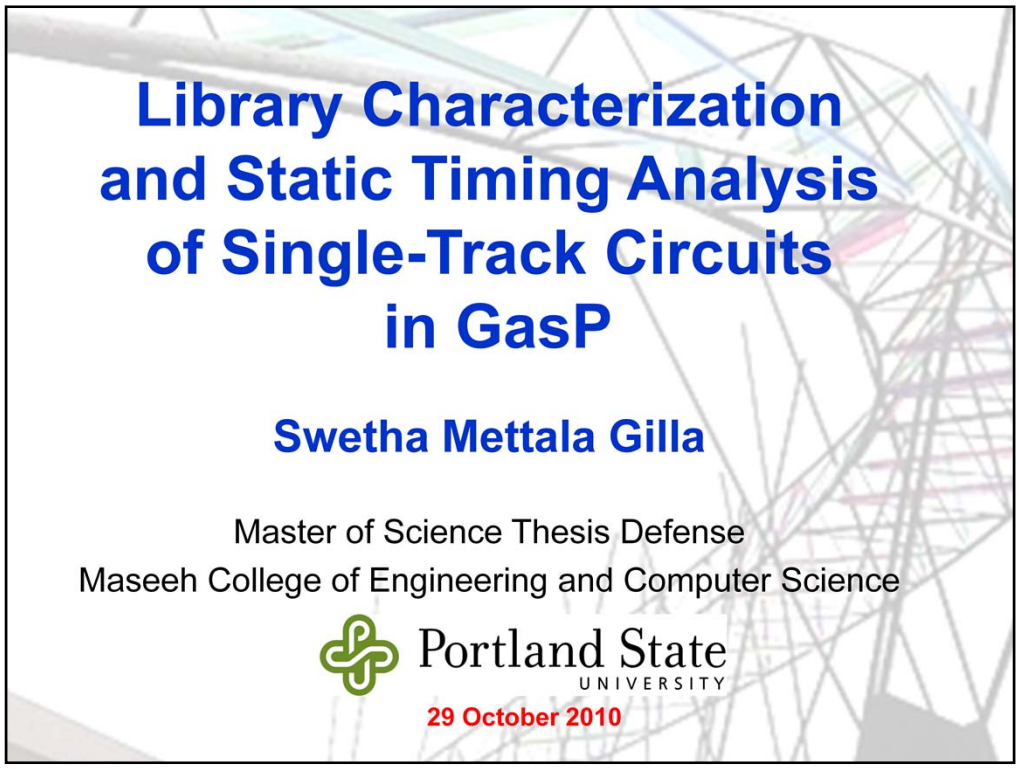


Good afternoon!

- My name is Swetha Mettala Gilla – you can call me Swetha.

I'm a student at the Electrical and Computer Engineering Department and at the Asynchronous Research Center.

This talk is about the work I did for my Master's thesis.

- It is about timing validation of a very fast asynchronous circuit family.
- The circuit family is called GasP.

Acknowledgements

Thesis Committee

- Dr. Xiaoyu Song (chair) (ECE)
- Dr. Douglas V. Hall (ECE)
- Dr. Christof Teuscher (ECE)
- Marly Roncken (CS, ARC)

Special Thanks

- Ivan Sutherland (ECE, ARC)

Before I move on to the technical part of my talk,
I would like to thank my thesis committee.

I like to thank especially Prof. Song, Marly, and Ivan.
It was a wonderful experience to work with you.

Introduction: GasP

Fast Asynchronous Symmetric Pulse Protocol

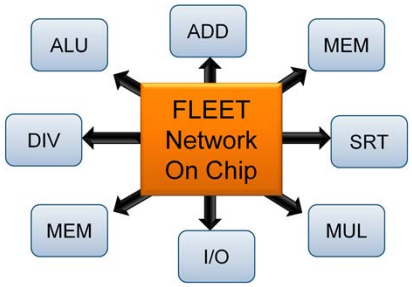
- Uses Single-Track Handshake signaling
- Light-weight in area and in power
- Flexible elastic communication

Flexibility + High Speed + Low Power + Low Area

- Makes an excellent circuit family for on-chip communication

More details on:

- http://arc.cecs.pdx.edu



Important for Multi-core and Parallel systems

OK, here comes the technical part.
I will start with an introduction to GasP.

GasP circuits are asynchronous.
They communicate by handshake signaling over single-track wires.
I will explain more about this in the following slides.

What's important about single-track communication is
that it results in light weight circuits - light in area, light in power.

If you add the flexibility of asynchronous design to high speed, low power, and low area,
you get an excellent circuit family for on-chip communication.

And that's an important application domain
for the multi core and parallel systems that are built these days.

I will not really talk about parallel systems.
All I want to say HERE is that the Asynchronous Research Center is building one.
It's called FLEET.
(CLICK)

Here is a picture of FLEET,
- showing a GasP network-on-chip connecting various computation blocks

Such as:
- low-power adders and multipliers, high-speed adders and multipliers
- I/O functions
- and memory blocks

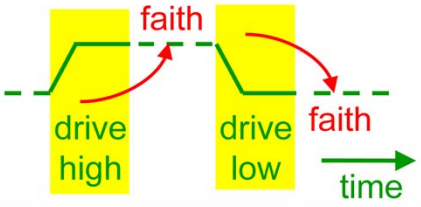
The computation blocks can be synchronous or asynchronous.
They can even be GasP circuits.

The Gasp circuits used for computation are a bit more complex
- than the GasP circuits in my thesis,
- but they use the same single-track handshake communication.

Introduction (2): Single-Track Handshaking

- Bi-directional 1-wire communication
- Request=high, Acknowledge=low
- 2-phase return-to-zero handshake

- Wire is driven briefly
- $Wire_{end1}$ starts/stops up-transition
- $Wire_{end2}$ starts/stops down-transition



requires FAITH
in engineering and tools

ASSUMPTION-1:
- The brief drive is long enough to get the transition to the other end

ASSUMPTION-2:
- Voltage observed at the near-end reflects the voltage at the far-end

This slide explains what single-track handshaking is.

(CLICK)
• it's a bi-directional communication using a single wire.
• A high voltage level on the wire indicates a request signal,
• and a low voltage level on the wire indicates an acknowledge signal.

The wire has two drivers – one at each end.
• One for the request, one for the acknowledge.
• This leads naturally to a 2-phase return-to-zero handshake.
(POINT TO the UP and DOWN transition in the PICTURE)

(WAIT)

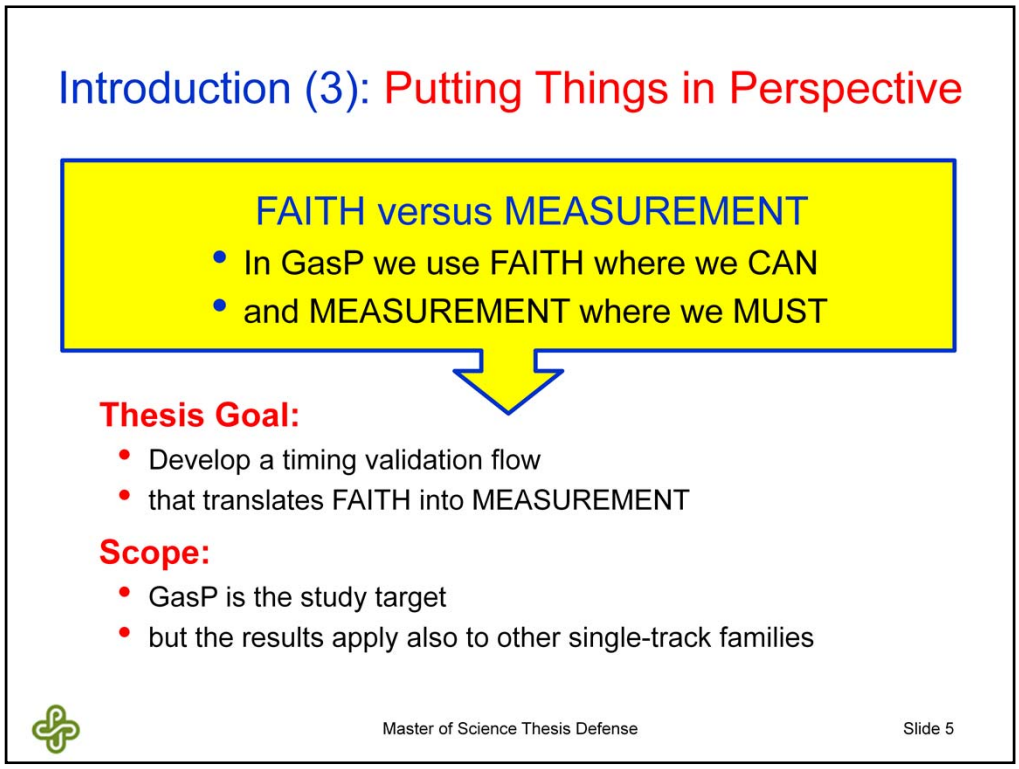
The wire is shared, so each driver must drive ONLY briefly,
• as indicated by the high-lighted areas in the picture.
• The brief drive avoids a drive fight between the two ends.

In addition, each driver responds only to changes at ITS OWN end.
• The local response completes the communication in two phases.

To make this work, we need some faith:
(CLICK)

Faith in our own engineering and in the design tools that we use.

