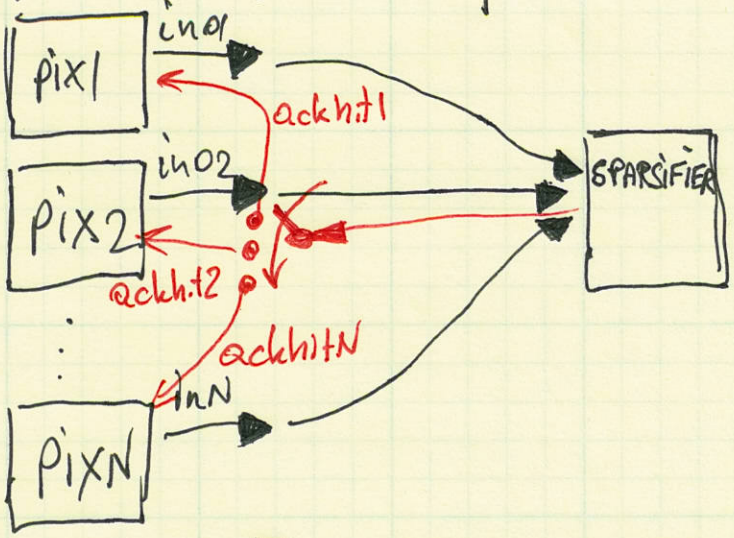
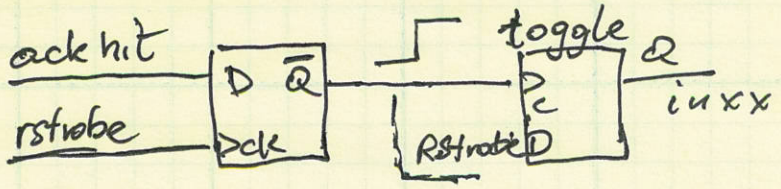
VIPIC – LARGE
ANALYSIS OF RStrobe and RStrobe-less solution
VIPIC 1 – first brief look

(1)



- At every ↓ of TSClk pixels with hits activate their inxx lines
- Automatically the sparsifier switches on 'low state' to pixels ~~acco~~ according to its priority order. So when pixel is chosen it sees a change ↓ on its ackhit line
- pixel cannot know which pixel was readout before – it just knows: 'it is your time now'
- the reverse path of the sparsifier looks like a switch that connects 'low state' to a pixel with hit
- Rstrobe signal is sent to all pixels simultaneously and it is a short pulse, which rising edge latches all current states of ack_hit signals (in every pixel)



important is to notice that:

— pixel data is released immediately when ackhit goes down for a given pixel

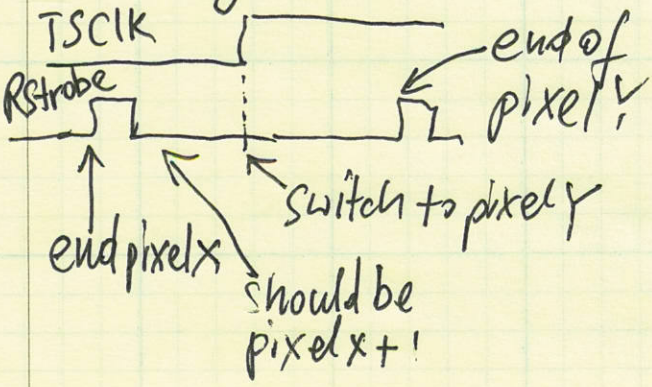
— Rstrobe, sent globally, finds a pixel that is currently outputting its data

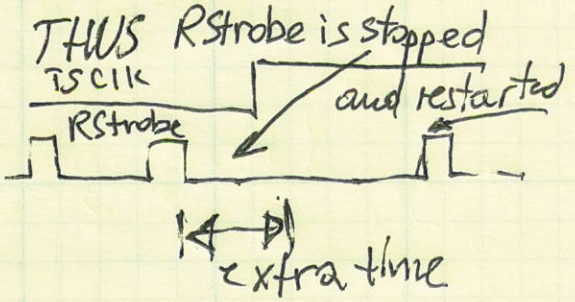
and:

- when latching ackhit 'low state' generates pulse ~~toggling~~ toggling the inxx pixel
- creates a reset RStrobeD strobe that is used to delete the contents of the counter that was just read out. (it can be noticed that duration of RStrobeD will be equal to time between RStrobes)

TSClock introduces some discontinuity in the readout process, because ↑ on TSClk can move ackhit 'low state' to another pixel than the one that was in the process of reading out

DANGER! This newly pointed pixel may not have enough time to settle its outputs for a serializer



