Course on Caeleste's in house SAR ADC

History record

20161011	Bart	started PPT, largely empty
20220922	SCIB team	added chapter4 and list of future improvements
20220926	Bart	edits, reordering chapters
20220926	Arne	incorporate earlier course for Test team
20221005	Bart	complete revision after ATON NCR07
20221006	SCIB team	offchip mitigation of odd-even effect
20221007	Bart	presented first part at CU
20221104	Bart	presented remainder at CU
20230127	Bart	created separate section on the non-binary SARADC
20230315	Ahmed	preparation of review
20241014	Bart	cleaned up and streamlined the section on SASC

purpose

This course is about Caeleste's in-house SARADCs.

- Primer and reference on the basic concepts
  - For the designer of on-chip SARADCs
  - For the test engineer evaluating these
- Overview of building blocks and features
  - In dept discussion on issues
- Paths to future improvements

Disclaimer

- This is not a general course on ADCs
- An onset of general course is in BP049, "general ADC design"

History of Caeleste SARADCs

caeleste

Generation 1 was a 16-bit traditional SARADC designed for Campanion1

Generations 2-9 and 11 were 12-bit SARADCs with improvements as unary/binary/RDAC, postcorrection and double conversion. They had in-ADC regulation and oversampling. It used interleaving groups.

Up to SARADC generation 9, the architecture was essentially a binary code SARADC

As a consequence of NRC07 or ATON, radical improvements were proposed, under which a new approach to substitute the unary/binary CDAC by a pure capacitive non-binary CDAC.

Generation 10 is the non-binary SARADC. The reason to move to non-binary is self-correction of conversion errors and code gaps. Secondly it has a much more compact sequencer and logic, simpler and smaller CDAC, faster operation and lower power. It has however a calibration burden.

Then came Helena. Helena inherited much of the Generation 10 and before, however it is conceived for low EPC and high CSA. It is built around the “SASC” library. The SARADC is of the monotonic switching type. It abandons in-ADC regulation and replaces it by the “split capacitor” trick. It allows both binary and non-binary implementations. It abandons interleaving.

Table of contents

caeleste

outline

caeleste

1. Binary SARADC
2. Non-binary SARADC
3. Helena

Student prior knowledge

caeleste

Study relevant parts of CAMPANION1&2 deliverables

Look into Notebooks 0022, 0032, 0045, 0226, 0287, 0516, 0635, 0645, 0746, 0769

See also the paragraphs on ADC in BP031 Measurement procedures

- Review all quantities measured and understand what they mean.

Goal of this course

caeleste

Understand why we chose a certain SARADC concept

- Why binary and why go to non-binary?
- Why differential? Why pseudodifferential?
- Why interleaving? And full parallel?

Understand the CDAC concept, how to construct a CDAC

- Why unary, binary, RDAC,
- Why non-binary and how to calibrate that?

Understand the comparator, clocked or not clocked, its auto-zeroing, its output signals.

What is double conversion, what is postcorrection and how to solve the odd/even effect

Why and how can we do oversampling

Understand the array aspects

Understand and contribute to future improvements

- Improving speed / power
- Increase resolution
- Reduce large conversion errors, reduce noise, improve DNL/INL, improve PSR
- Strive to a true standard cell, with simple parametrization.

section

1. Binary SARADC

Generation 2-9 and 11

Goal of this course

caeleste

Understand why we historically chose the binary SARADC concept

- Why differential? Why pseudodifferential?
- Why interleaving? And full parallel?

Understand the CDAC concept, how to construct a CDAC

- Why unary, binary, RDAC,

Understand the comparator, clocked or not clocked, its auto-zeroing, its output signals.