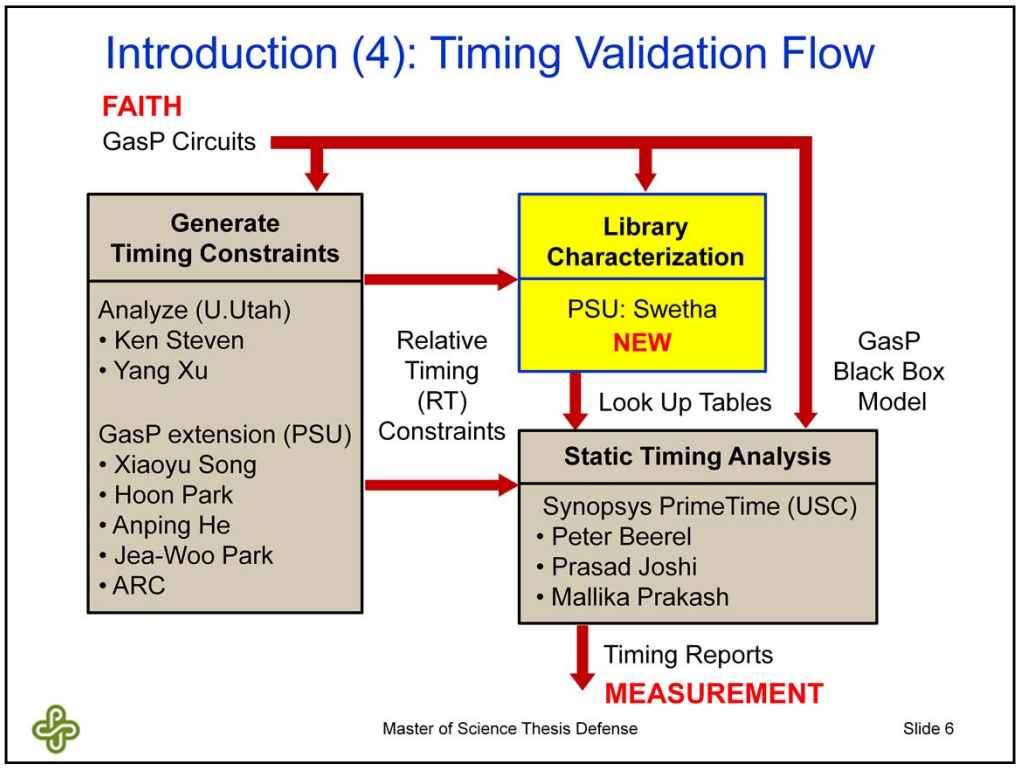
Our faith in a single-track handshake depends on two assumptions:
• Assumption 1 is that the brief drive is long enough to traverse the wire.
• Assumption 2 is that the voltage observed at the NEAR-END of the wire
• reflects the voltage at the FAR-END.



The design of GasP circuits is light-weight,
because we GasP designers use FAITH where we CAN and MEASUREMENT where we MUST.

And this is where my Thesis comes in.
(CLICK)

I developed a timing validation flow that translates FAITH into MEASUREMENT,
• so a GasP designer can rely on FAITH to make her design light-weight
• and use my tool to confirm that faith.



Here is an overview of the Timing Validation Flow I'm aiming for.
(CLICK)
Ideally, the first step is to take a GasP circuit and generate its critical timing paths.
There is a tool for this step, developed at the University of Utah by Professor Ken Stevens and his students.
- The tool is called Analyze.

It's being extended for GasP circuits here at PSU
- by Prof. Song and his students, in collaboration with the Asynchronous Research Center.

The tool uses formal verification techniques.
The critical timing paths are called relative timing constraints.

In my thesis I assume that this step is done.
I start with a GasP circuit and its Relative Timing Constraints.

(CLICK)
The second step in the flow, and the first step in my thesis is called Library Characterization.
This is the key part of my thesis, and it's all new work.
In this step:
- I take a GasP circuit and its relative timing constraints,.
- I simulate these for various operating conditions,
- and generate a collection of Look Up Tables ,
- that store the timing information of the paths under those conditions.

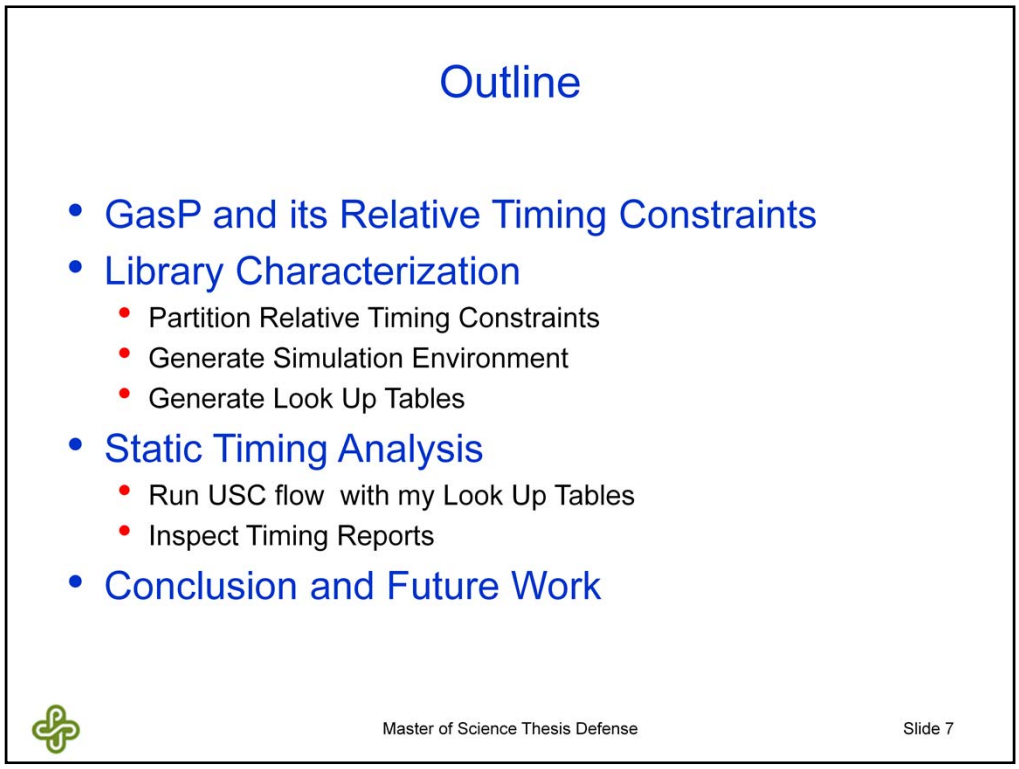
(CLICK)
I feed these Look Up Tables with a Black Box model of the GasP circuit into Static Timing Analysis.
The Black Box shows only the circuit input and output pins,
and hides the combinational loops and single-track details of GasP.

(PAUSE)
This step is not new.
- The Black Box model and the analysis commands for the relative timing constraints
- were developed at the University of Southern California
- by Prof. Peter Beerel and his students.
- The tool itself is a commercial tool by Synopsys; It's called PrimeTime.

All I had to do was
- get it running at PSU, and import my constraints and Look Up Tables

The result of Static Timing Analysis is a collection of timing reports,
- one for each Relative Timing Constraint ,
- showing how much slack there is to make or break the constraint.

(CLICK
So, in total, these three pieces take a GasP circuit, built using FAITH
and translate it into timing reports that measure how well the FAITH holds up.
So, like I promised:
The validation flow that I outline here translates FAITH into MEASUREMENT.
(PAUSE)
Of course the real measurement is the silicon, but that's a very expensive and time-consuming measurement.
The measurement I am talking about is a simulated measurement.



I have now told you
• Why I built a timing validation flow for GasP,
• and which pieces I borrowed, and which pieces are new.

The rest of my talk is about the new pieces of the flow.
• I will not be able to cover every detail
• but I will cover the most important ones.
• You can read the rest in my thesis.

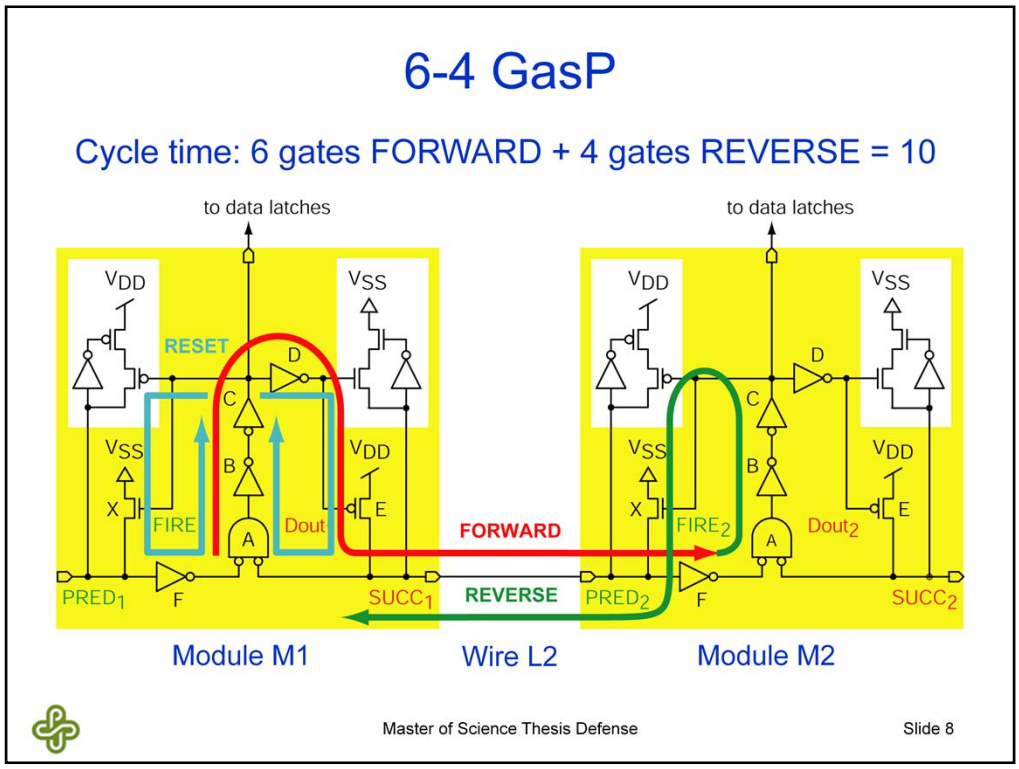
I will start with
• design details on GasP and the timing constraints in GasP

The biggest part of my talk is about library characterization.
I will show how timing constraints are translated into look up tables.
This requires three steps:
• In the first step, I partition the timing constraints.
• In the second step, I build a simulation environment for each partitioned timing constraint,
• using Electric and SPICE.
• In the third step, I run the SPICE simulator to generate the Look Up Tables.

The timing information in the Look Up Tables is used by Static Timing Analysis
• to validate the timing constraints.

Because this piece of the flow is not mine
• I will skip the flow details
• and instead focus on the resulting timing reports.

I will end with a summary and suggestions for future work.



In my thesis, I use a GasP family called 6-4 GasP.

• This slide shows two 6-4 GasP modules, M1 and M2, forming a 2-stage FIFO.
• The bundled datapath is omitted.
• The modules are connected by a single-track handshake wire in the middle. (POINT)
• Module M1 can drive the wire high via the P-transistor labeled E. (POINT TO E)
• Module M2 can drive the wire low via the N-transistor labeled X. (POINT TO X)

In GasP, we call the single-track wire a state wire, because it holds state.
• It is high when there is valid data - we call this full.
• And it is low when there is a bubble – we call this empty.

