Return to index

IBM'S OFTEN NONSTANDARD TERMINOLOGY AND PRACTISES FOR FLIPFLOP AND LATCH CIRCUITs

by Carl Claunch < Carl . Claunch @ gartner . com >

This section will serve as a 'Rosetta Stone' for readers familiar with modern terminology and digital design practices, helping them interpret the names and practices used by IBM, which can be confusing and thus tough to follow when reading documentation. Many nearly universal design principles that are used today are missing in the 1401 and other mainframes of the 1950s, 1960s and even into the 1970s.

One has to remember that the circuits, logic components and best practices for the entire industry were being pioneered back when these machines were designed. Many times IBM employees were the inventors of a principle but even in those cases when others first came up with an approach, industry terminology hadn't settled to a commonly accepted choice; the same idea would be described different ways depending on the preference of each company or academic researcher. Today, however, many of these terms are universally agreed, well defined concepts, albeit usually different from the original choices made by the inventors and by IBM engineers.

As an example, an IBM engineer invented the logic circuit family now universally named ECL (Emitter Coupled Logic), but the inventor called it current mode or current steering and that name continued to be used by IBM mainframe design engineers. The components used widely as flipflops in the 1401 and throughout the later 360 and 370 systems, were mostly called a trigger by IBM. The IBM trigger circuit does not correspond directly to any of the standard flipflop types that are covered in today's digital logic books and courses. The trigger circuit is flexible, able to be used in more than one way, each usage has similarity to common flipflop and latch types.

A flipflop or a latch is a memory device, used to hold some state or condition in a digital system. People often differentiate these by whether they are edge or level sensitive, allowing input signal changes to pass though and change the output state either just at the edge or during the entire time an enabling signal is active. A latch is level sensitive, in that modern usage, while a flip flop is edge sensitive. IBM, on the other hand, uses the term latch to refer to a combination of AND and OR gate components that act as a modern day latch, other times building a latch (as we would recognize one) using their trigger component.

Circuits that depend on a memory are generally called sequential logic, often implementing a 'state machine' which moves between defined states based on the

memory of its current state plus some input conditions that determine the next sequential state. Modern designers consider it a best practice to control the change of state in a flip flop or other memory in sequential circuits using the rising or falling edge of a special signal, a clock. This global clock is distributed to many gates simultaneously such that they all change their state together, in synchronization, at the clock edge. Flipflop components you would find today have a clock input that accepts this signal. The clocked flip flop holds its output state steady until the next clock edge, when it switches to a new state based on the then current input values.

The input conditions that exist at the time of the clock edge determine which state the output will take on. Proper operation often requires minimum periods of time that inputs must be stable at their intended values (setup time) prior to the clock edge and as well minimum times may exist that the input values must remain steady (hold time) after the edge has occured. The mandated duration to keep inputs steady is very short compared to the duration of a clock cycle, but if these minimum times are not met, the logic gate can fail to operate as intended or enter pathological states (e.g. metastable states).

Today many would say that a latch is a level sensitive device rather than edge sensitive one. That is, changes in the input can change the latch output state at any time the enabling signals attain an activating level. If a clock signal is hooked to a latch, it allows the latch to transparently pass input signal changes to the output for one of the two clock levels. During half the clock cycle, while the clock signal was in one of the two binary states, the latch would be changing its output continuously based on the other input signals, while in the other half of a clock cycle the latch freezes the output state that existed just as the clock level was reached. It acts as a memory for half a cycle and a bit of combinatorial logic changing in real time during the other half of a cycle. Contrast this to a clocked flipflop whose output state is frozen at all times except at the instant of the chosen clock edge (either when the clock is raising from 0 to 1 or the falling edge when it is dropping from 1 to 0).

If changes in input conditions can alter output states transparently, it has the same timing (race) hazards of combinatorial logic. If a designer intended that both 'set' and 'reset' signals were to be inputs causing an appropriate latch to toggle to its opposite state, but due to slight timing variations, one of the signals arrived first, the result could be the opposite state of what was intended since the level sensitive gate would first act on the single input and then change again when the tardy signal arrived. A change in the output state might make its way through other gates that affect input conditions, which could lead to unstable, glitchy or unintended operation. The practice of changing sequential states at a clock edge goes a long way to eliminate these risks. This practice of using a global clock to synchronously change states in sequential logic is essentially missing entirely in the IBM mainframes. In the era in which these systems were designed, the practice was not well established as it is today. Further, the cost of individual gates were very high, and the additional transistors and other parts needed to add clock synchronization would have driven up the size, cost and power demands of the system.