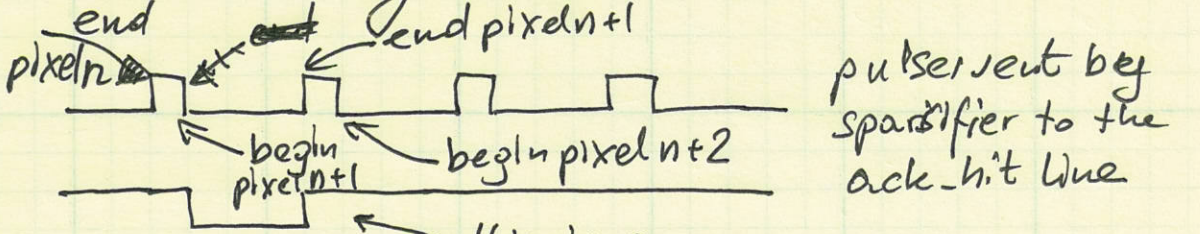


VIPIC LARGE

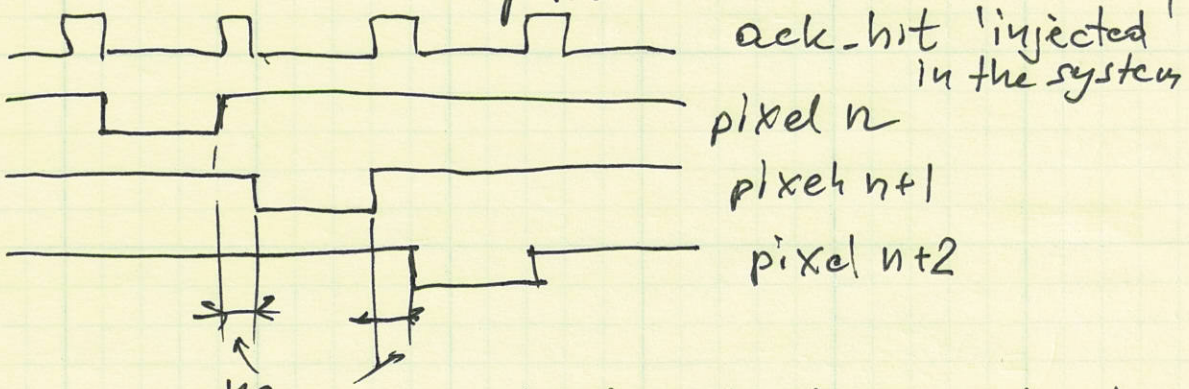
- RStrobe in VIPIC1 is a fast signal that has to be sent to every pixel (simultaneously) one pulse Rstrobe for reading data from single pixel — thus its: routing, capacitive load, interference } are critical
- if RStrobe is eliminated, the design of the sparsifier is simplified
- To eliminate RStrobe pulsing of ack_hit line can be used to signal end of data readout from a pixel and moving the priority encoder to the next pixel
- Because of the high speed of serialization in the VIPIC-L chip, it would be ~~desired~~ desired to run the serializer synchronously of the high speed clock (up to 400MHz.)
  - so, the signalling of end of data readout should be synchronous and continuous too
- It would be also desired to decouple division of acquisition into time frame 'TSClk' from the sparsified readout and data serialization. Period of TSClk can be varied in a huge range and the TSClk can be provided from outside. (in principle without requiring its period to be related with the serialization clock)

- IMPORTANT NOTICE

IF such a pattern is ~~send~~ sent on the ack_hit line, a pixel that is outputting does see only that