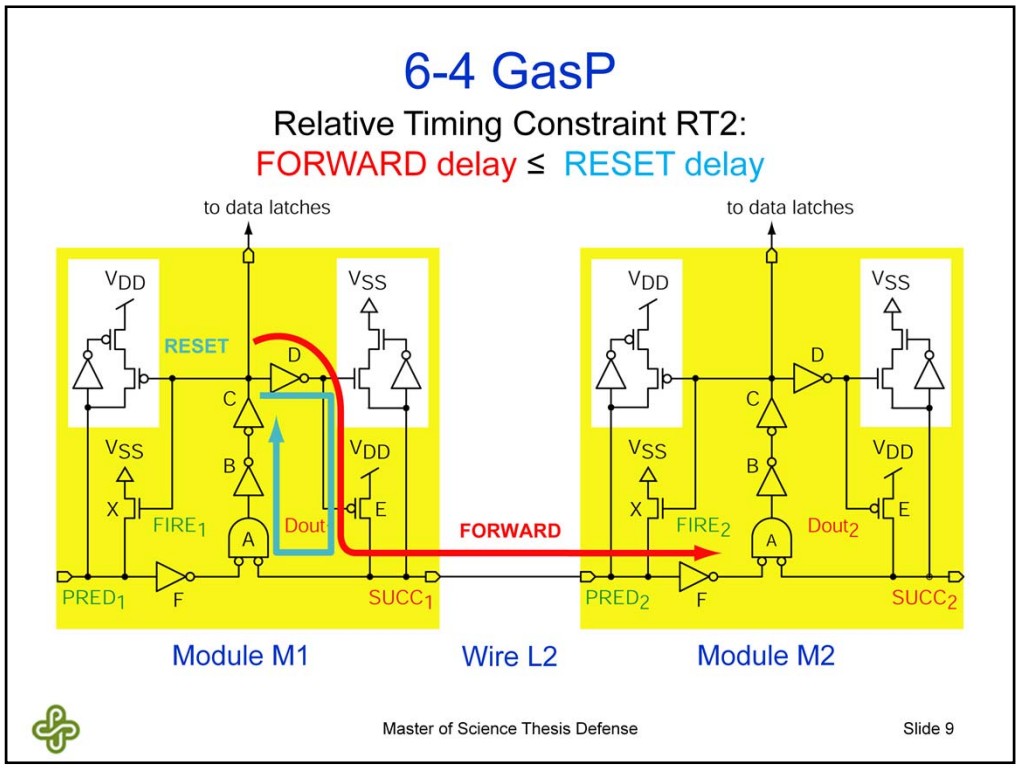
GasP module M1 acts when its predecessor is high, meaning full,
and its successor is low, meaning empty.
• When M1 acts, it forwards the full state to M2.
(CLICK)
• using the 6 gates on the red arrow: namely ABCDEF (POINT while naming)
• As soon as M2 can act, it drains the state wire.
(CLICK)
• using the 4 gates on the green arrow: namely ABCX. (POINT while naming)

The name 6-4 GasP comes from the 6 gates in the forward direction and the 4 gates in the reverse direction.
• The total cycle-time is 10 gate delays.
(CLICK)

In addition, each 6-4 GasP module has two self-resetting loops of 5 gates.
(CLICK)
• These two loops control the drive transistors E and X (POINT to left E and X)
• making sure that they drive only briefly.

When the single-track wire is not driven, its state is held by the half-keepers at the two ends of the wire,
• shown in these white areas (POINT TO the two MIDDLE KEEPERS)

(PAUSE)
The relation between the red, green and blue arrows can be specified in four relative timing constraints.
I will show you one.



Relative Timing constraint RT2
- gives the relation between the red arrow and the forward self-resetting blue arrow.

These two arrows operate concurrently.
- When the red arrow starts, the blue arrow also starts. and it will eventually kill the red arrow.

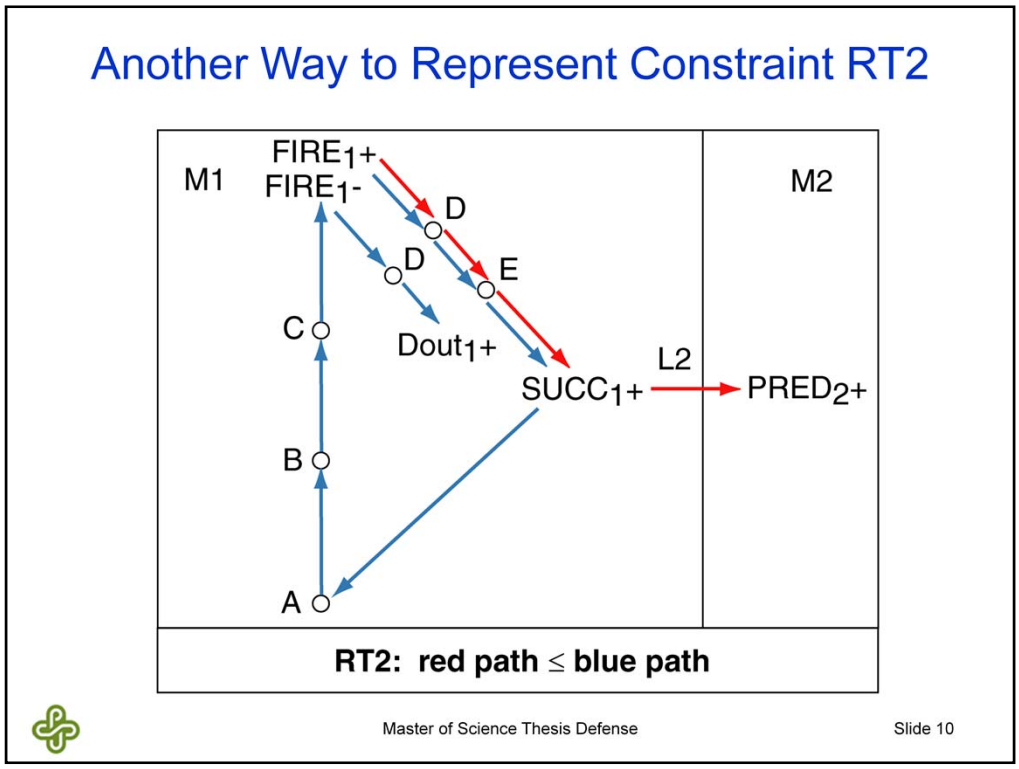
So, to operate correctly:
- the delay over the red arrow
- must be less than the delay over the blue arrow.

We start counting the delay of each arrow from the common point of divergence FIRE1.
Like this…
(CLICK)

This is what RT2 is about.
More precisely, RT2 expresses that:
(FOLLOW WITH POINTER OR FINGER)

- The forward delay over the red arrow
- from FIRE1 up
- through gates D and E
- until PRED2 in module M2 goes up

is less than
- The delay over the blue arrow
- from FIRE1 up
- through gates D, E, A, B, and C
- until Dout1 in Module M1 goes up
- and kills the red arrow
- by stopping pull-up transistor E.



Here is another way to represent RT2.

Let me repeat the idea:

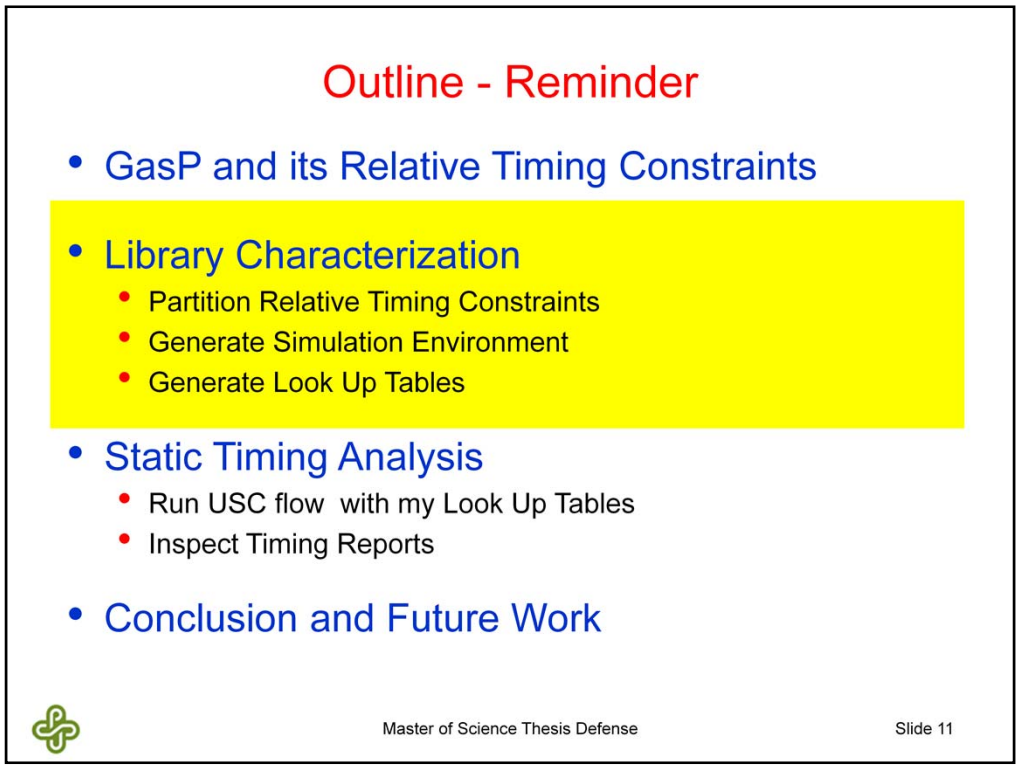
- The forward delay over the red arrow from the time that FIRE1 goes up through gates D and E
- until PRED2 in module M2 goes up

is less than

- The delay over the blue arrow from the time that FIRE1 goes up through gates D, E, A, B, and C
- until Dout1 in Module M1 goes up.

We are now ready for library characterization.

