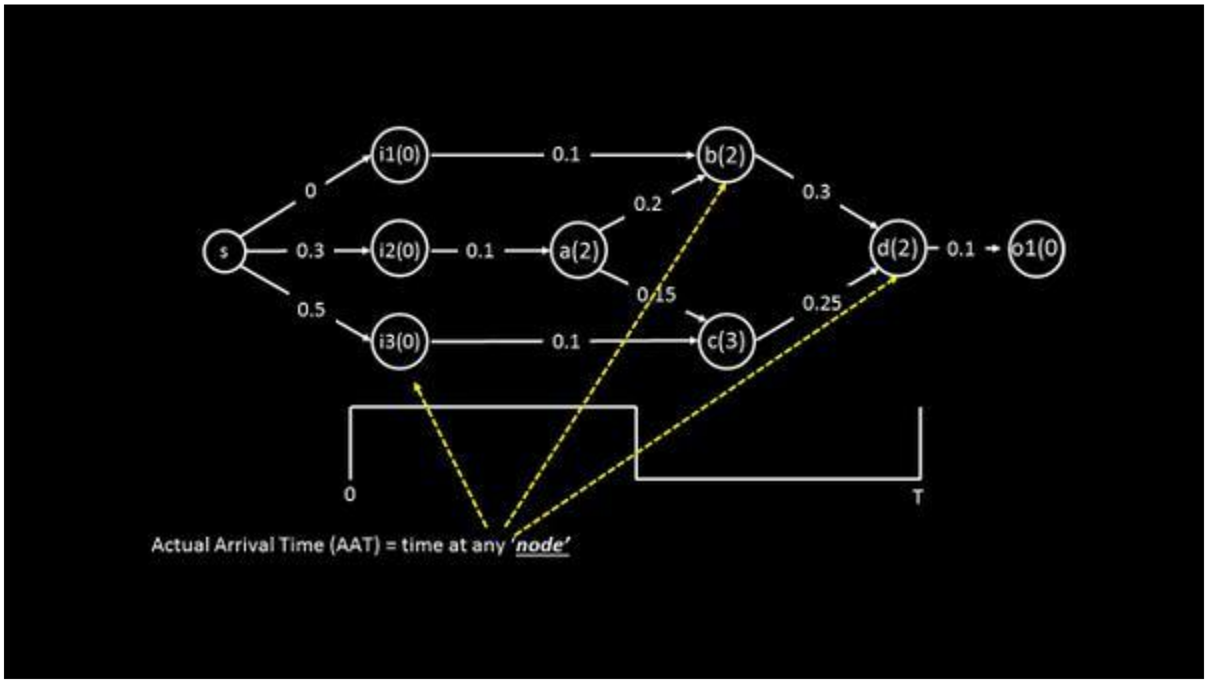
For the output pin, we use a node, and represent the delay from output of cell 'd' to input of 'output' node as below

