ASYNCHRONOUS RESEARCH CENTER

Portland State University

Subject: **Third Class Handout – Linear GasP**
Date: **October 8, 2010**
From: **Ivan Sutherland**
ARC#: **2010-is45**

References:

ARC# 2010-is43: Class 1 – Ring Oscillators, Ivan Sutherland, 26 September 2010
ARC# 2010-is44: Class 2 – GasP Rings, Ivan Sutherland, 1 October 2010

PURPOSE

This memo introduces a linear pipeline of GasP circuits with latches that can carry data. We will consider "source limited" and "sink limited" cases. We will also learn about "keepers." To do all this we'll learn some more things about Electric.

HIERARCHY – UP AND DOWN IN ELECTRIC

I like to assemble complicated circuits from simpler parts. For example, a sorting circuit may have several registers, a register may consist of many latches and each latch may consist of entrance gates, a state holding element, and some amplifiers. Thus the parts of the circuit form a "hierarchy."

I think of a hierarchy as a tree structure with a root, branches, more branches and so on out to the leaves. The leaves of the tree are the smallest of its parts: they have no smaller sub-parts. Because they are indivisible one might call them "atomic."

Unfortunately comparing a hierarchy to a tree leads to confusion about what "up" and "down" mean in the hierarchy. In a hierarchy of people, "up" generally means towards people with more authority and "down" means towards the common folk. That is the sense in which I shall use "up" and "down." Down is towards the leaves; up is towards the root. It always seems a little strange to me to go "up" towards the root of a tree and "down" towards its leaves. But that is the sense of "up" and "down" used by Electric.

It helps me to think of going "down" into a sub-circuit. In Electric if I select an icon, I can go down into it. When I do that I get to see the icon's "underlying" circuit. I

go down to the circuit definition. Electric will remember where I came from, so if I go back up I'll get back to where I was before.

A sub-circuit can appear in several places; one can use many "instances" of an icon. Thus to make a register I include many instances of the latch icon. And so it is true that although there's only one way to go "down" the hierarchy, there may be many ways to go up. Unlike a living tree, a hierarchy can re-use an identical leaf icon on many branches. Two separate hierarchies may use the same set of circuit ions and thereby share their definitions. This is what makes libraries of circuits useful.

LIBRARIES

You can download a collection of libraries called **a180Library-20sep10** from the ARC web site. In the collection is something called **anOpener** and inside it is a cell called **gallery{sch}**. The cell called **gallery{sch}** is a convenient way to focus Electric's attention on all the other parts of the 180Library. When you ask Electric to open **anOpener** Electric will load the entire collection of libraries, making them available for you to use.

For today's assignment you will need circuits from **a180Library-20sep10**. Without the library you can't open the circuits we want you to simulate. You may wish to explore the content of **a180Library-20sep10.** It contains a lot of circuits, and as you can see from their dates, they have accumulated over several years.

Of particular interest is a cell called **aGallery** in the library called **purpleGeneric180**. This cell has 7 sub-parts each of which contains a group of related icons. When you first open **aGallery** you may see only seven rectangular blocks, or you may see a whole bunch of colored icons. If you see only the rectangular blocks, you need to "expand" them. Select them by clicking on each with your mouse. Then the command "cell -> Expand Cell Instances -> All the Way" will exhibit their content for you to see. Each of the seven blocks holds a collection of related icons: transistors, simple one and two input gates, three input gates, and so forth. You have to go "down" into a block before you can copy its icons for use. I often copy icons from this library and paste them into my circuit drawings.

IMPORTANT NOTES: In the Cell menu are two similar commands: Expand and Extract. USE "Expand" to see what symbols are in each group. AVOID "Extract" because it changes the hierarchy. For now we'll ignore all of the purple gate symbols; you don't need them for this week's assignment.

You can go down into the symbols in **aGallery** to see how they're built. Many of them have two icons related by deMorgan's rule. For example there are two inverter icons, one with bubble on input and the other with bubble on output. Also for example, a HI-input, LO-output AND is the same as a LO-input, HI-output OR. Electric permits each circuit definition to have multiple icons; for example, the two different inverter icons both represent the very same inverter circuit.

Remember, you can always go "down" into an icon to see what's inside it. For example, if you go down into a red inverter icon, you'll find that it's made of two red transistor icons. If you go down into that red transistor icon you'll find a yellow transistor icon. If you go down into the yellow transistor icon, you'll find Electric's leaf node icon for a transistor. Electric computes strengths in X and widths in lambda along the way.

THREE LINEAR FIFOs

This assignment involves a row of GasP circuits, each of which drives a latch. This structure forms a First-In-First-Out buffer, also known as a FIFO. At the beginning of the FIFO is a "source" and at the end of the FIFO is a "sink." This source is an oscillator that feeds data to the FIFO using "fake" data values from a pulse generator. The sink is a "bit bucket" that discards the data values it receives. Source and sink together let us watch data flow through the FIFO.