(NEXT SLIDE)

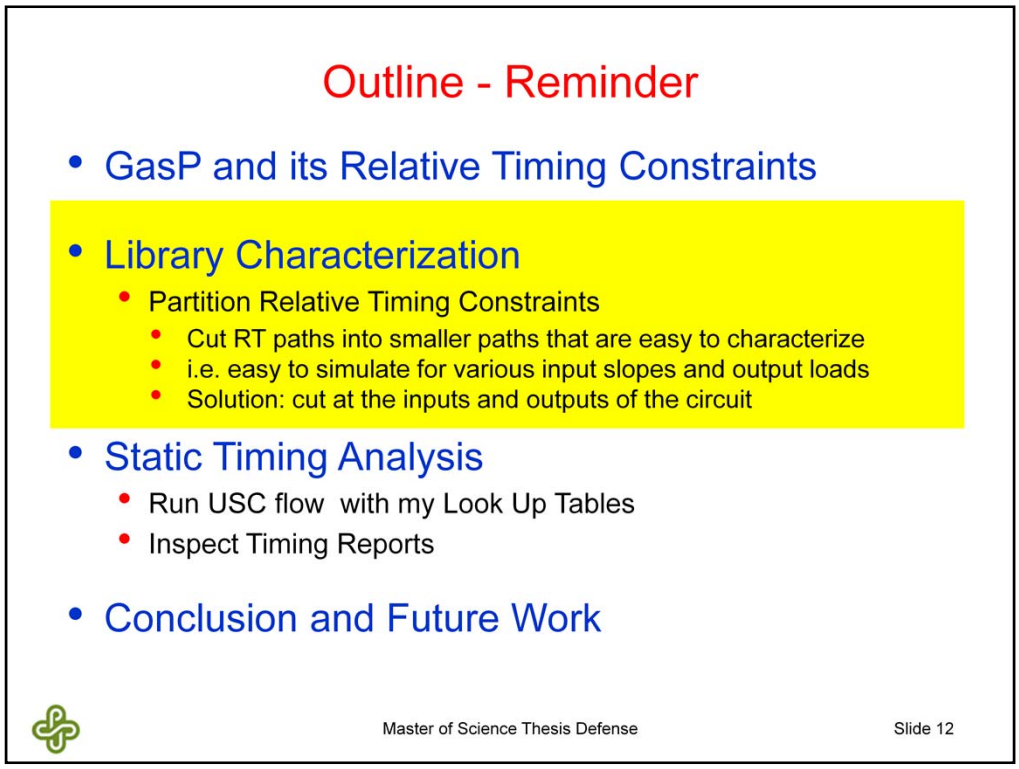


As a reminder of where we are, here's the outline again.

As you can see, the first step in library characterization is:

- to partition the timing constraints .

(NEXT SLIDE)



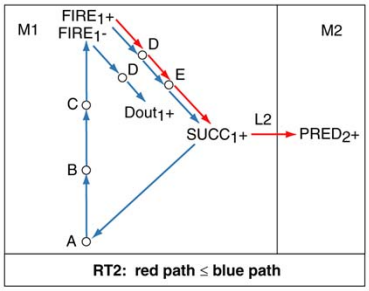
I will partition them into smaller paths that are easy to characterize.
Easy to characterize means that it's easy to simulate the paths for various input slopes and output loads.

So if we cut the paths at the inputs and outputs of the circuit,

- we can immediately control the input slopes and output loads.

(NEXT SLIDE)

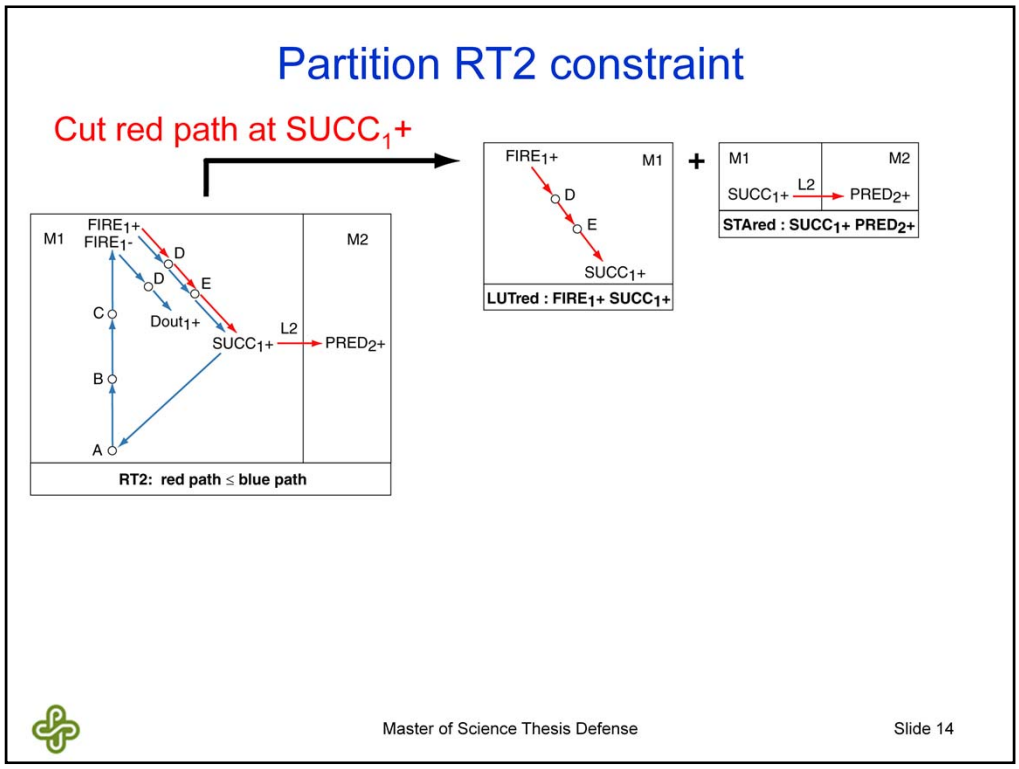
Partition RT2 constraint



For RT2, this means that

- I cut the red path at SUCC1-going up.

(RESULT shown in NEXT SLIDE)

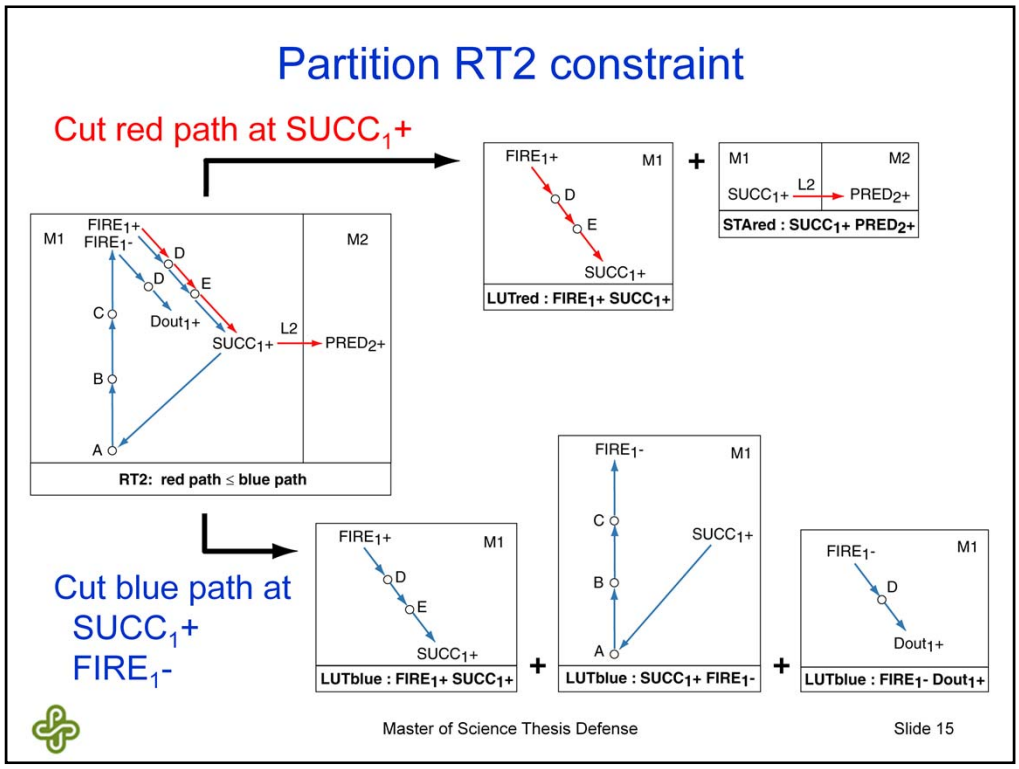


This gives two red sub-paths:

- One from FIRE1-up to SUCC1-up in module M1,
- the other from SUCC1-up in module M1 to SUCC2-up in module M2 over single-track wire L2.

And likewise, I cut the blue path.

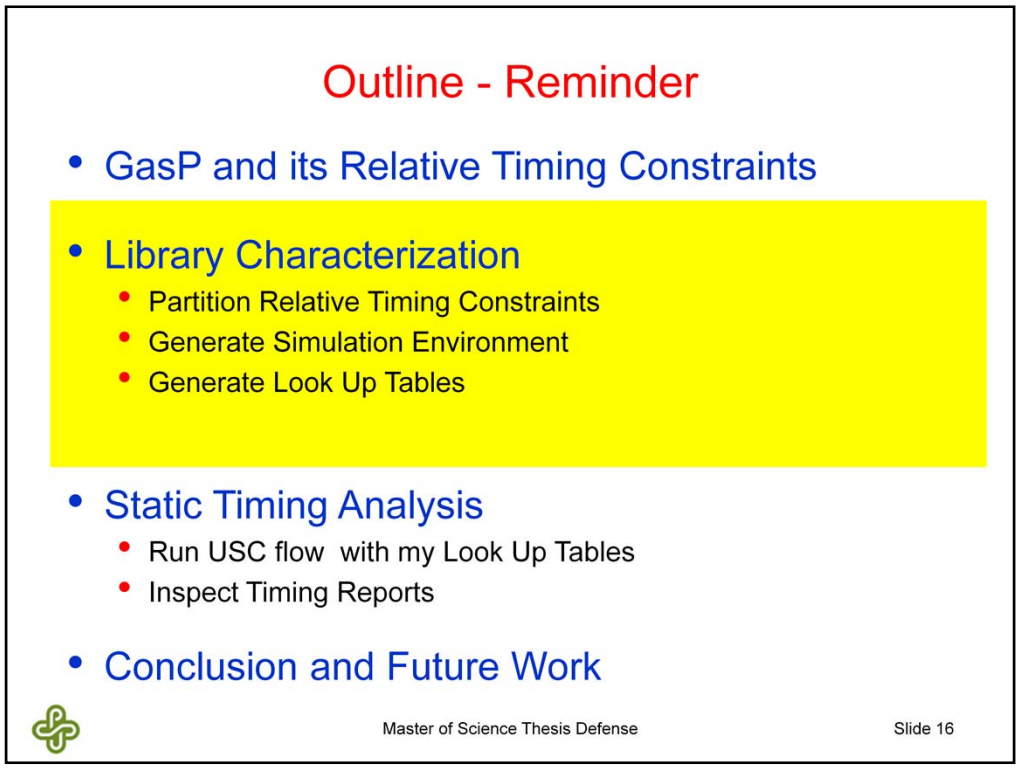
(Results shown in NEXT SLIDE)



I cut the blue path at SUCC1-going up and FIRE1-going down.