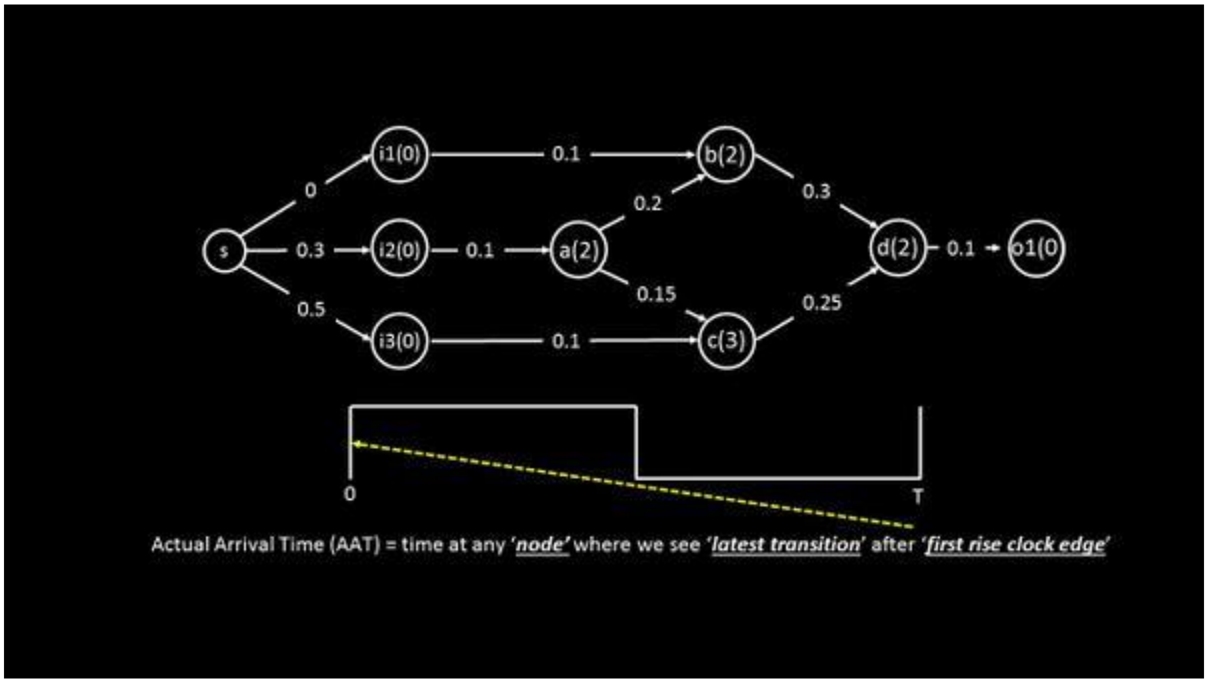


Above is called the 'graph'. I still should introduce 'timing' to it, to call this one as a '**timing graph**'. So, we found out 30% of 'timing graph' hidden story and I can promise you that things become interesting when we discover the rest 70%.

Now that we know what a timing graph is, let me unveil **actual arrival time (AAT), required arrival time (RAT) and slack.** We have seen these terms in a timing report, but what I will be talking about in this post is more from an algorithm viewpoint. Stay with me!!

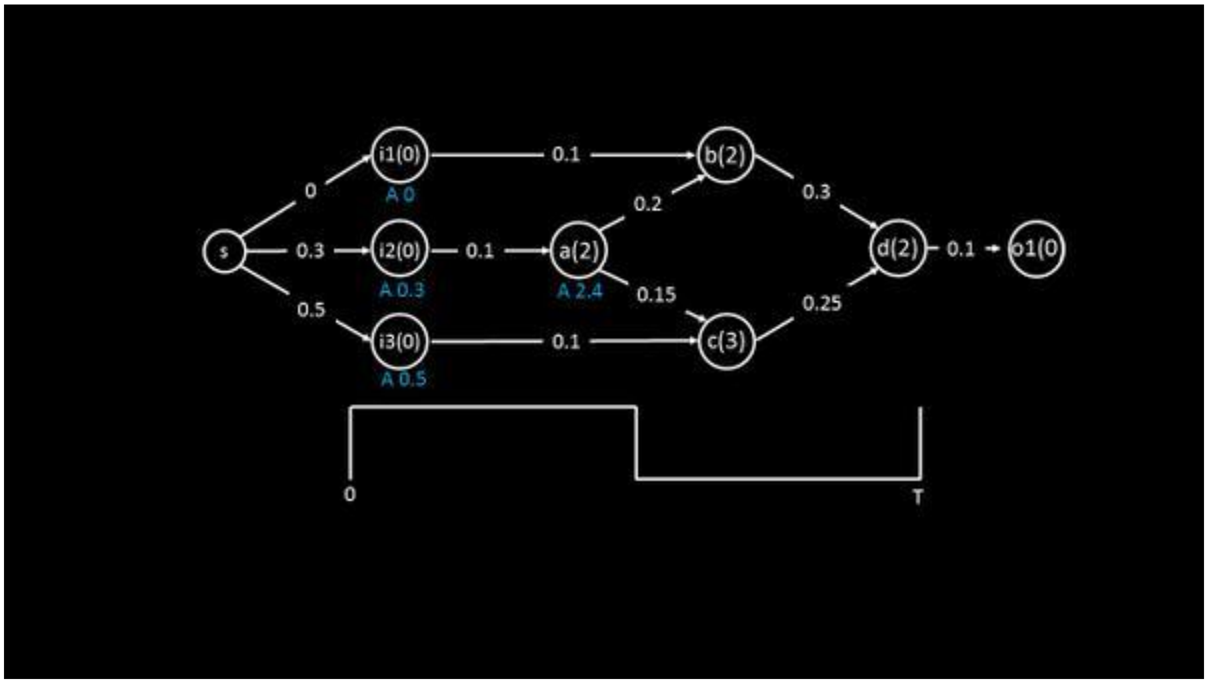
Actual arrival time (AAT) is the time at any node where we see latest transition after first rise clock edge