What is double conversion, what is postcorrection and how to solve the odd/even effect

Why and how can we do oversampling

Understand the array aspects

Understand and contribute to future improvements

- Improving speed / power
- Increase resolution
- Reduce large conversion errors, reduce noise, improve DNL/INL, improve PSR

caeleste

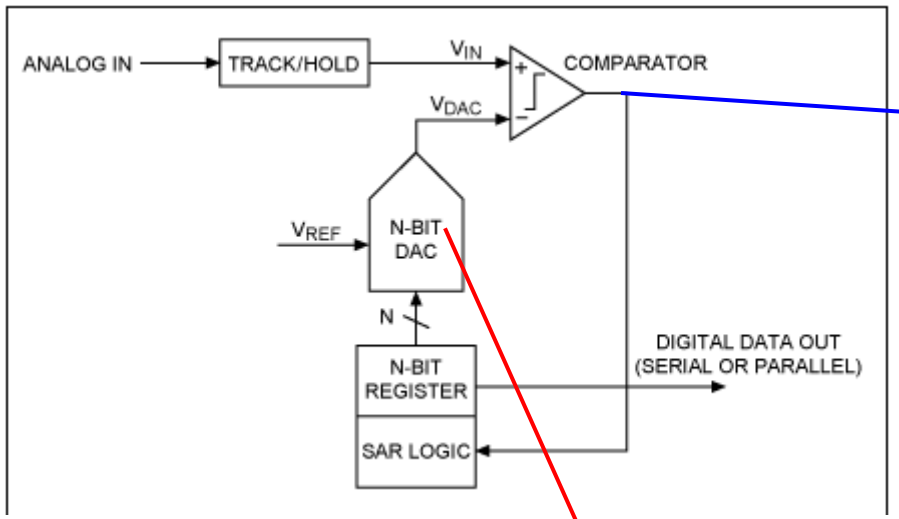
SARADC Concepts

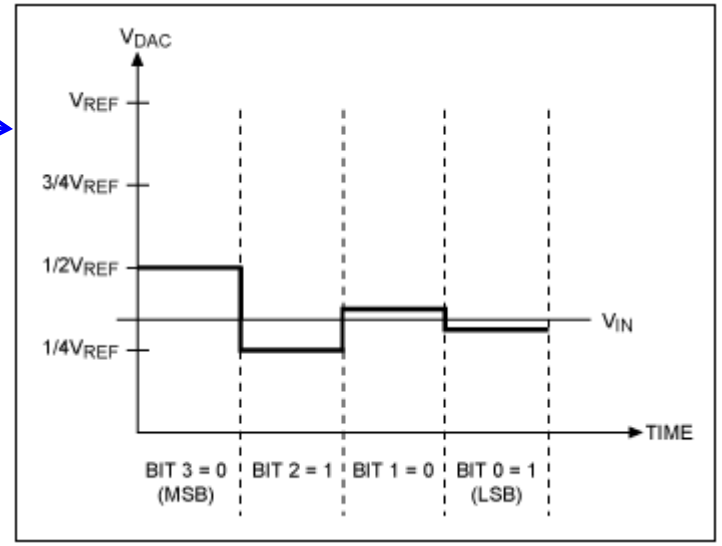
SAR ADC principle of operation

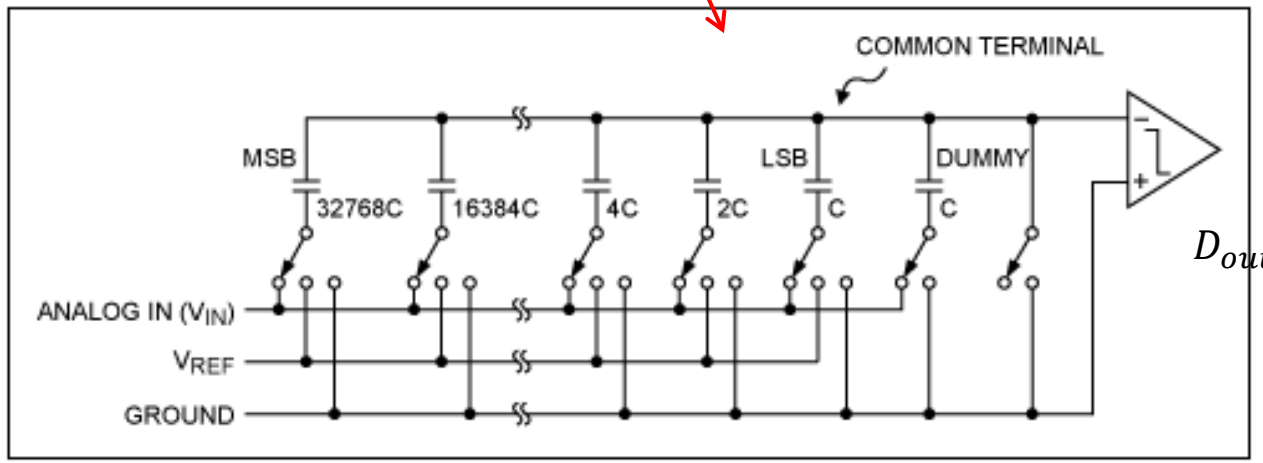
caeleste

https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1080.html

Textbook example of single-ended Successive Approximation Register (SAR) ADC with a Capacitive DAC (CDAC)