This gives three blue sub-paths:

- From FIRE1-up to SUCC1-up.
- From SUCC1-up to FIRE1-down.
- And from FIRE1-down to Dout1-up.

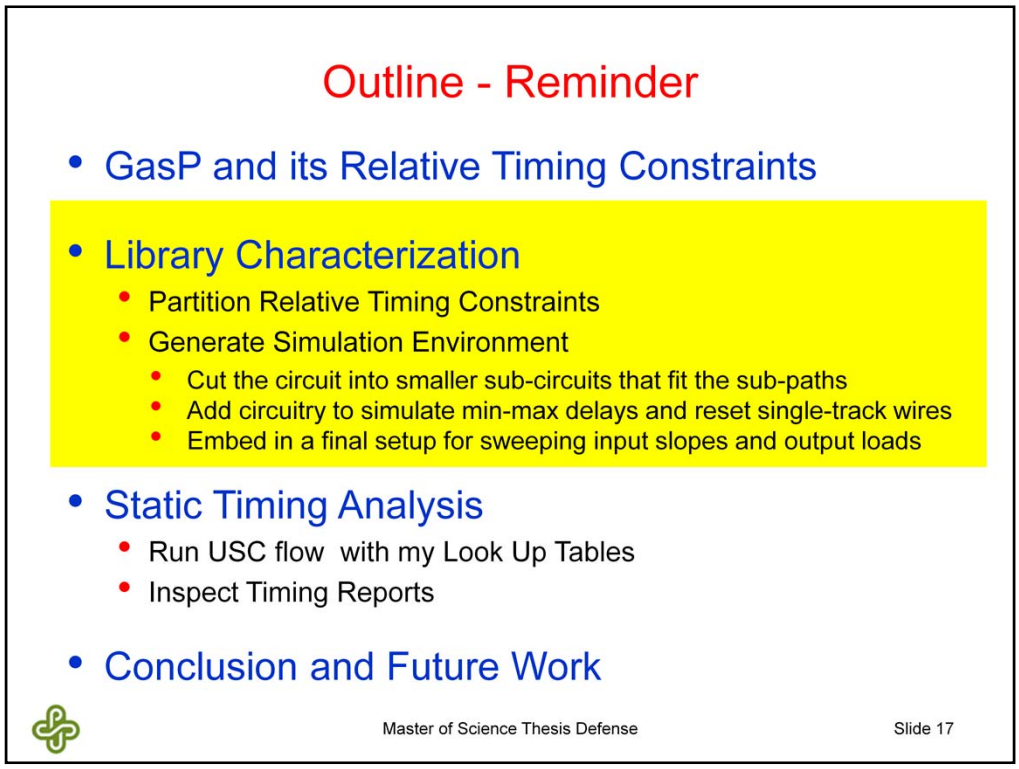


We are now ready for the next step in library characterization.
Here is the outline again.

As you can see, the second step is:

- to generate the simulation environment.

(NEXT SLIDE)

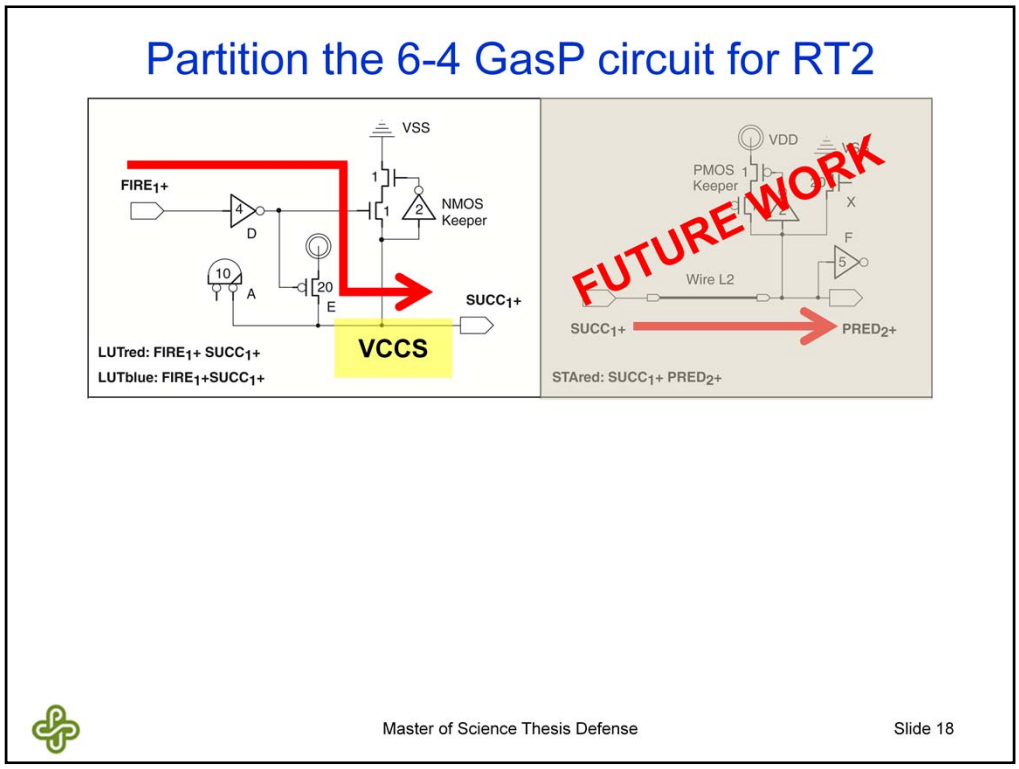


To get there, I first cut the circuit into sub-circuits that fit the partitioned sub-paths.
I include all the gates on the path and their output loads.

I then add circuitry to simulate minimum and maximum path delays as needed,
and to solve some issues in simulating paths that end in a single-track wire.

I embed the resulting circuit in a setup that allows me to sweep the input slopes and output loads.

(NEXT SLIDE)

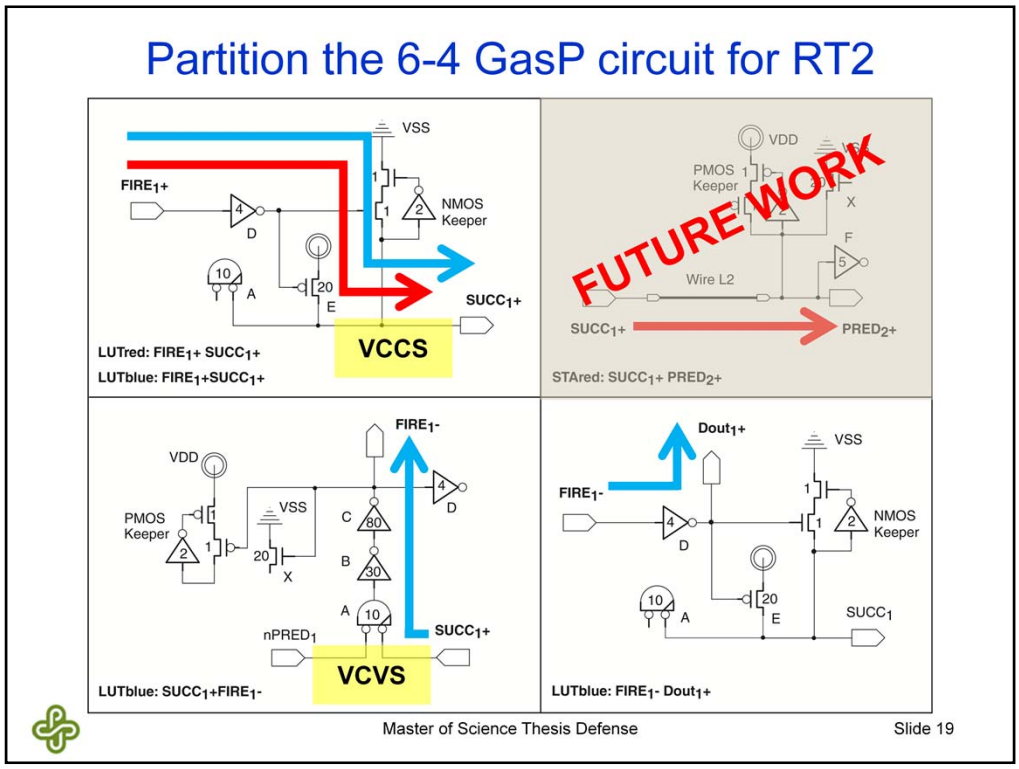


Here are the two sub-circuits for the two RED sub-paths in RT2.

In my thesis I simulate the first red path but not the second one.

(CLICK)
The second path requires a wire delay model for short- to-medium-to-long single-track wires.
I leave that for future work.

(NEXT SLIDE)



And here are the three sub-circuits for the three BLUE sub-paths.

One of them is shared with the first red path.

(PAUSE)
The yellow boxes are the extra circuit features that I mentioned in the outline.
I will discuss these in the next few slides.

What you need to remember is

- that RT2 requires that the total RED path delay is less than the total BLUE path delay.
- In particular, the total red path delay must be less than the total MINIMUM blue path delay.

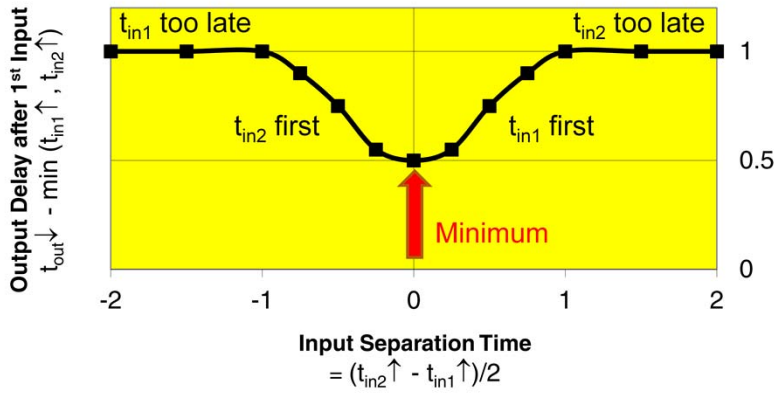
(POINT TO NOR-gate A in lower-left circuit and the VCVS device)
The yellow box with the text VCVS guarantees that this particular blue sub-path has minimal delay.

- This blue sub-path goes from SUCC1-up to FIRE1-down .
- It is the only path in RT2 with a multiple-input gate, namely NOR-gate A.
- The falling delay for gate A depends on when the inputs arrive relative to each other.