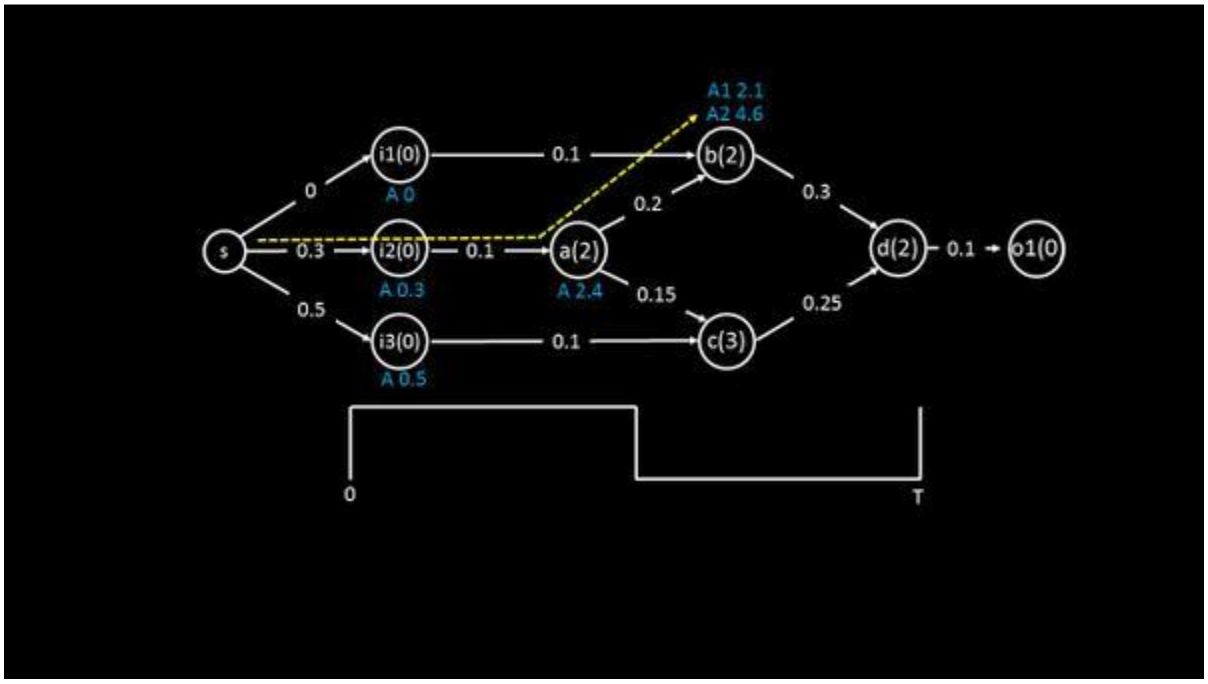

s
0
0.3
0.5
i1(0)
i2(0)
i3(0)
0.1
0.1
0.1
a(2)
0.2
0.15
b(2)
c(3)
0.3
0.25
d(2)
0.1
o1(0)
0
T
Actual Arrival Time (AAT) = time at any 'node' where we see 'latest transition' after 'first rise clock edge'

With that definition, if we calculate the arrival times at nodes i1, i2, and i3, they will be 0, 0.3, and 0.5 units respectively. Let us represent actual arrival time by 'A' as shown below. The **AAT at node 'a' will be AAT(i2) + wire delay which is 0.1 + the delay of node 'a' which is 2.**