$$D_{out} = CalVal - 2^{12}\left[\frac{signal - reference}{2 * (V_{high} - V_{low})}\right]$$

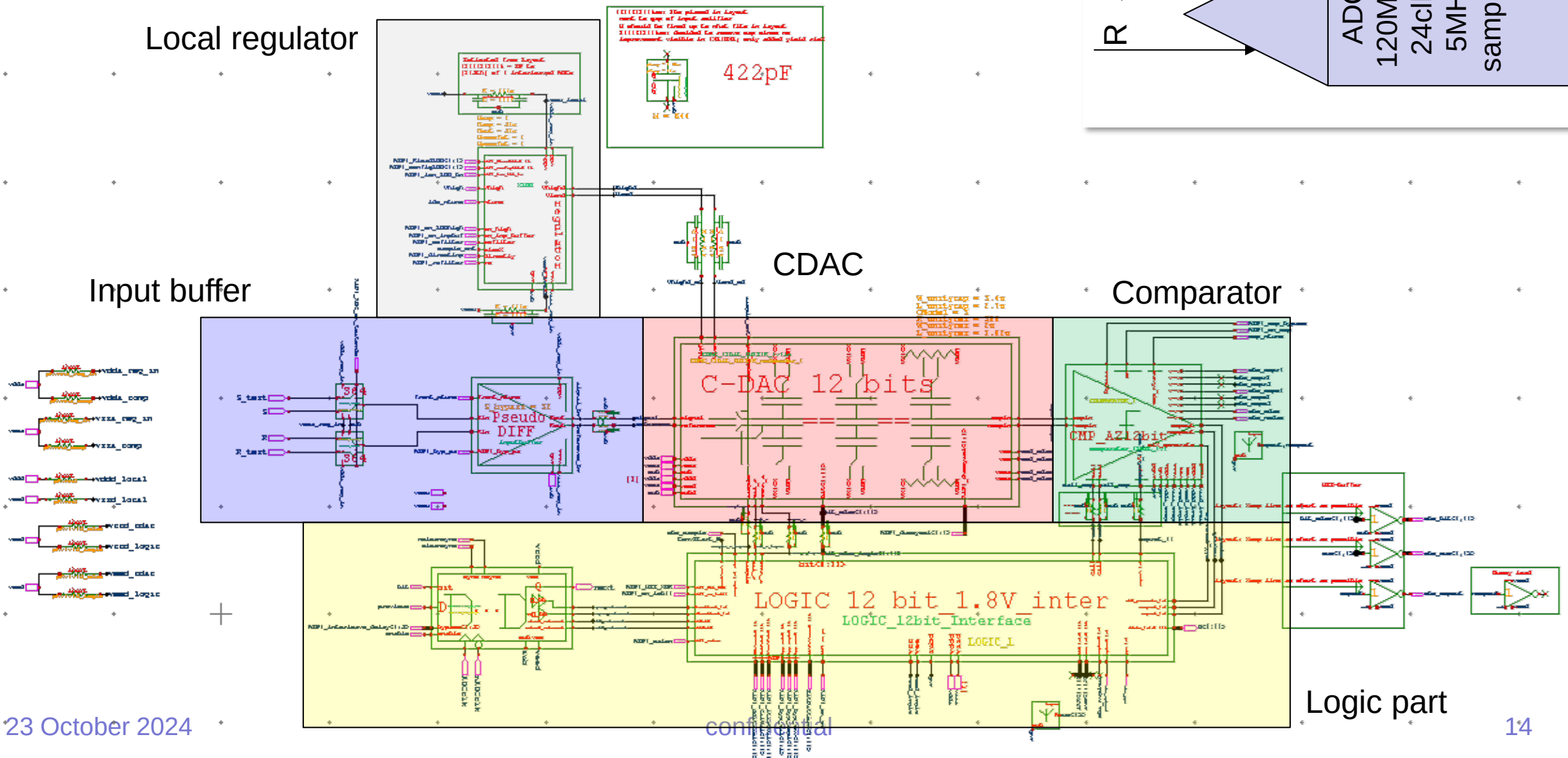
Why our SARADC concept

caeleste

- The Successive Approximation Register ADC concept
  - Is widely used (“thus it must be good!” – this is *not* a good reason)
  - It is often reported by others with the best FOMs – this is a good reason
    - Especially conversion rate per unit area, for resolutions in the order of 12 bit