First Circuit Enhancement for RT2

What:

- Minimize the (BLUE) Reset Loop Delay
- by minimizing the delay of NOR-function A going down



The graph on this slide shows the dependency

- between the falling delay on NOR-gate A and the two inputs to A.

This graph is a variant of the “Charlie Diagram”, named after the late Charles Molnar.

- It maps the separation time between the two rising inputs
- to the delay of the falling output over the first input.

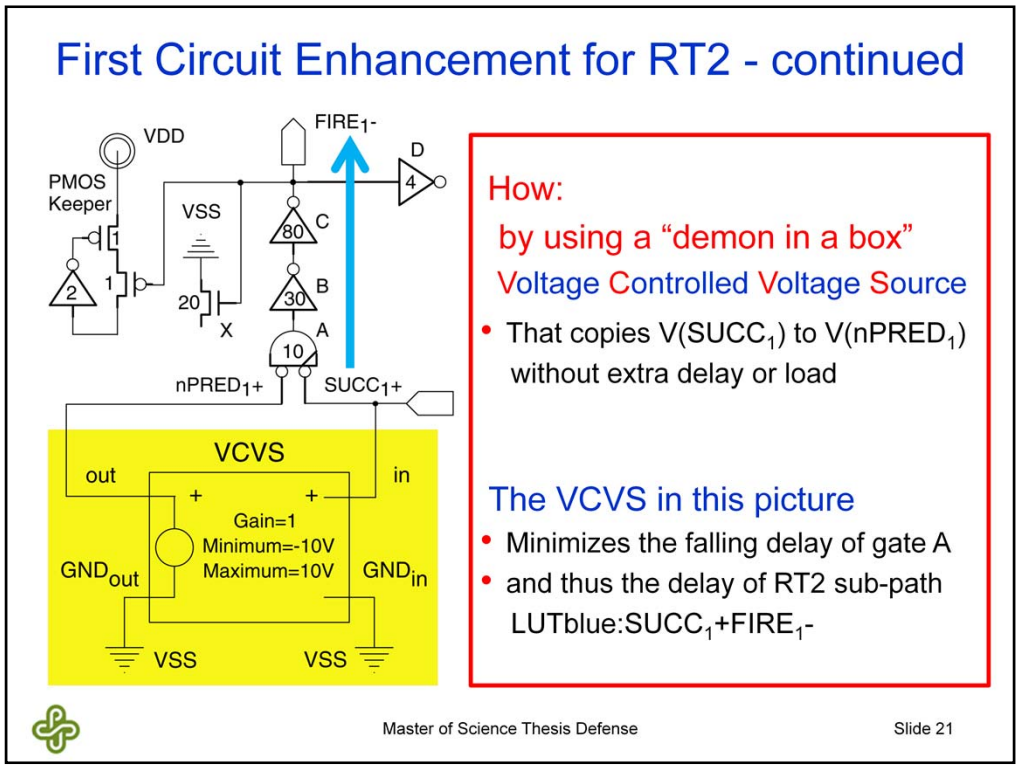
As you can see:

- The output delay is minimal when both inputs arrive at the same time.
- This is because the two N-transistors in the NOR-gate work in parallel.

When they work in parallel the gate goes TWO TIMES faster

- than when the gate operates on only one N-transistor.

So, for a minimal falling delay on A we need to synchronize the two rising inputs.



I do this with a Voltage Controlled Voltage Source.

A VCVS is a fictional device.

At the Asynchronous Research Center we call it a "Demon in a box".

- It copies the voltage difference between its input pin and input Ground
- to its output pin and output ground.
- It does this without adding delay or load – that's the fictional part.

This device exists in the library models of Electric and SPICE.

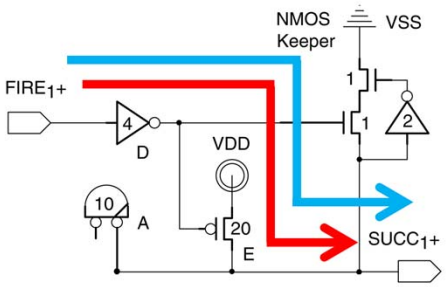
The VCVS in this picture

- copies the rising voltage level on A input SUCC1
- to the other A input nPRED1.

From the previous slide we know that this minimizes the falling delay for gate A

- and thus the delay of the blue RT2 sub-path LUTblue:$SUCC_1$+$FIRE_1$-.

Second Circuit Enhancement for RT2



What:

- Module M1 can only raise $SUCC_1$ because it's a single-track signal
- So: $SUCC_1$ must be re-initialized before the next simulation cycle

The second circuitry enhancement is needed for RT2 sub-paths that go from FIRE1-up to SUCC1-up.

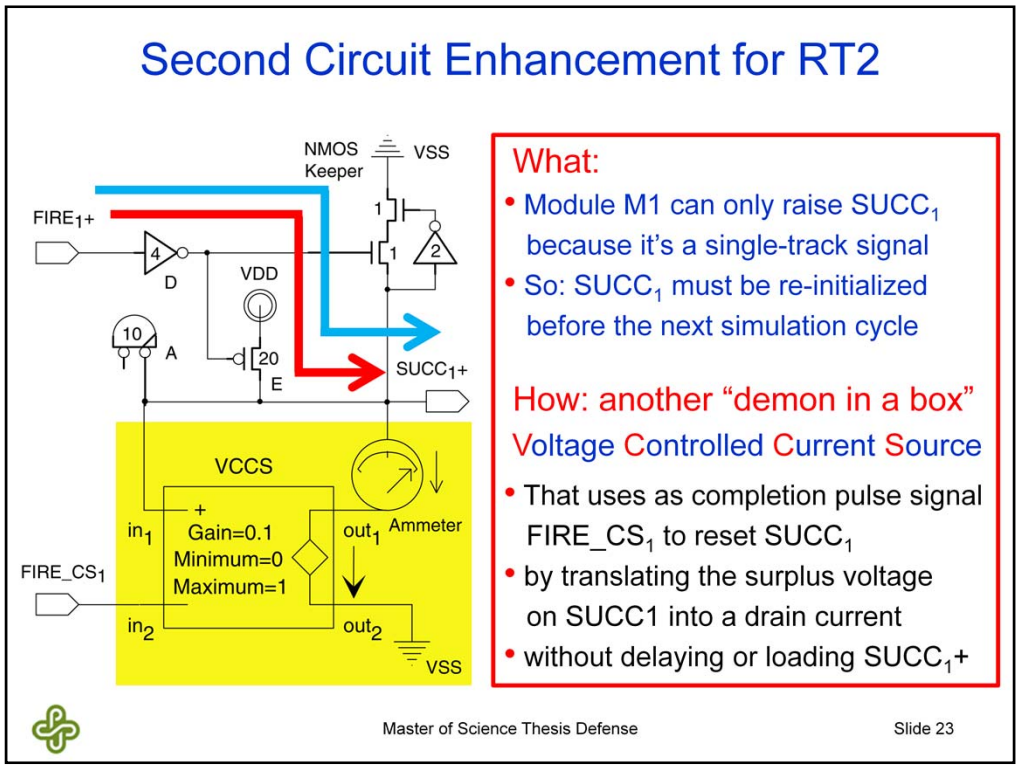
(NOTE: No need to say: LUTred:$FIRE_1$+$SUCC_1$+ and LUTblue:$FIRE_1$+$SUCC_1$+)

It is needed to solve a problem with the single-track output SUCC1.

Because this is a single-track signal, module M1 can only raise SUCC1.

- So, if we I want to simulate more than one path transition,
- I must reset SUCC1 to its initial low state
- after completion of the current simulation cycle
- and before the start of the next one.

(NEXT SLIDE)



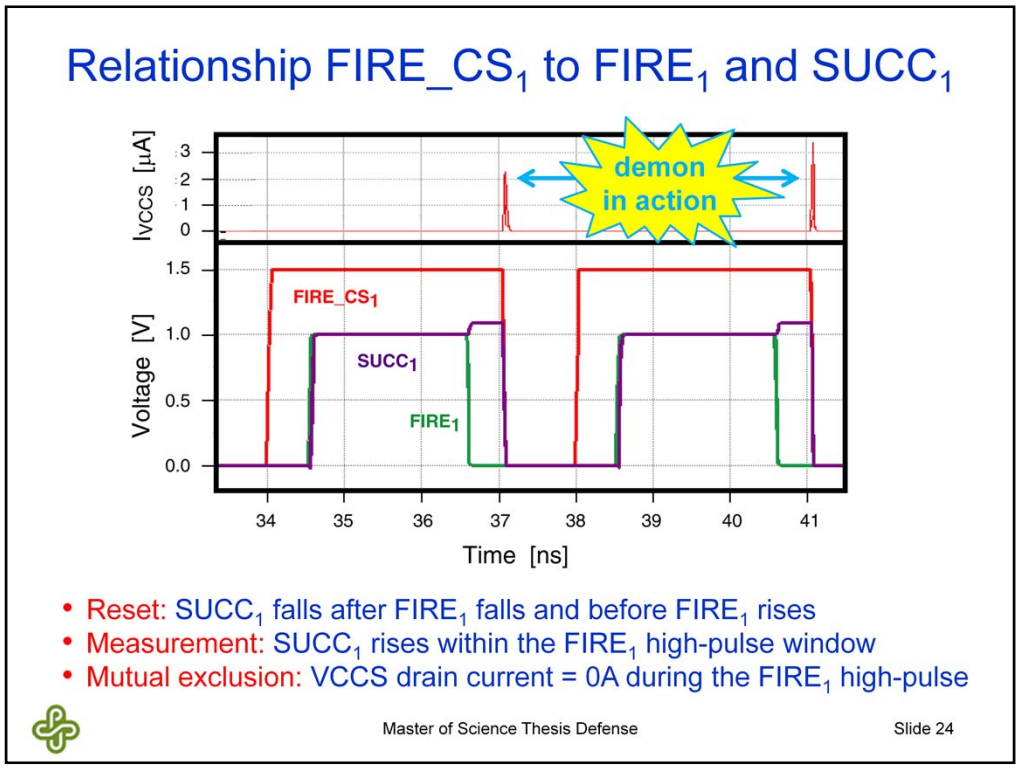
I do this with another "Demon in a box".

This one is called a Voltage Controlled Current Source.

It uses a completion pulse signal, called FIRE_CS1

- to drain SUCC1 at the right time
- by translating the surplus voltage on SUCC1 into a drain current.

And this is done without extra delay or load
on the rising transition for SUCC1.



This slide shows that the measurement and the VCCS work correctly.