There are three copies of the FIFO, each in its own row. The top FIFO has a free running source and sink. It should run as fast as its GasP circuits can go. Inside **timerA** there are two pulse generators. One pulse generator produces a "master clear" pulse, `mc`, to get the FIFO started. The other pulse generator produces fake data, `data`, so the latches change output values from time to time.

To the left of the middle FIFO, **timerB** has two more pulse generators that produce the signals called, `source` and `sink`. The `source` and `sink` signals start and stop the FIFO. You will be able to watch this FIFO fill and drain.

The bottom FIFO looks just like the top one except for the inverters connected to `fire[20:29]`. These inverters provide loads to vary the delay of the different GasP circuits. Notice that the biggest load inverter makes the `fire[24]` stage slowest.

THE REAL GasP CIRCUITs

Go down into one of the gaspPlain10 modules. You will see three black symbols, a red inverter and an orange wire model. The red inverter is of strength X=5 which means that it's 5 times as strong as the minimum size inverter in this technology. The wire model represents 100 lambda of metal2 wire 2.8 lambda wide.

Predecessor Driver. The black symbol on the left labeled "10" is a predecessor driver. It is of type **`predDri10wMC`**. If you look inside it you will find the strength X=10 N-type transistor that drives the predecessor state wire LO whenever the GasP module fires. You will also find a strength X=3.667 N-type transistor connected to master clear. During master clear this N-type transistor drives the state wires between modules LO, meaning empty, to initialize the FIFO.

The rest of the stuff in the predecessor driver is a "half keeper." Whenever the state wire called `pred` is HI, the stack of three P-transistors will keep the state wire

HI except during master clear or when the GasP module needs to drive the state wire LO. This amounts to half of a latch that gets out of the way when it's not needed.

Successor Driver. The black symbol on the right labeled "10" is a successor driver. It is of type **`sucDri10`**. It's main component is the strength X=10 P-type transistor at the top that will drive the successor state wire, `succ`, HI when the GasP module fires. The rest of the stuff at the bottom of the schematic is a "half keeper." Whenever the state wire called `succ` is LO, the stack of two series N-transistors will keep the state wire LO except when the GasP module drives the state wire HI.

Center AND. In the center of the GasP module is an AND function with amplification. The black icon suggests the circuit, even showing the sizes of the gates. Look inside and see that it also contains two wire models of length 308 and 400 lambda.

When there is a layout available, Electric can "back annotate" its wire models to show the length of the wires in the layout. I tend to include wire models in all my schematic diagrams, making a guess about their probable layout length. My guesses are always too short which is why back annotation is important.

HOW IT WORKS

Soon after the center AND condition is satisfied, the signal called `fire` goes HI; we say that the GasP module has "fired." For the GasP module to fire its predecessor state wire, `pred`, must be HI, meaning full, and its successor state wire, `succ`, must be LO meaning empty.

Notice the loops to the right and left of `fire`. When the GasP module fires the predecessor driver drives `pred` LO. The successor driver also drives `succ` HI. Both these actions turn off the AND gate, forcing the fire pulse to end. You can count the gates to see that the fire pulse will last about 5 gate delays.

Remember that each state wire has a predecessor driver on one end and a successor driver on the other end. Thus the state wire carries signals in both directions. A GasP module can tell when its predecessor GasP module drove its predecessor state wire HI to indicate that data are available. I like to say that a HI state wire indicates that the predecessor GasP module is "proffering" data. Look up "proffer" in the dictionary.

A GasP module can also tell when its successor GasP module drove its successor state wire LO. By driving the state wire LO, the successor GasP module indicates acceptance of the proffered data.

Notice that two successive GasP modules always fire alternately. This must be so because one drives the state wire between them HI and the other drives it LO. This alternating action of successive stages is characteristic of all latch-based FIFOs.

WHAT TO EXPLORE

The questions on the answer sheet will guide your exploration of these circuits. In the free-running FIFO you can observe that successive stages fire alternately. They work just like alternating clock signals, often called phi0 and phi1. However, because the forward and reverse latency differ, successive fire pulses overlap slightly. How much overlap is there?

You can also observe data values flowing through the latches. A pulse generator provides fake data, a string of ones and zeros, for the latches to capture. You can follow the progress of a particular data item through the latches by looking at the `data[0:29]` signals between the latches. Notice that the latch output changes only when its control input, `fire[0:29]`, is HI.

When `fire` is HI, the latch is transparent. When `fire` is LO the latch is opaque. Notice that I avoid the words “open” and “closed” because plumbers and electricians differ in what those words mean. An open valve allows water to flow but an open switch blocks electricity. I don’t know what open and closed mean for data; are data like water or like electrons?