  - Is easily portably as it only needs plain CMOS and symmetric (MIM, POD) capacitors.
  - Accepts the “native” pseudodifferential signal of imagers. And then performs CDS.

- Interleaving
  - Fits nicely on top of interleaving X-scanning buses as also used in image sensors.
  - Elegant solution to output multiple sub-ADCs on a single serial stream.
  - Equalizes the peak current on the CDAC reference voltages

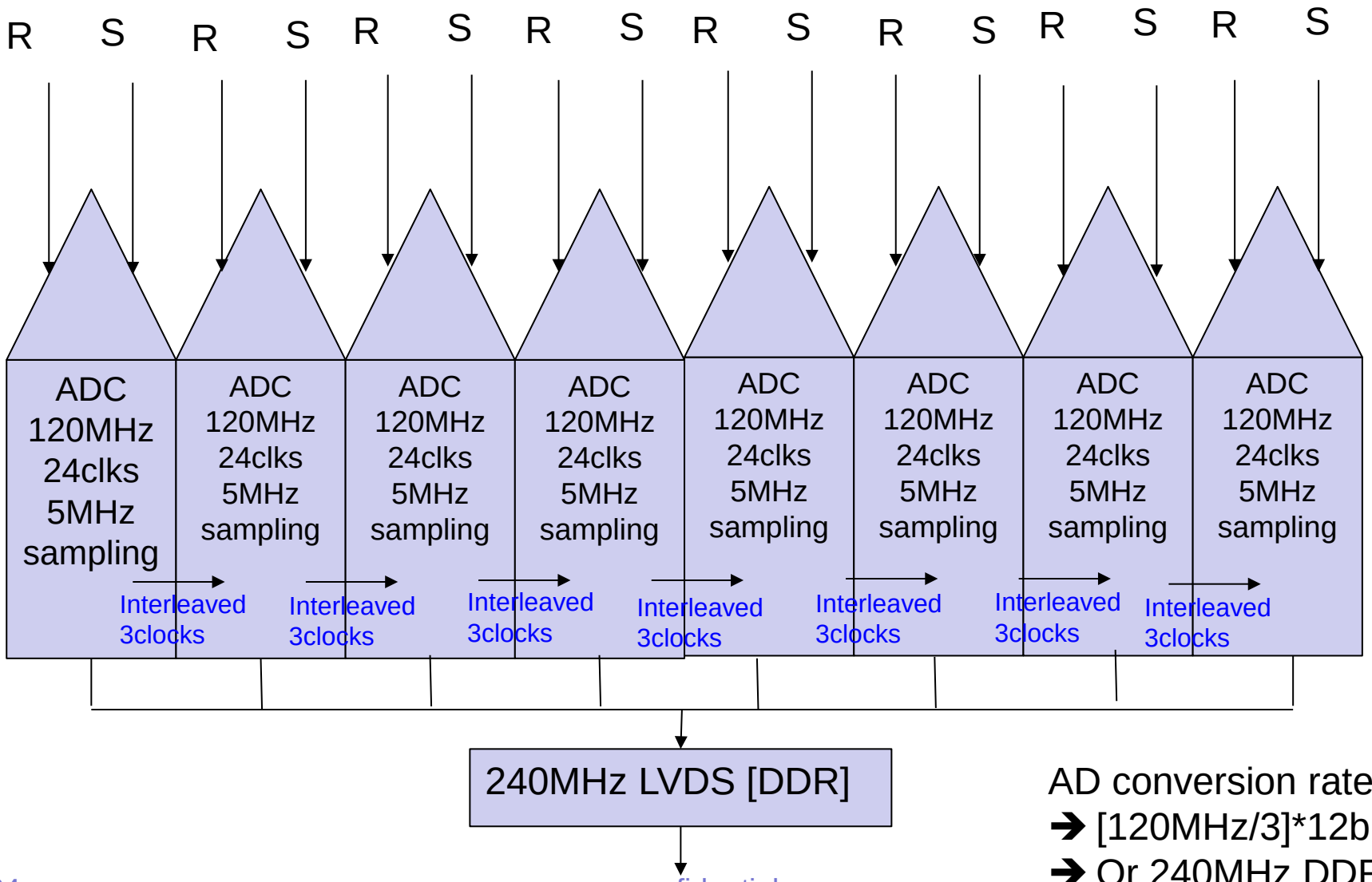
SAR ADC Caeleste implementation generation 9