The bottom window shows
- input signal FIRE1 in green,
- output signal SUCC1 in purple,
- and completion signal FIRE_CS1 in red.

The top window shows the drain current drawn by the VCCS.

You can see that the measurement and the VCCS work correctly by observing three things.

ONE:
- The purple output transition on SUCC1 high
- falls completely within the FIRE1-high measurement pulse.

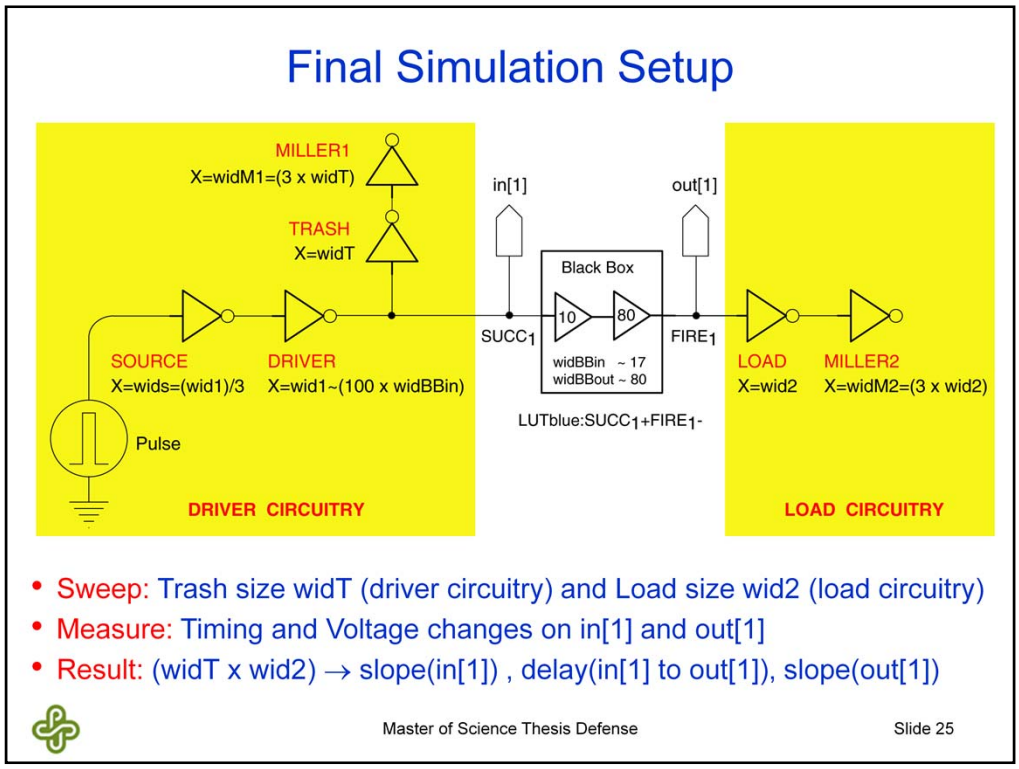
(POINT OUT)

TWO
- The VCCS device correctly resets SUCC1 back to zero volts
- after a green FIRE1 high pulse and before the next green FIRE1 high pulse.

(POINT OUT)

And THREE:
- The VCCS AND measurement operations are mutually exclusive
- as you can see by the lack of drain current during the green FIRE1-high pulse.

In other words, the demon acts ONLY after the measurement is over.
(CLICK)
(PAUSE)



The circuits are now ready to be embedded in a final simulation setup
- where we can sweep the input slopes and output loads for each critical timing path.

The picture shows the final setup for the BLUE sub-path LUTblue:SUCC1+FIRE1- of RT2.
The other sub-paths use a similar setup.

I have embedded the sub-circuit with the VCVS in the Black Box, and added:
- Driver circuitry at Black Box input SUCC1 (POINT to left YELLOW area)
- and Load circuitry at Black Box output FIRE1. (POINT to right YELLOW area)

The Driver circuitry starts with a pulse generator (POINT)
- which sets the pace for the simulation cycles.

The pulse generator is followed by a Source inverter and a Driver inverter.
- The source inverter is there to ensure that the Driver receives realistic input slopes,
- independent of what the pulse generator produces.

The Driver drives a Trash inverter as well as the Black Box input.
- This combination makes it possible to generate a wide range of input slopes for the Black Box.

The Load circuitry consists of a Load inverter followed by an extra inverter, called Miiller2.
The Miller2 inverter prevents that the output of the Load inverter floats.
(PAUSE)
If the OUTPUT of the Load inverter would float, it would change too fast,
and this would delay the INPUT of the Load inverter.
(PAUSE)
This is called the Miller effect.
The extra Miller2 inverter ensures that the Load inverter has a realistic Miller effect.

The same is true for the Miller1 inverter at the output of the Trash inverter.

All inverter sizes use a step-up of 3 going downstream.
- The only exceptions are the Driver-and-Trash combination and the Load inverter,
- which we sweep.

I sweep the input slope by sweeping the Trash size.
I sweep the capacitive load by sweeping the Load size.
And I measure timing and voltage changes on the input and output of the Black Box.

… and the Look Up Table that it generates

LUTblue:SUCC1+FIRE1-

Output Load [fF] \ Input Slope [ps]	14.2		15.1		16.3		21.4	
	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]
0.0	35.8	8.8	36.3	8.6	37.4	8.5	39.2	8.3
18.2	37.7	10.0	38.0	10.1	39.3	9.6	40.8	10.8
59.3	41.6	13.8	41.9	14.2	43.0	13.9	44.7	13.9
101.4	45.0	18.0	45.3	18.0	46.3	18.3	48.1	18.2
142.8	48.0	22.8	48.4	22.5	49.3	22.8	51.2	22.7
185.1	50.9	26.9	51.2	27.0	52.1	27.1	54.0	27.0

- Stores: (Input Slope x Output Load) → delay(in[1] to out[1]), slope(out[1])

This is what the simulation environment generates:
- a Look Up Table with the timing information for the BLUE sub-path LUTblue:SUCC1+FIRE1- of RT2.

I have translated the Trash and Load sizes into input slopes and output loads
- because that is what Static Timing Analysis tools like PrimeTime of Synopsys work with.

You can read the translation details in my thesis.

In the next slide I will show how PrimeTime uses this table.

… is used as follows

LUTblue:SUCC1+FIRE1-

Output Load [fF] \ Input Slope [ps]	14.2		15.1		16.3		21.4	
	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]
0.0	35.8	8.8	36.3	8.6	37.4	8.5	39.2	8.3
18.2	37.7	10.0	38.0	10.1	39.3	9.6	40.8	10.8
59.3	41.6	13.8	41.9	14.2	43.0	13.9	44.7	13.9
101.4	45.0	18.0	45.3	18.0	46.3	18.3	48.1	18.2
142.8	48.0	22.8	48.4	22.5	49.3	22.8	51.2	22.7
185.1	50.9	26.9	51.2	27.0	52.1	27.1	54.0	27.0

delay(12.24ps ,0fF)= **?**

slope(12.24ps, 0fF)= **?**

In my thesis, PrimeTime uses this table to find the delay and output slope for a table entry
- with an input slope of 12-point-24 picoseconds (12.24ps),
- and an output load of 0 femtofarads (0fF).

This table entry falls outside the Look Up Table range.
The nearest table entries in the Look Up Table are these high-lighted entries
(CLICK)
- with an input slope of 14.2 and 15.1 picoseconds,
- and an output load of zero femtofarad.

Primetime calculates the delay and output slope of the non-existing table entry
But using linear extrapolation from these two points.

Here are the results.
(Results follow in NEXT SLIDE)

… is used as follows

LUTblue:SUCC1+FIRE1-

Output Load [fF] \ Input Slope [ps]	14.2		15.1		16.3		21.4	
	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]	delay [ps]	output slope [ps]
0.0	35.8	8.8	36.3	8.6	37.4	8.5	39.2	8.3
18.2	37.7	10.0	38.0	10.1	39.3	9.6	40.8	10.8
59.3	41.6	13.8	41.9	14.2	43.0	13.9	44.7	13.9
101.4	45.0	18.0	45.3	18.0	46.3	18.3	48.1	18.2
142.8	48.0	22.8	48.4	22.5	49.3	22.8	51.2	22.7
185.1	50.9	26.9	51.2	27.0	52.1	27.1	54.0	27.0

delay(12.24ps ,0fF)= 35.8 –(((14.2 – 12.24) / (15.1 – 14.2)) x (36.3 – 35.8)) = 34.71 ps

slope(12.24ps, 0fF)= 8.8 –(((14.2 – 12.24) / (15.1 – 14.2)) x (8.6 – 8.8)) = 9.24 ps

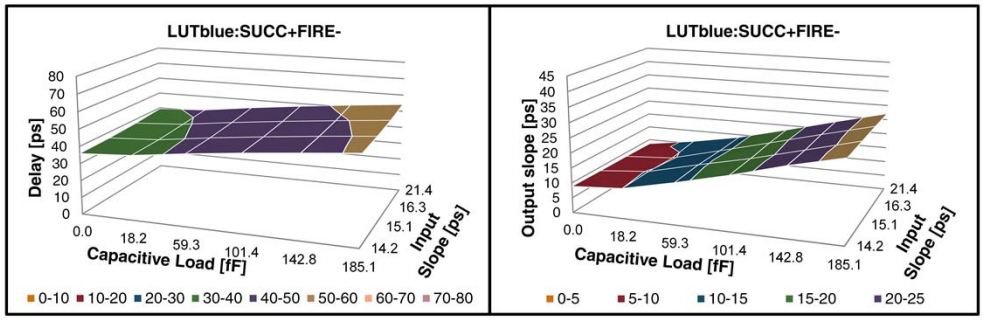
(results – no explanation needed)

(PAUSE – before going on to next slide)

Landscape Graphs for the Look Up Table

DELAY landscape
(color distinction every 10 ps)

OUTPUT SLOPE landscape
(color distinction every 5 ps)