So, the AAT(a) = 0.3 + 0.1 + 2 = 2.4 as shown below

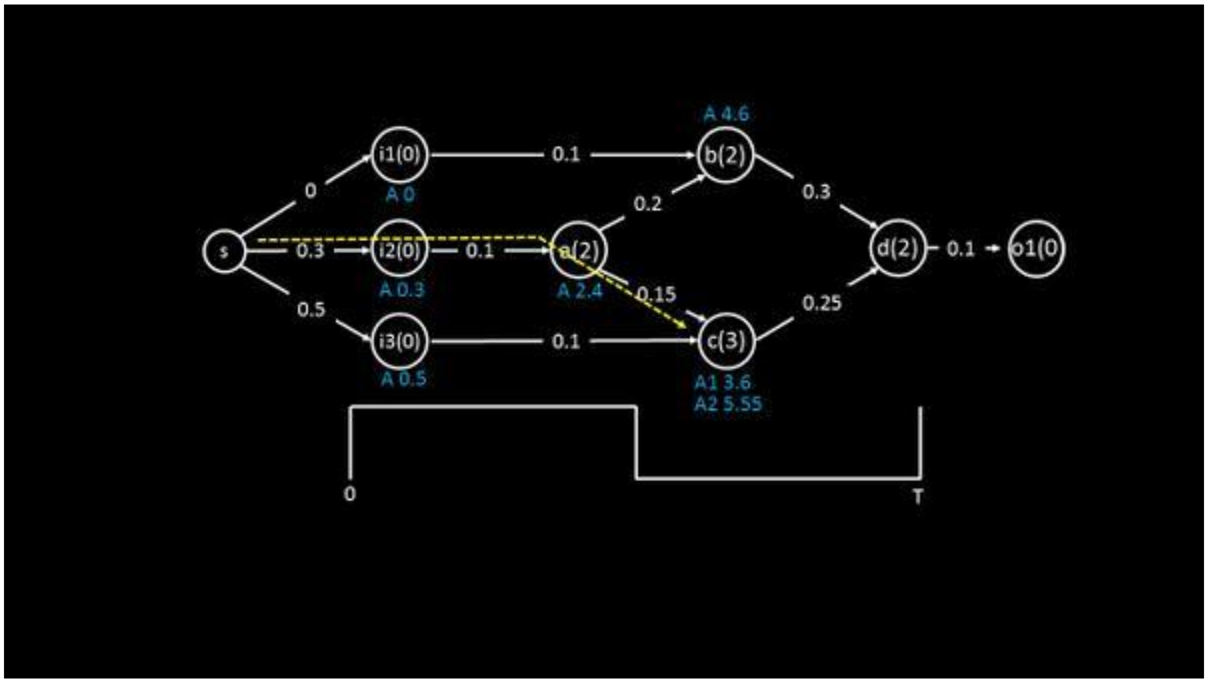


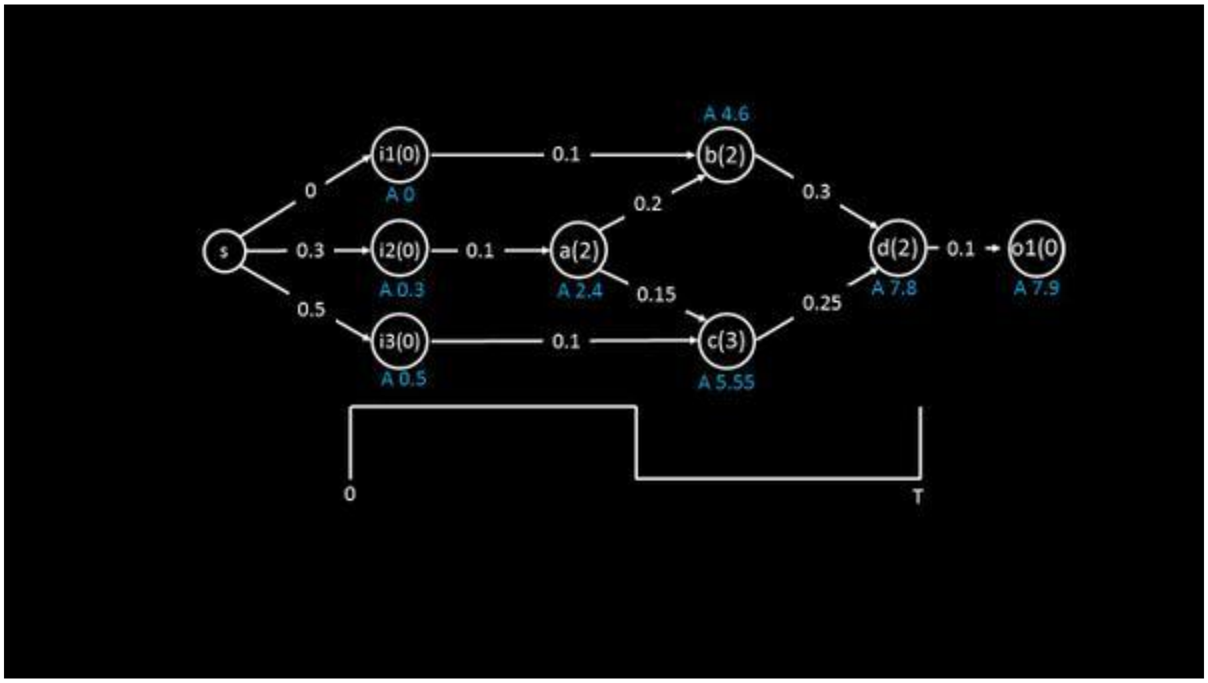
The **arrival time computation** was still simple till this point. The trick comes when we are required to calculate the AAT at node 'b', as there are two incoming paths to it, one through node i1 and other through node i2, and hence two arrival times A1 and A2 as shown below



The rule says we need **to go for a worst-case analysis for setup timing and hence, even in this case we select the worst arrival time** which A2 = A = 4.6. The reason is, if in real scenario, the signal takes A2 path and if, while computation of AAT, we take A1 path (which is through i1), we might be a bit optimistic.

Same explanation goes for node ‘c’ and ‘d’ as well and is shown below





Now that **we have a complete picture of arrival times at every node**, the next step is get the **required arrival time and each node and finally the slack at each node.**

Remember, I had promised in my first post, that I will make this interesting for you. Well, I think I have done it, so far. And things become exciting when I take you to next level of computation. Learning static timing analysis will be never the same and that I promise.