Because successive fire pulses overlap, there’s a moment when successive latches are both transparent. What is the risk of data sneaking past two latches instead of only one? In order for the FIFO to work properly, its latches must fit within both upper and lower delay bounds. If the latches are too slow, the data values will be left behind. If the latches are too fast, data might get past two stages during the short interval when both latches are transparent.

The middle FIFO runs and stops alternately because its source and sink are connected to pulse generators in **timerB**. When the sink stops, you can watch the FIFO fill up with data. It takes a while for the congestion to reach the source. When the sink restarts you can watch the data start to flow again. It takes a while for the congestion to clear from the source. You are asked to observe how “elastic” this FIFO is. Such a FIFO can smooth the flow of data from a supplier to a user by storing excess input data until a user is ready to accept the data.

The bottom FIFO supports extra loads on its fire signals. This makes it run slower. Because there’s a big load on `fire[24]` that stage will be slowest of all. The slow stage in the middle lets you observe the intervals when the stages before and after `fire[24]` must wait for data or space. A stage can fire only when the LO-input AND function is satisfied. Which edges, rising or falling, at the bubble inputs of the AND will initiate a fire action? By looking at the inputs to the AND function inside a GasP modules you can tell both when it waits and whether it is waiting for its predecessor or for its successor.

ANSWER SHEET – due by beginning of class on 19 October 2010 **3**

Name _____

Turned in on Date _____

Answer from simulation:

My simulation uses _____ technology – e.g. 180 MOSIS

My free-running FIFO operates every _____ psec for a throughput of _____ GDI/sec.
(GDI/sec is GIGA data items per second)

I observe that the fire signals of successive stages are both non-zero for _____ psec.

The forward latency of my FIFO for the 9 stages, `fire[0]` to `fire[9]` is _____ nsec.
(hint: Watch the data values flow to know which fire pulse is which).
The data advance rate of my FIFO is _____ psec/stage.

My FIFO accepted ____ data items at `fire[10]` before its sink stopped.

My FIFO delivered ____ data elements at `fire[19]` before its sink stopped.

Where are the excess data elements after the sink stopped?

The reverse latency of my FIFO for the 8 stages `fire[19]` to `fire[11]` is ____ nsec.
(hint: Look at `fire[11:19]` as the FIFO restarts after being full.)
The bubble reverse rate of my FIFO is _____ psec/stage.

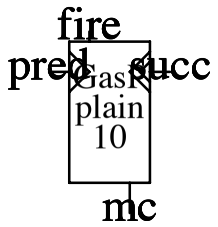
My loaded FIFO operates every _____ psec for a throughput of _____ GDI/sec.
Measure at `fire[22]`. (GDI/sec is Giga Data Items per second)

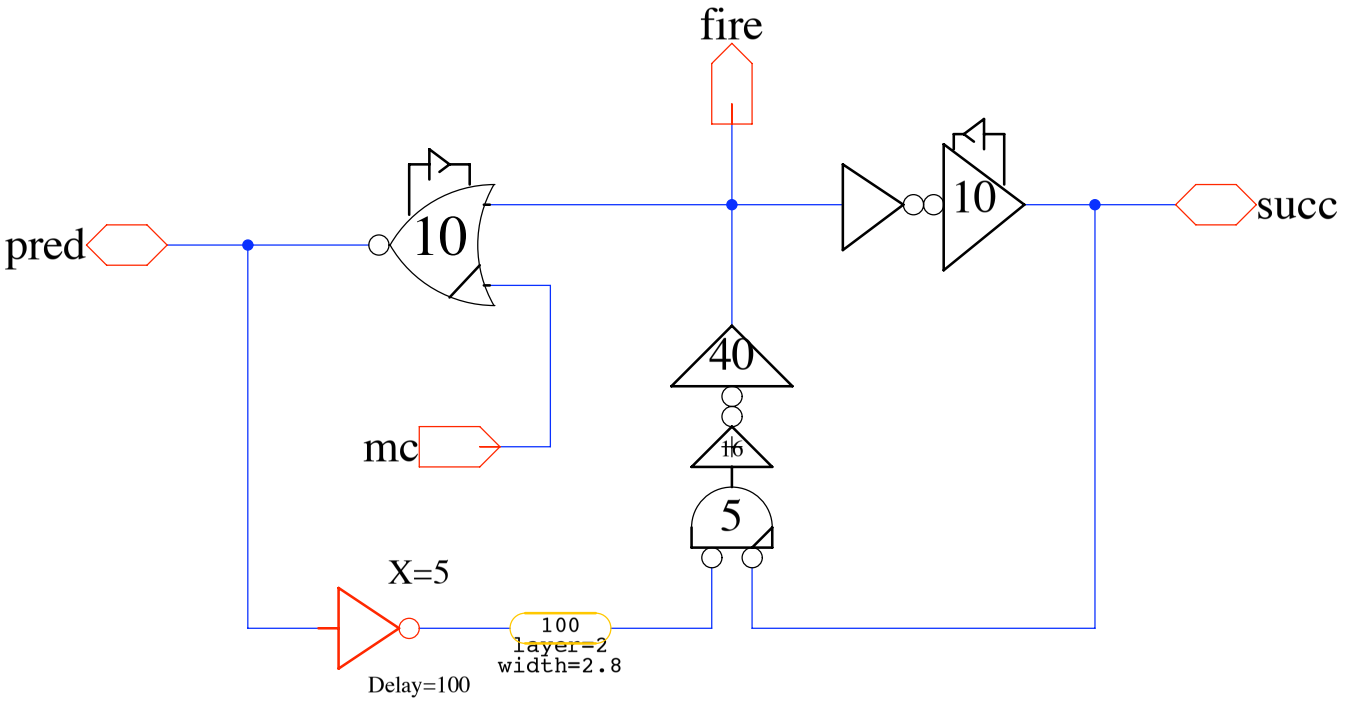
Examine the two inputs to the AND function in stage [23].
The (rising) (falling) edges control the onset of `fire`. (Hint: measure the proper edge.)
Stage [23] predecessor signal arrives _____ psec (before) (after) its successor.