Latch and flipflops are now referred to with names like D, SR and JK, plus variants and extensions of these. Each type can be a level sensitive latch or, with additional components inside, operate as a clocked, edge sensitive flipflop. D (Data) type is the simplest, it sets its output condition to the state of its single input signal. T type (Toggle) devices will alternate output states when the T input is active. SR types (Set-Reset) (SR) have a set and a reset input, which operate as you would expect from the names of the inputs, but the behavior may not be defined if both S and R inputs are simultaneously active. J-K types are like SR, the J corresponding to a set and the K corresponding to reset but if both J and K are active at the same time, it toggles just like a T type. During the vacuum tube era, flip flops and latches were called triggers, but that is relatively obsolete terminology today. IBM engineers retained the name trigger for the circuit type even as it evolved to transistor and then integrated circuit designs. The trigger is IBMs building block for flipflops and latches. Their trigger circuit can be used as a level-sensitive or an edge sensitive device, and it is most analogous to a JK type.

IBM refers to the edge sensitive operation of the trigger as "AC" mode and used a capacitor signal on the input line to indicate this on SLT era logic diagrams but not on the 1401 generation diagrams. The letter assigned to the input on the component on 1401 diagrams indicates its role - the G input is the DC mode that is level sensitive to set or reset the trigger. The A and C inputs are the edge trigger and gate for AC mode operation. However, this AC mode use is not a clocked JK, because the J and the K sides of the trigger are individually edge sensitive. In a clocked JK flipflop of today, one clock is used for the edge that activates the function requested by the J and K inputs. If J is on, K is off, then the flipflop is set on the clock edge. If the J and K are both on, the flipflop toggles. With the IBM trigger circuit, an edge on one line drives the output change based on the gating input signal, but as there are two different edge inputs, there are essentially two independent sides for J and K.

The reset (K) gating input may be on but it would not reset the output of an AC mode trigger if the reset triggering input did not present a rising edge. At that same instant, if the set (J) gating input is on and its set trigger input rising edge has arrived, that would be setting the output state of the flipflop. With a single clock on a JK flipflop, having both J and K inputs on would cause a toggle when the clock arrived. On the IBM AC mode trigger, if both sides are not triggered simultaneously, only the triggered side would affect the gate, either setting it or resetting it, ignoring the untriggered input. This behavior is not one that would be familar to a modern digitial designer and could lead you to misunderstand the behavior of a 1401 or 360 design.

When IBM was putting a latch into a circuit, the engineer could use either a pair of AND/OR gates or the trigger circuit in DC mode. For the 1401, the engineers preferred to use the pair of combinatorial gates whereas in the 1130, the preference was more to the use of the trigger component, and in 360 it varied by model and by the portion of the machine. Thus, these seemed to be stylistic predilictions of individual designers. There are very valid reasons to make the choice on a case by case basis, especially with this higher circuit density of SLT where one may have

spare AND/OR gates on a card already being used, while use of a trigger could mean an incremental card is needed; obviously, the opposite situation could equally pertain. With SMS, this is less likely to occur because the density is much closer to a one to one ratio of logic function and card than with SLT.

The trigger circuits were not exactly the same across the various logic families in SMS nor between SMS and SLT, adding another small complication when reading logic diagrams of those eras. The current mode families of SMS are more generally operating as edge sensitive devices, than do the CTDL or SLT trigger components.

The term binary trigger is used with current mode trigger circuits for essentially an edge sensitive T latch. DC mode trigger was used to mean a gate that sets if both inputs are on, resets if both inputs are off, but does not change state if the inputs are mixed. This is not analogous to any standard flipflop type today, but would provide some protection from race hazards where one input lagged the other; during the short interval where the inputs were mixed, the binary trigger would do nothing, but when the pair of inputs were both present it would set or reset as selected. DC mode Slit P is a term used with current mode triggers for an SR latch with a reset line added. Bipolar trigger is the term IBM used for a D latch, the where an edge on one input would cause the latch to take on the state present on the other input at that instant. It does not transparently pass the input to the output at any time, thus is not exactly like a D latch or flipflop.

CTDL family triggers are gates with individually 'clocked' J and K inputs, each side with an independent data input and a gating (edge sensitive or AC) input. The CTDL trigger also has asynchronous reset and set lines (DC set and reset). When the IBM SLT trigger is used with both its set and reset inputs tied together, it is acting like a T flipflop and would be referred to by IBM as a binary trigger. If the DC mode set and reset lines are used, they operate like an SR flipflop and are level sensitive. They can also be thought of as the asynchronous set and asynchronous reset that can be added to the basic flipflop/latch types. More often than not, the trigger circuit is used as a level sensitive SR latch, using just the DC model (level sensitive) inputs. As you can see, IBM tended to use DC and AC rather than level sensitive and edge sensitive terminology.

</