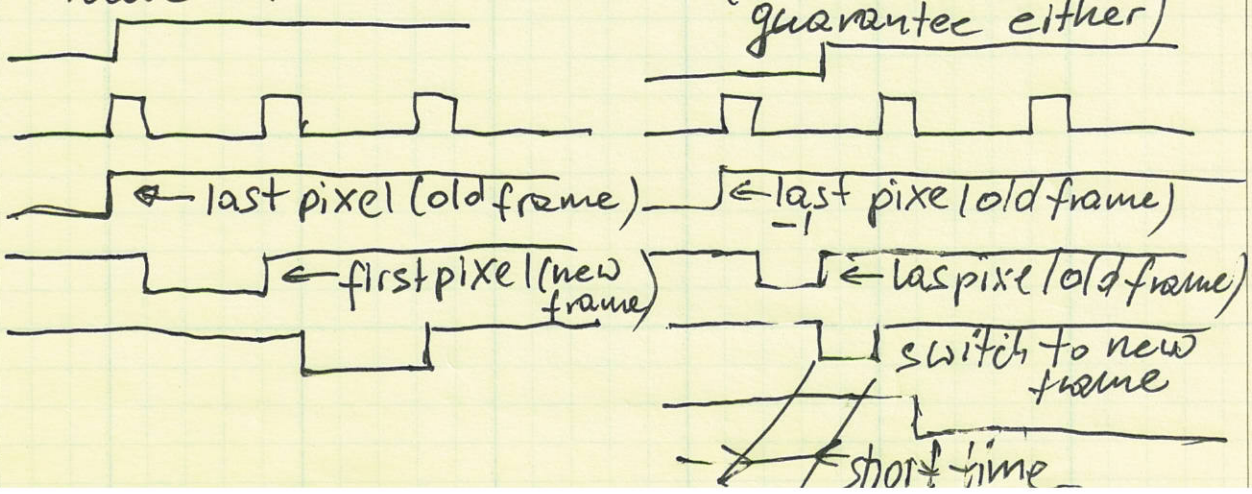


this is an effective ack_hit signal that is experienced by a pixel



- there are no miracles

if TS CLK $\frac{1}{4}$ is added we may have two situations (but we cannot guarantee either)

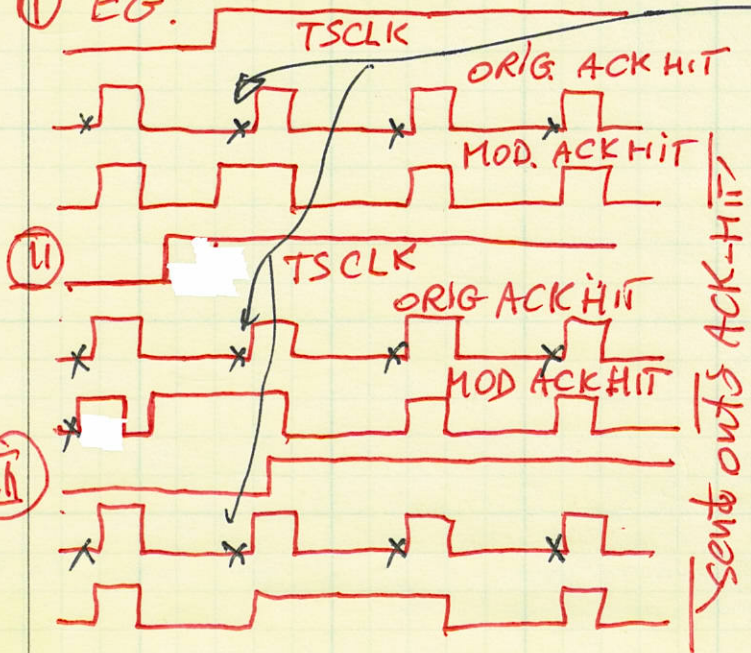


— IN ORDER TO
- KEEP ACK_HIT PULSES NOT TO ADJUST THE TEMPO OF STEPS AROUND THE TSCLK ↓
- ALLOW TS_CLK ANY TIME
- NOT TO LOOSE FIRST PIXELS FROM THE NEW TIME FRAME (FIRST PIXELS ARE MORE IMPORTANT THAN LAST FROM OLD FRAME)

ACTUALLY AFTER LONG TIME SPENT ON ANALYSES I'M CONVINCED THAT THIS IS THE ONLY ONE POSSIBLY TO BE USED IN UI0/C-L

- PULSE ON ACK_HIT MUST BE STRETCHED FROM THE DETECTED ↓ OF TSCLK TO THE ↓ OF THE ORIGINAL ACK_HIT (NEAREST)
- DETECTION OF ↓ TSCLK NEEDS TO FORCE COUNTER VALUES BEING LATCH BY SERIALIZER BE FORCED TO ALL '0' — WHICH MEANS 'NO MEANINGFUL DATA'

① EG.



will latch counter = '0' 'will let know 'useless data'

× — latching of data by serializer

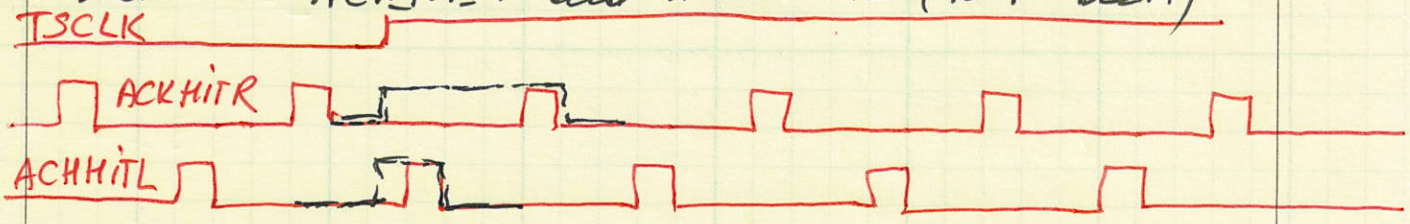
* times at which words are latched are not changed