Examine the two inputs to the AND function in stage [25].
The (rising) (falling) edges control the onset of `fire`. (Hint: measure the proper edge.)
Stage [25] predecessor signal arrives _____ psec (before) (after) its successor.

gaspPlain10

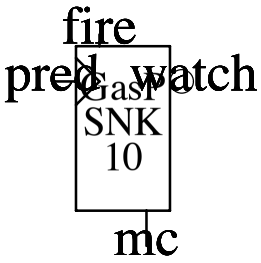
ies 17 June 2010

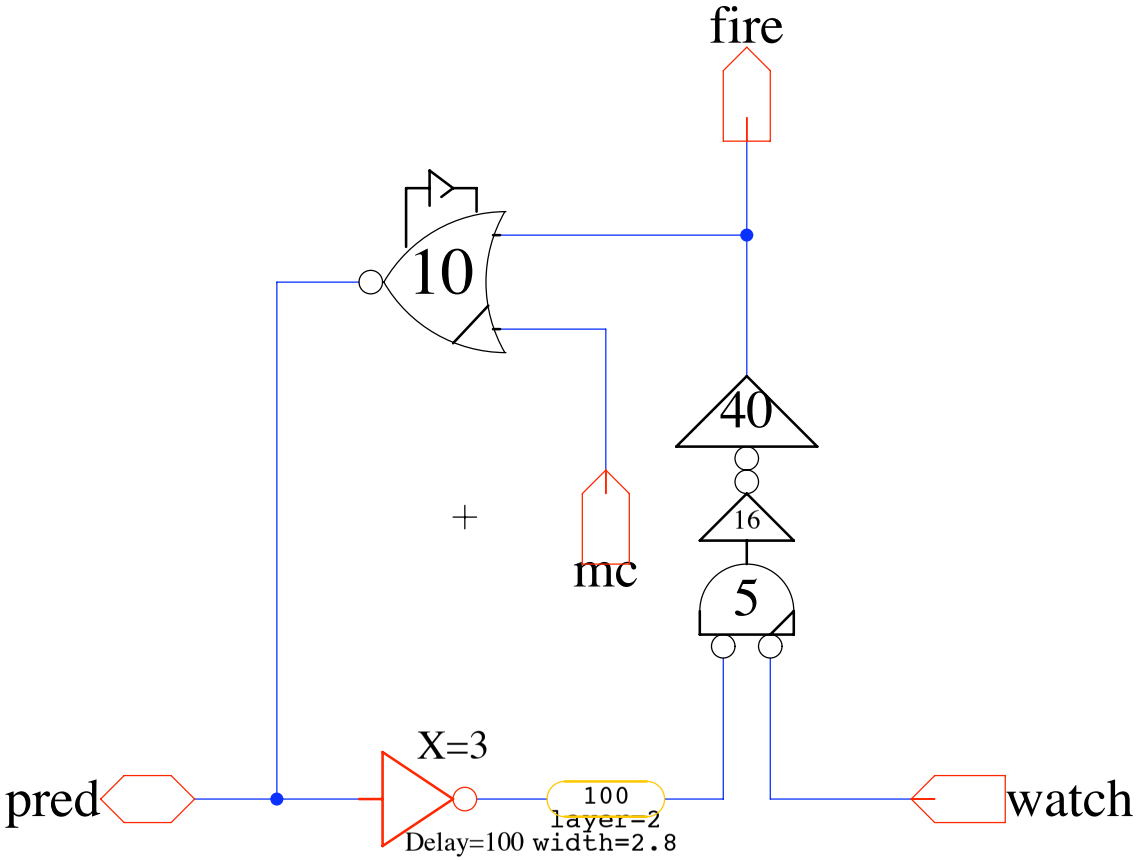




gaspSink10

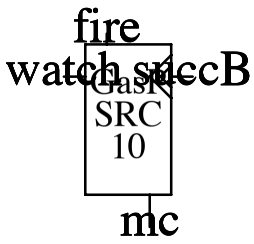
ies 17 June 2010

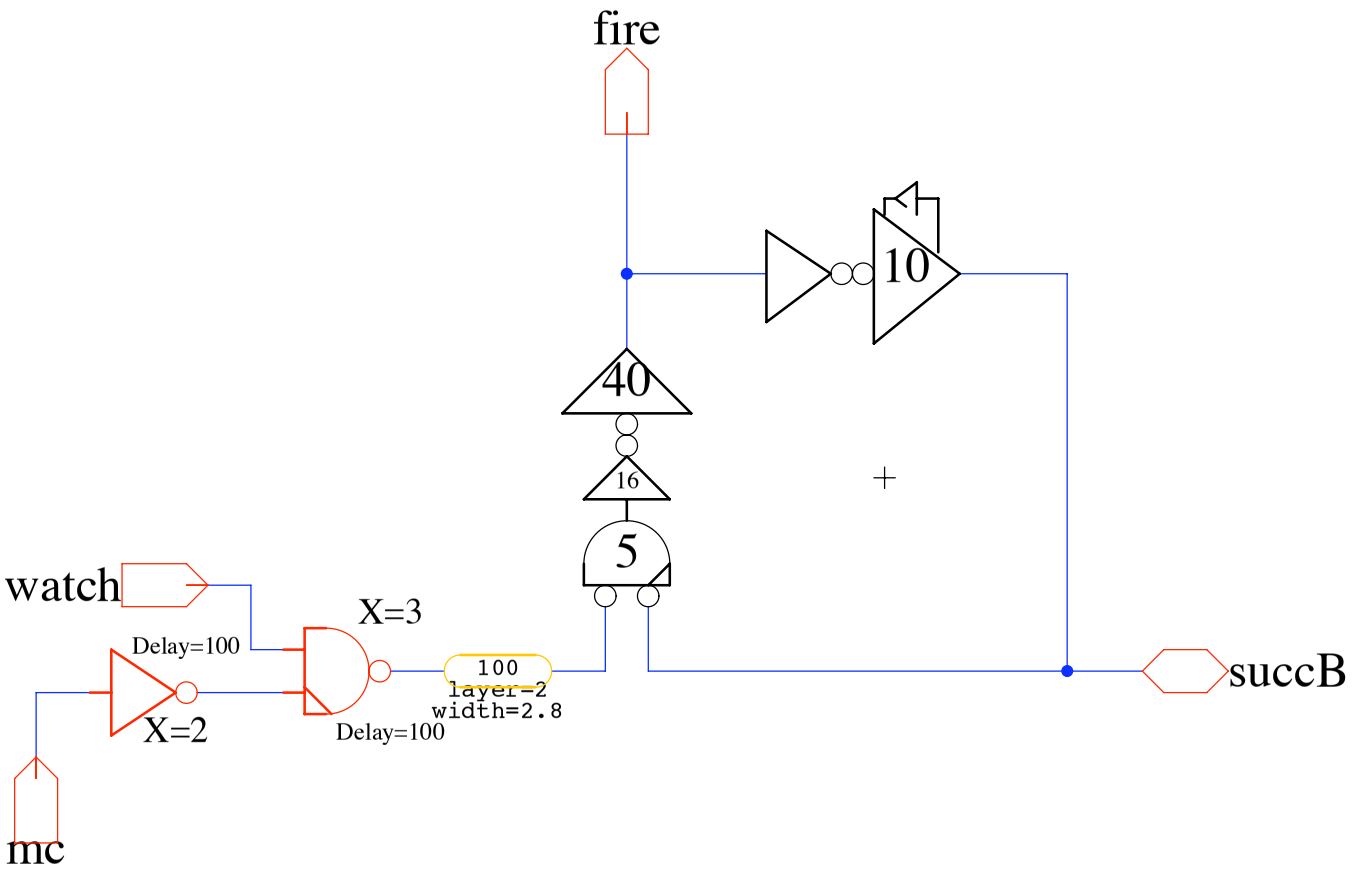




gaspSource10

ies 19 June 2010