SAR ADC group of 8 interleaving

caeleste

8*5MHz sampling = 40MHz sample rate ADC



ADC clock 120MHz

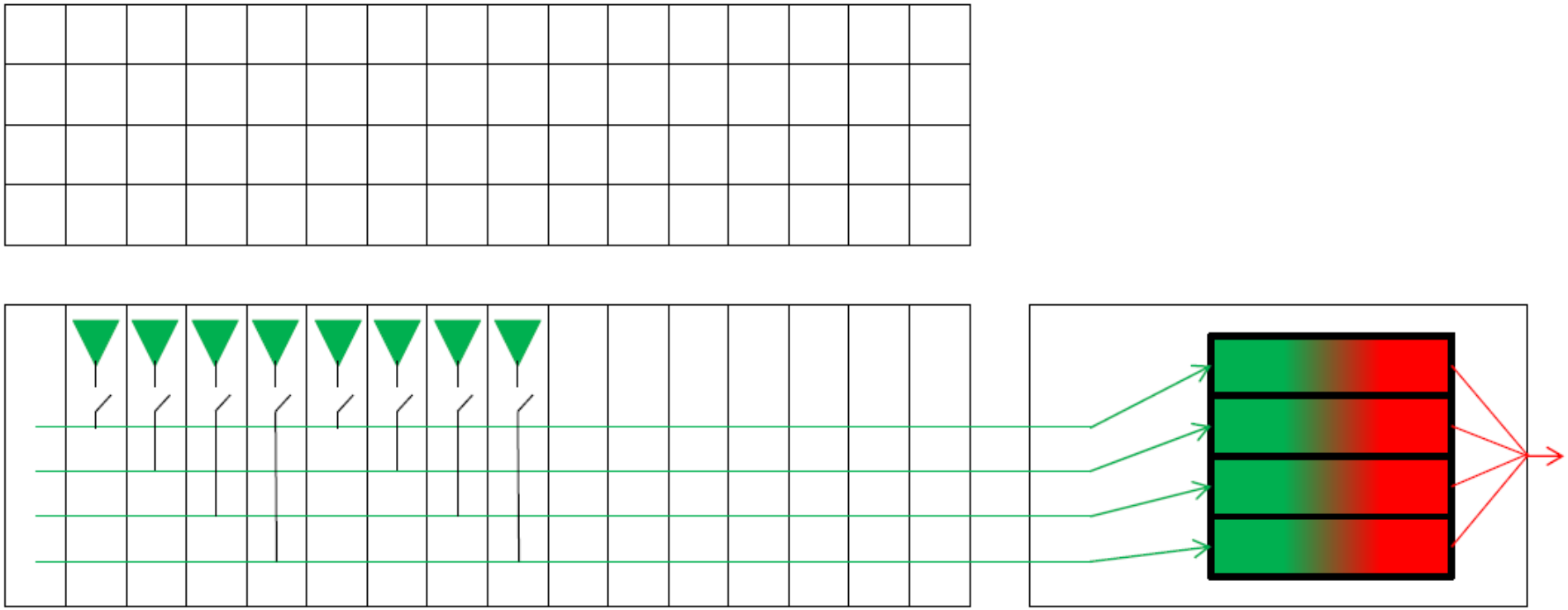
AD conversion rate is 120MHz / 3
➔ [120MHz/3]*12bit = 480MHz
➔ Or 240MHz DDR