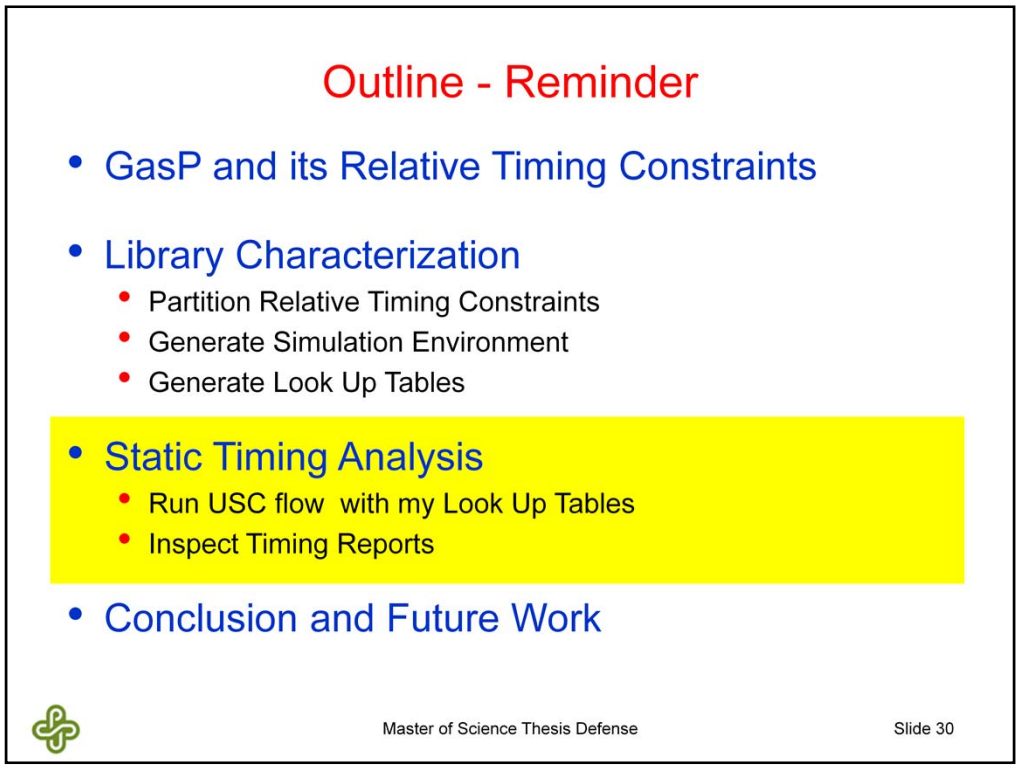
BOTH are very amenable to linear approximation techniques

The reason why interpolation and extrapolation works is because
- the landscapes for the slope and delays are very smooth and linear in this range.

This slide shows the two graphs for the BLUE sub-path LUTblue:SUCC1+FIRE1- in RT2.
As you can see, BOTH graphs are very amenable to linear approximation techniques.

Nevertheless:
- it is always wise to make sure that the Look Up Table covers the trends in changes.

You can do this by simulating enough points
- in the typical region
- as well as in the outlier regions.



This is the point where Library Characterization ends and Static Timing Analysis takes over. Here is the outline again, as a reminder.

I will skip the details of the timing analysis and focus on the timing reports.

(NEXT SLIDE)

Timing Report for RT2

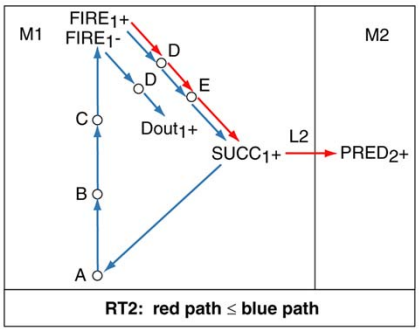
```
****************************************
Report : timing
       -path_type full
       -delay_type max
       -max_paths 1
Design : GASP_FIFO2
Version: C-2009.06-SP3
Date   : Mon Oct 11 10:28:45 2010
****************************************

  Startpoint: M1/FIRE (clock source 'M1/FIRE')
  Endpoint: M2 (rising edge-triggered data to data check
                                        clocked by M1/FIRE)
  Path Group: M1/FIRE
  Path Type: max

  Point                                   Incr       Path
  ------------------------------------------------------------
  clock M1/FIRE (rise edge)               0.00       0.00
  clock source latency                    0.00       0.00
  M1/FIRE (GASP_Module)                   0.00       0.00 r
  M1/SUCC_OUT (GASP_Module)              26.36      26.36 r
  M2/PRED_IN (GASP_Module)                0.00      26.36 r
  data arrival time                                 26.36

  clock M1/FIRE (rise edge)               0.00       0.00
  clock source latency                    0.00       0.00
  M1/FIRE (GASP_Module)                   0.00       0.00 r
  M1/SUCC_OUT (GASP_Module)              26.36      26.36 r
  M1/FIRE_PS (GASP_Module)               34.71      61.07 f
  M1/Dout (GASP_Module)                  19.32      80.39 r
  data check setup time                   0.00      80.39
  data required time                                80.39

  data required time                                80.39
  data arrival time                                -26.36
  ------------------------------------------------------------
  slack (MET)                                       54.03
```



RED path delay = 26.36 ps

BLUE path delay = 80.39 ps

RT2 is satisfied
Slack = 54.03 ps

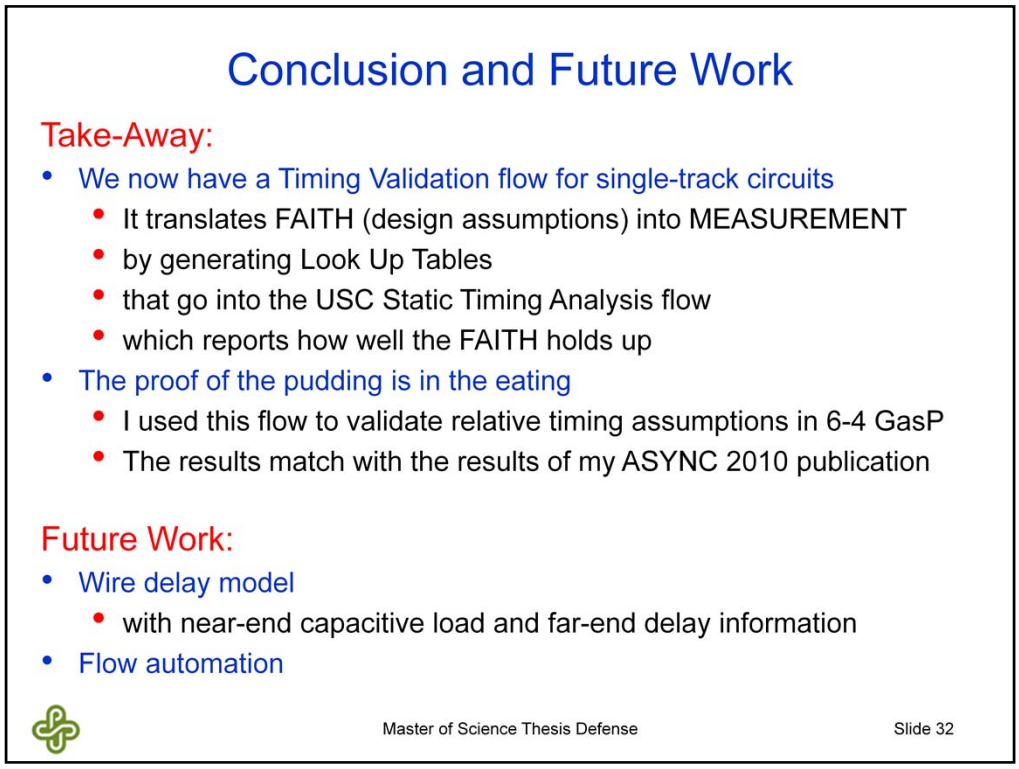
Here is the timing report for relative timing constraint RT2 - it was generated by Primetime.
As a reminder, I have added the graphical representation for RT2 on the side.
(POINT OUT)
• Remember that the delay of the red path from FIRE1-up to PRED2-up
• must be less than de delay of the blue path from FIRE1-up to Dout1-up.

Let's look at the timing report for the red path first.
• Primetime finds the first red sub-path from FIRE1-up to SUCC1-up and reports a delay of 26.36 picoseconds
(CLICK)
It finds zero delay for the sub-path from SUCC1-up to PRED2-up, because I did not provide a Look Up Table for L2.
(CLICK)
So, the total delay reported for the red path is 26.36 ps.
(CLICK)

Now, let's look at the blue path.
• Primetime finds the first blue sub-path from FIRE1-up to SUCC1-up.
• This sub-path is shared with the red path, so not surprisingly PrimeTime reports the same delay as before: 26.36 ps.
(CLICK)
It finds the second path from SUCC1-up to FIRE1-down, and reports a delay of 34.71 ps.
(CLICK)
• This is the same amount of delay that we calculated a few slides ago
• using the Look Up Table for "LUTblueSUCC1plusFIRE1minus" (LUTblueSUCC1+FIRE1-).
It finds the third and final path of RT2 from FIRE1-down to Dout-up, and reports a delay of 19.32 ps.
(CLICK)
Adding these three delay numbers gives a total delay for the blue path of 80.39 ps.
(CLICK)

If we subtract the delay of the red path from the delay of the blue path we get a slack of 54.03 ps.
(CLICK)
That's a large slack !!!
• The reason why this slack is so large is because we ignored the forward transfer delay for single-track wire L2
This implies that the wire delay over L2 can take up to 54.03 ps.

If the delay of L2 EXCEEDS 54.03 ps then RT2 no longer holds.
Invalidating RT2 amounts to module M1 stopping its drive before M2 sees the transition.
• This does not immediately invalidate the circuit operation,
• but it makes the circuit less robust and more noise sensitive.
I showed this in the paper that I published and presented at the ASYNC 2010 conference in France.



I developed a Timing Validation flow for single-track circuits, using GasP.

- The flow translates FAITH into MEASUREMENT.
- FAITH here means: design assumptions.

I did this

- by generating Look Up Tables
- that standard Static Timing Analysis tools can use.

I feed these tables into the USC Static Timing Analysis flow

- which produces a health report that tells you how well the FAITH holds up.

Because the proof of the pudding is in the eating:
I did not just develop this flow but I also used it.

I used it to validate relative timing assumptions in 6-4 GasP.

- The results match with the results of my ASYNC 2010 publication.
- You can find the details in my thesis.

As far as future work goes, what this flow needs most is a wire delay model .
The wire delay model should

- distinguish the effective capacitance seen by the near-end of the wire
- from the resistance and delay seen by the far end.

It may be that an Elmore delay model suffices.

Last but not least: this flow needs to be automated.


THANK YOU!
Title: TECOTOSH
TEnsion + COmpression + TOrsion + SHear
Location: Maseeh College
Installed: March 2006.
Dimensions: 130' x 40' x 40'.
Materials: Stainless steel truss, laminated dichroic glass, stainless steel cables and hardware. Aluminum light housings.
Engineers:
Bob Grummel and Grant Davis.
Project Manager: Oanh Tran.
Slide 33
Master of Science Thesis Defense

THE END