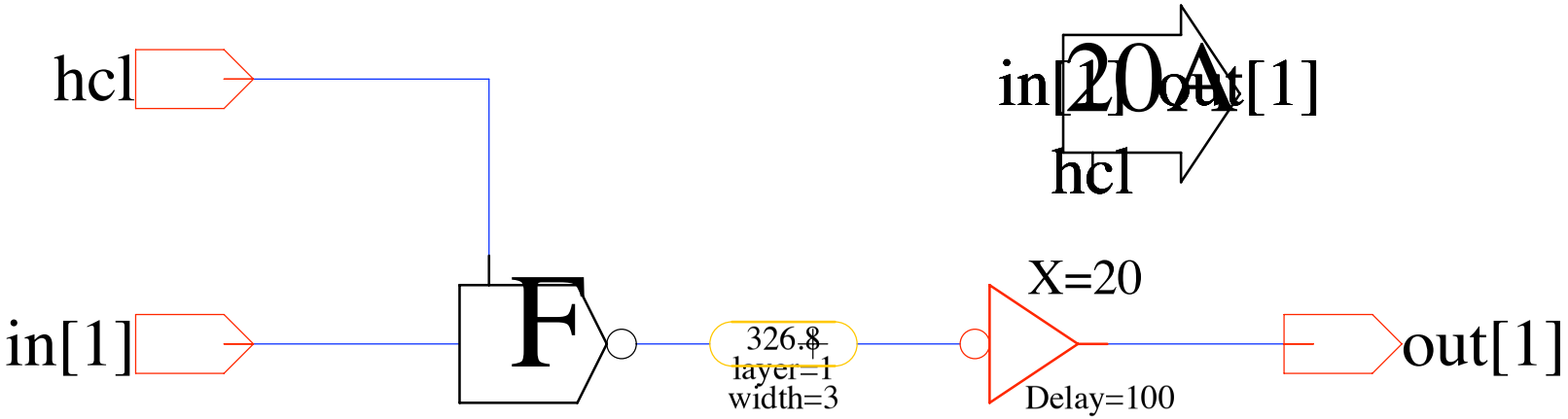




latch1in20A

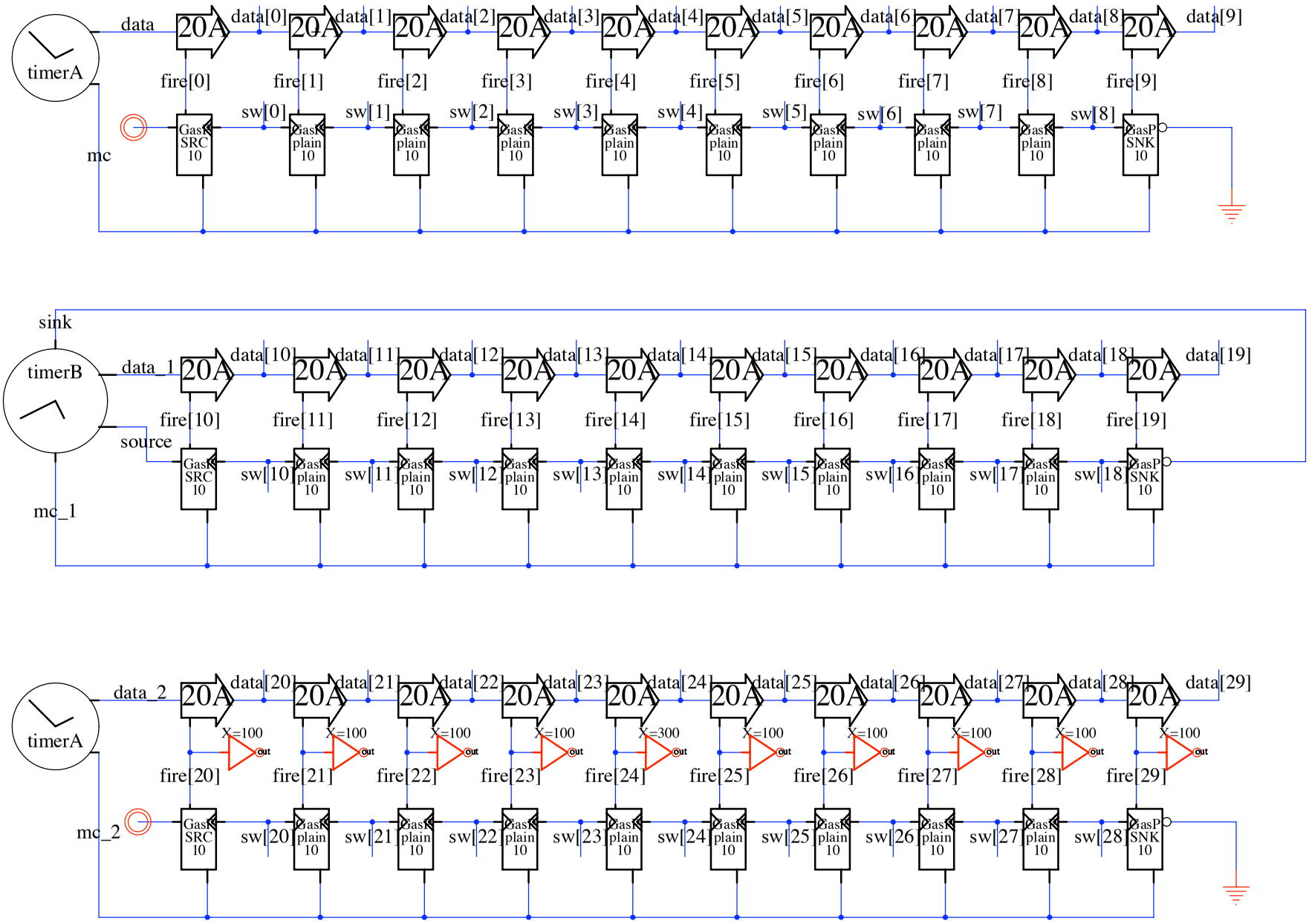
latch with one amplifier

ies 17 August 2010



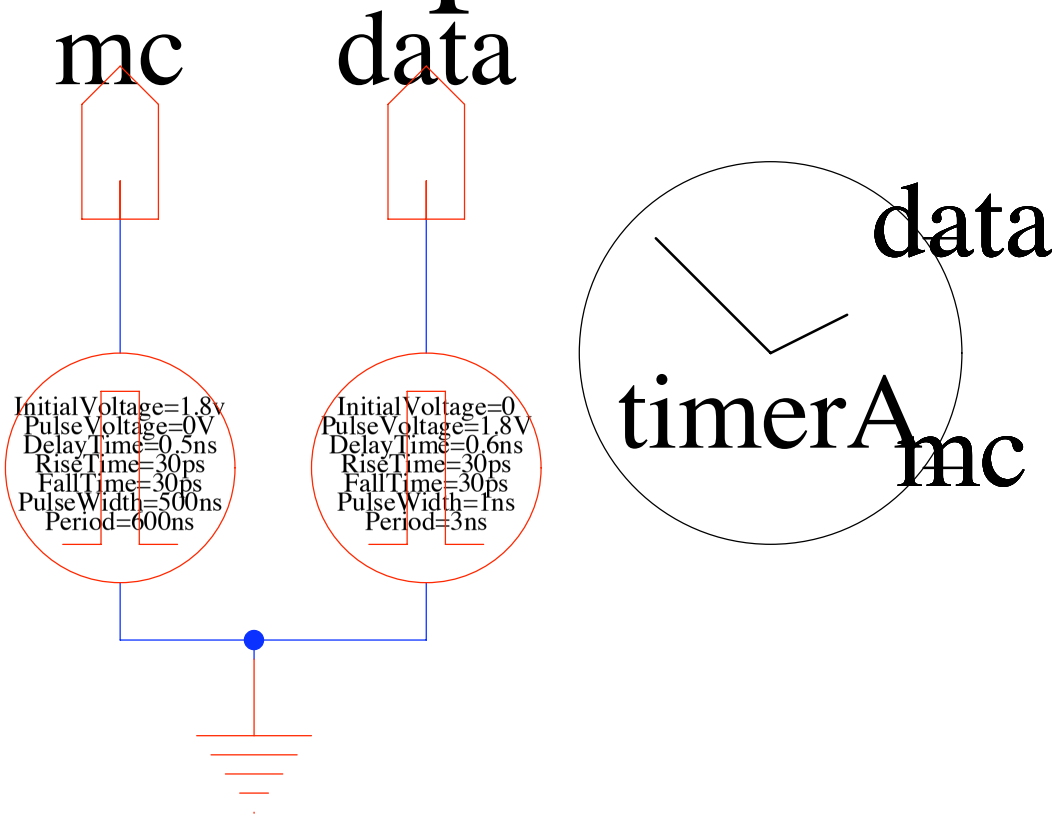
linearFifo

nsk 4 October 2010



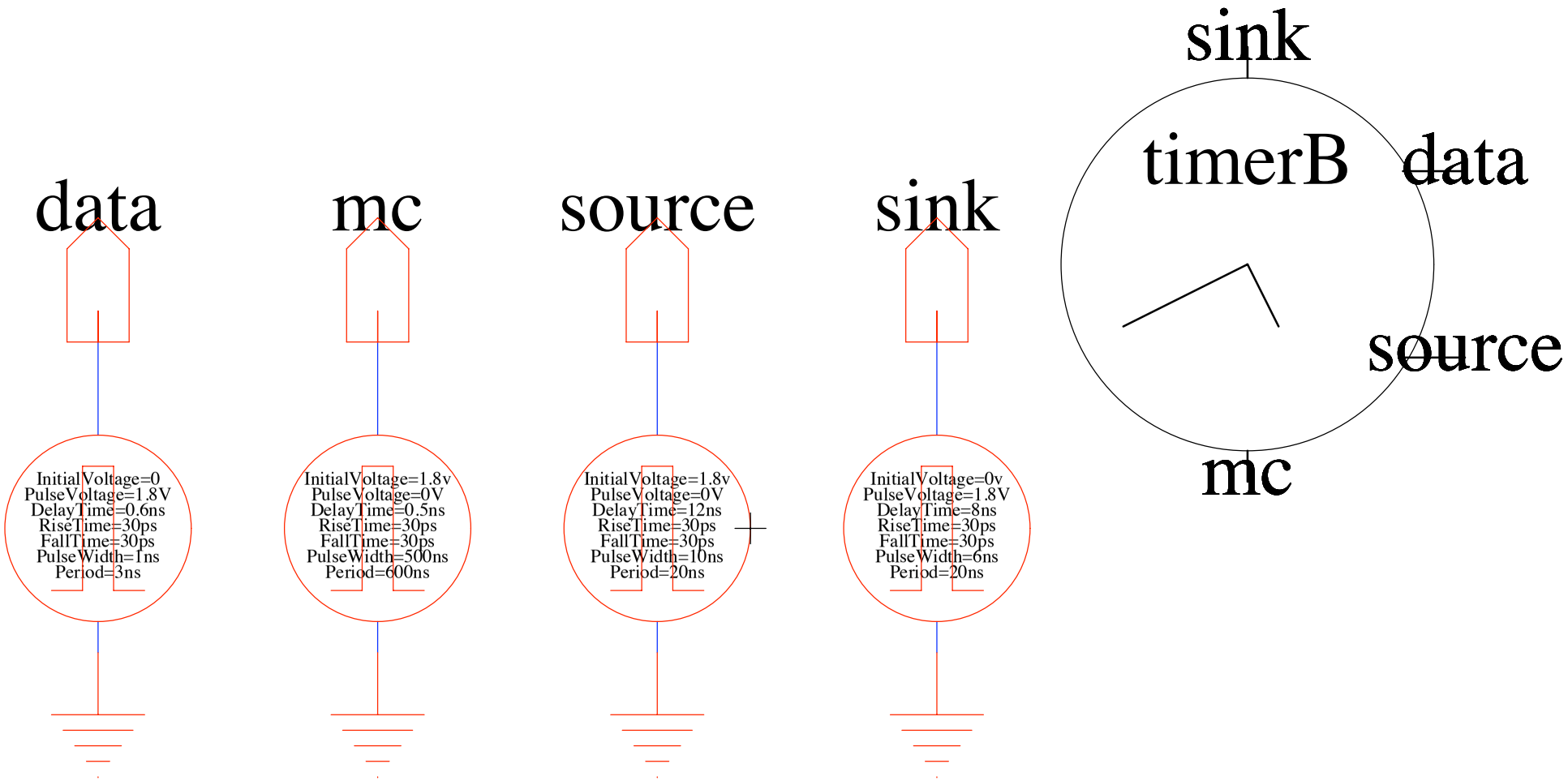
timerA

nsk 21 September 2010



timerB

nsk 21 September 2010



ANSWER SHEET – due by beginning of class on 19 October 2010 **3**

Name _____

Turned in on Date _____

Answer from simulation:

My simulation uses _____ technology – e.g. 180 MOSIS