Interleaving/interleaving

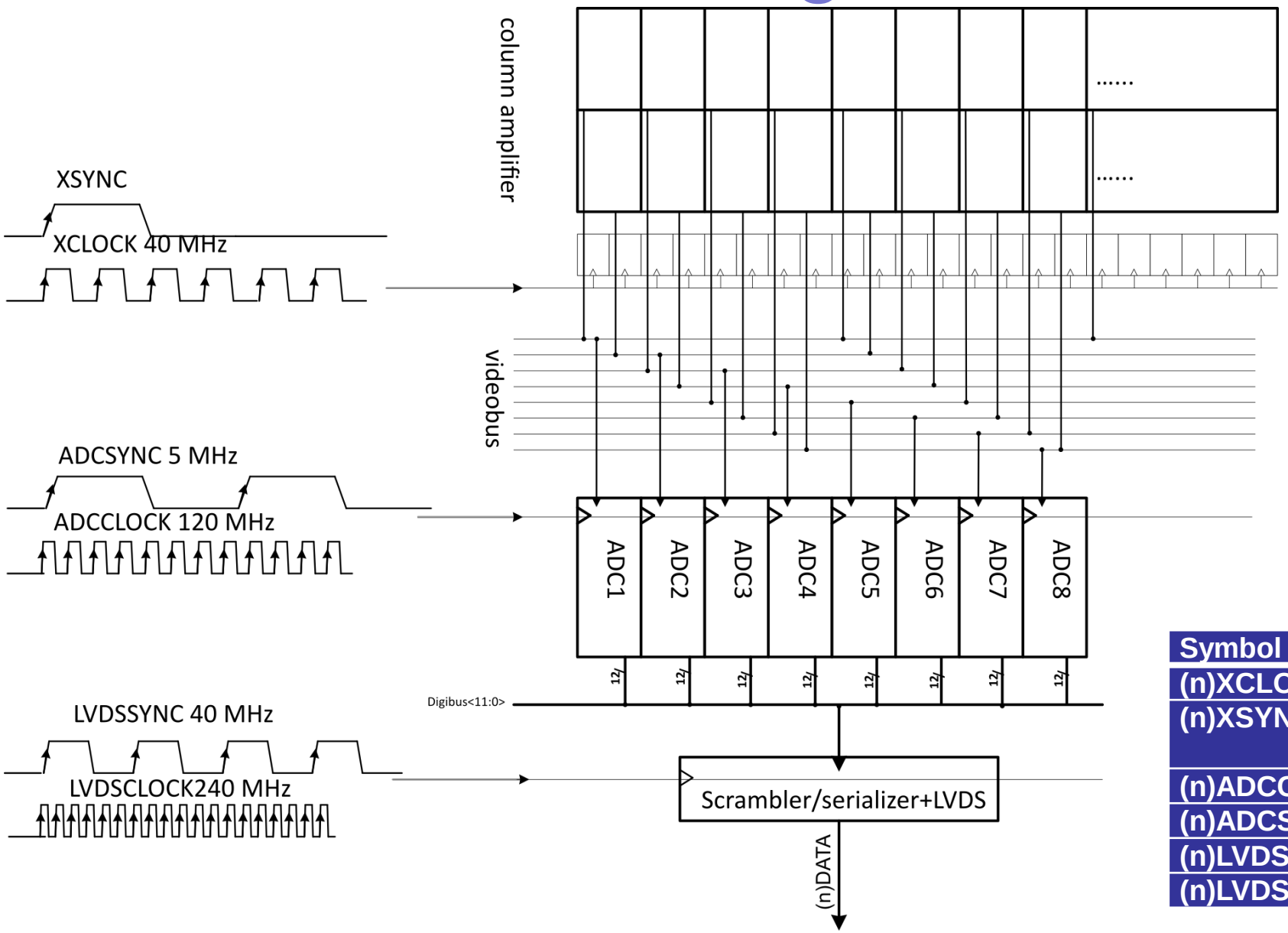
caeleste

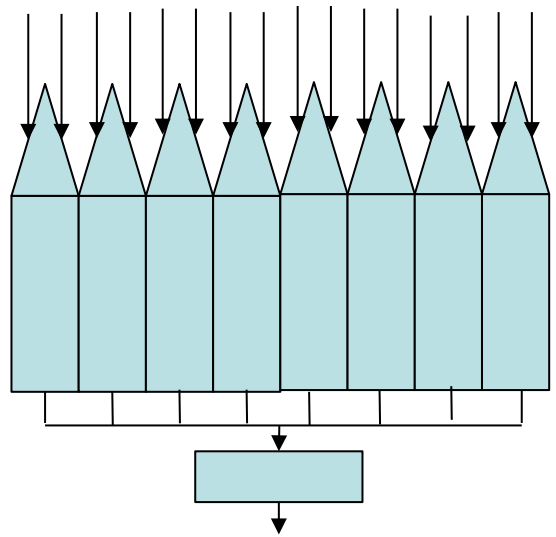
As in NOTEBOOK 0032:

Interleaving ADCs is especially convenient if one can combine it with an interleaving video bus.



SAR ADC interleaving matched to X-scanner