STRETCHER MUST MONITOR TSCLK WITH ↓ OF ACK HIT

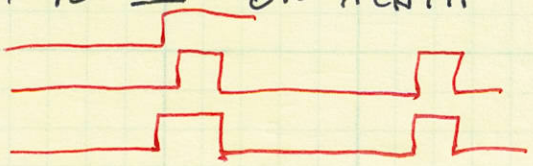
AFTER IMPLEMENTING SUCH A STRETCHER THE HIT HOLDER LOGIC IS SIMPLE AND RELIABLE

– THERE IS SOME INEFFICIENCY INTRODUCED IN THE SYSTEM (FOR A STRETCHED ACK_HIT NO USEFUL DATA IS ~~SEE~~ SENT OFF THE CHIP) BUT IT IS ~~HOPEFULLY~~ UNAVOIDABLE ! IT IS DUE TO THE TSCLK BREAKING THE CONTINUITY OF THE TIME AXES

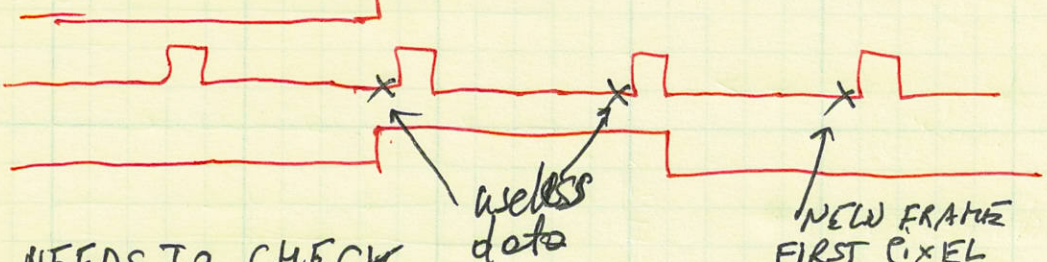
– IT IS KNOWN THAT THERE ARE TWO INTERLEAVED SIGNAL ACK_HIT_R and ACK_HIT_L (NO PROBLEM)