My free-running FIFO operates every _____ psec for a throughput of _____ GDI/sec.
(GDI/sec is GIGA data items per second)

I observe that the fire signals of successive stages are both non-zero for _____ psec.

The forward latency of my FIFO for the 9 stages, `fire[0]` to `fire[9]` is _____ nsec.
(hint: Watch the data values flow to know which fire pulse is which).
The data advance rate of my FIFO is _____ psec/stage.

My FIFO accepted ____ data items at `fire[10]` before its sink stopped.

My FIFO delivered ____ data elements at `fire[19]` before its sink stopped.

Where are the excess data elements after the sink stopped?

The reverse latency of my FIFO for the 8 stages `fire[19]` to `fire[11]` is ____ nsec.
(hint: Look at `fire[11:19]` as the FIFO restarts after being full.)
The bubble reverse rate of my FIFO is _____ psec/stage.

My loaded FIFO operates every _____ psec for a throughput of _____ GDI/sec.
Measure at `fire[22]`. (GDI/sec is Giga Data Items per second)

Examine the two inputs to the AND function in stage [23].
The (rising) (falling) edges control the onset of `fire`. (Hint: measure the proper edge.)
Stage [23] predecessor signal arrives _____ psec (before) (after) its successor.

Examine the two inputs to the AND function in stage [25].
The (rising) (falling) edges control the onset of `fire`. (Hint: measure the proper edge.)
Stage [25] predecessor signal arrives _____ psec (before) (after) its successor.