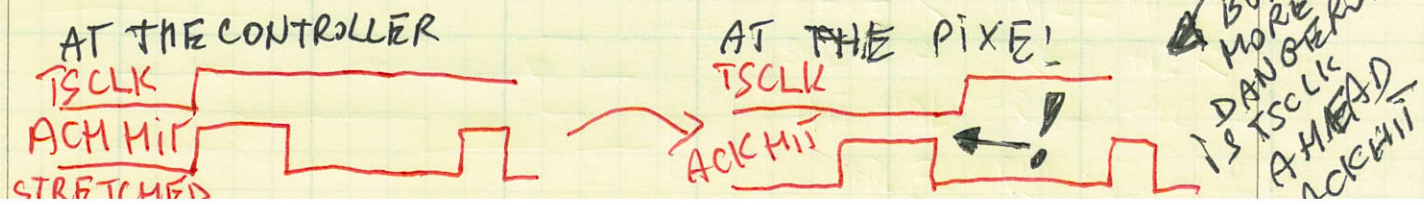
PROBLEM MAY BE ANTICIPATED IF TSCLK CHANGES VERY CLOSELY TO ↑ ON ACKHIT



MAYBE IT WOULD BE BETTER TO REQUEST STRETCHING ~~FOR~~ ~~UNTIL~~ UNTIL 2nd ↑ IS DETECTED ON ACKHIT

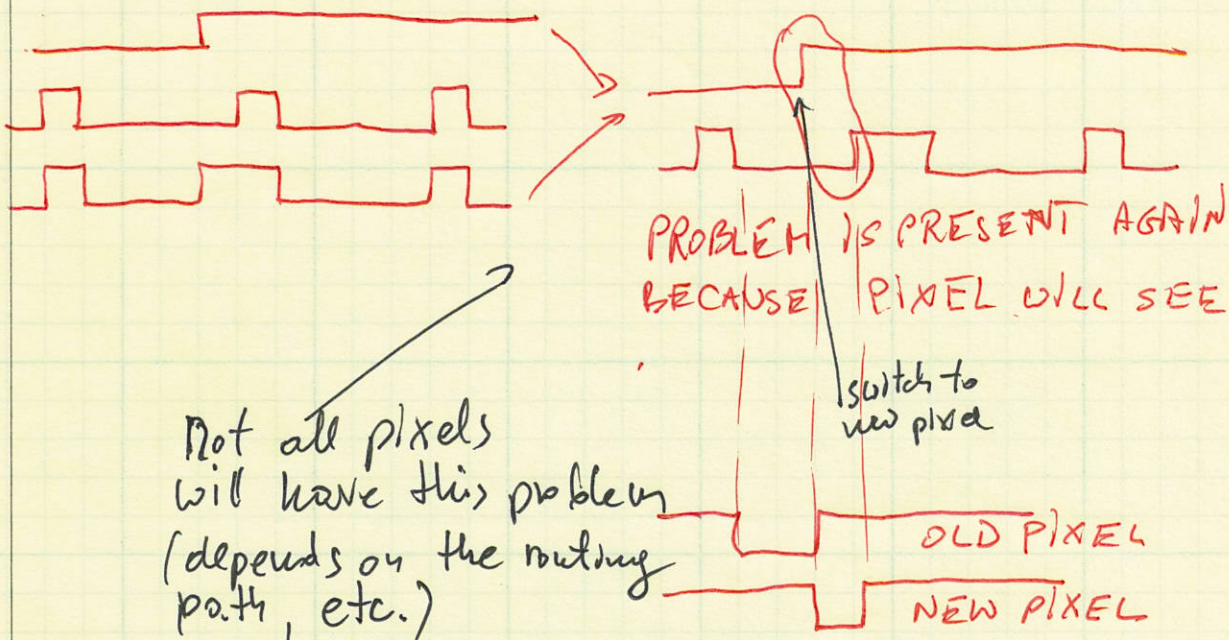


ONE NEEDS TO CHECK CAREFULLY DELAYS TO AVOID FALLING EDGE OF ACK_HIT RACING WITH RAISING EDGE OF TSCLK AT ANY PIXEL

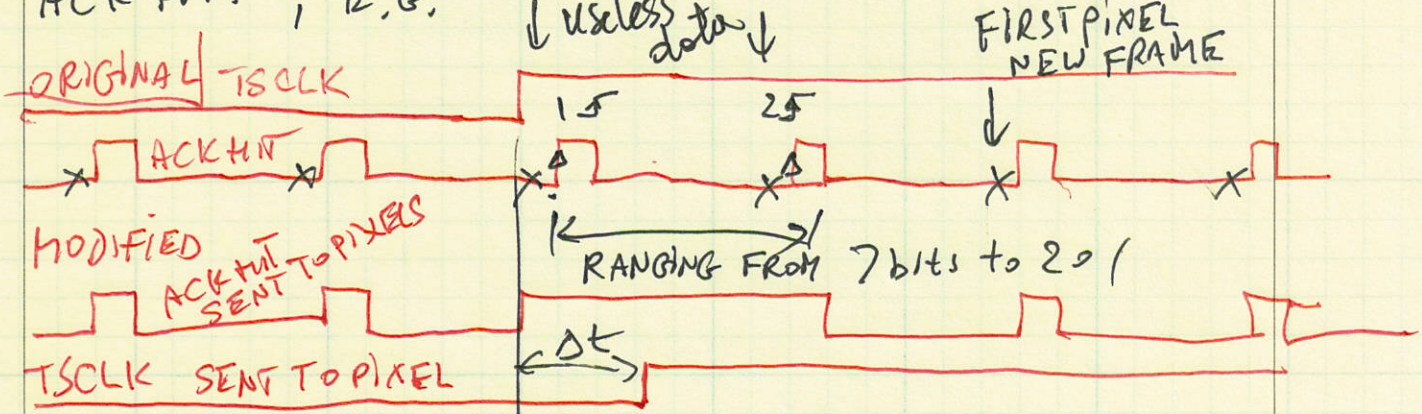


AT THE CONTROLLER

AT THE PIXEL



THUS MAYBE STRETCHING OVER TWO RISING EDGES OF ACKHIT IS REQUIRED AND SUCH ADJUSTMENT OF THE RISING EDGE OF TSCLK THAT WILL NEVER RACE OFF ACKHIT, E.G.



- SOME CHANGES REQUIRED IN THE CONTROLLER
- BUT ALL CAN BE KEPT SYNCHRONOUS (ACK_HIT, SERIALIZATION, OUTPUT)
- THERE IS NO FREE LUNCH

delayed by some number of sensitive clock cycles

needs to be found with delay analysis

Δt should be longer than any delay difference that can be experience in the matrix that TSCLK CAN ARRIVE BEFORE ACK_HIT!