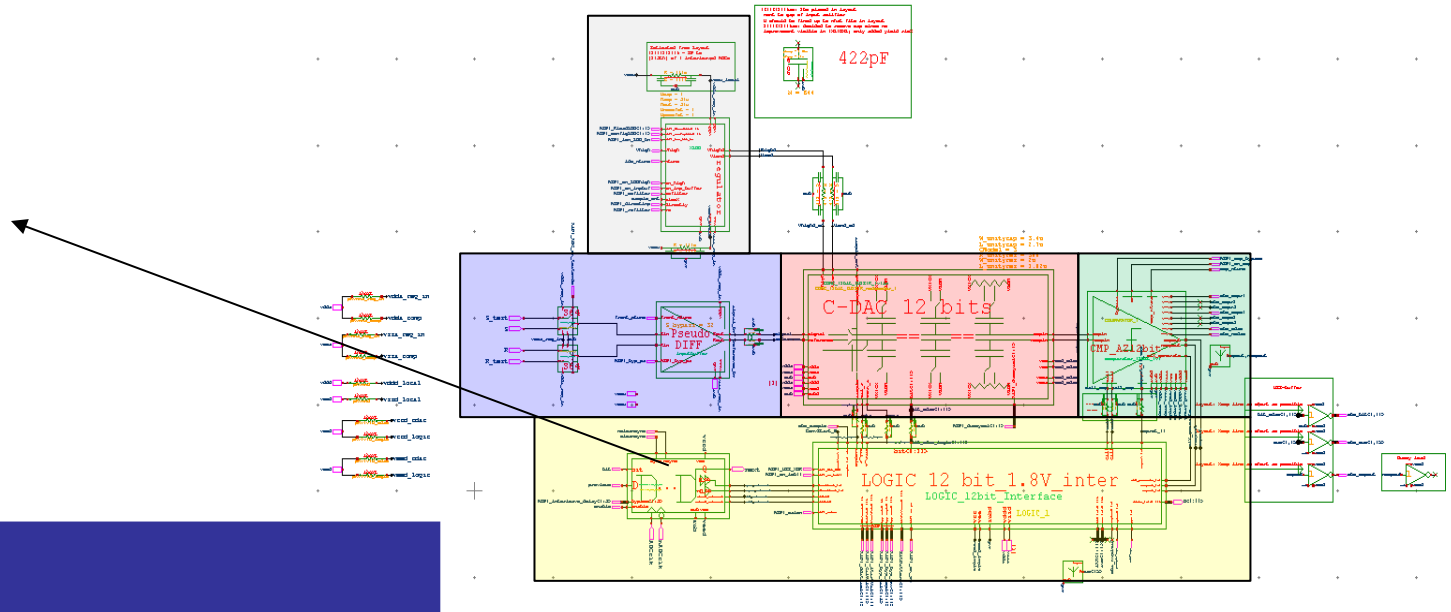
Symbol	Value	Unit
(n)XCLOCK	XCLOCK	MHz
(n)XSYNC	At the start of every row	XCLOCK periods
(n)ADCCLOCK	XCLOCK * 3	MHz
(n)ADCSYNC	XCLOCK * 0.125	MHz
(n)LVDSCLOCK	XCLOCK * 6	MHz
(n)LVDSSYNC	XCLOCK	MHz

SAR ADC Interleaving settings

caeleste

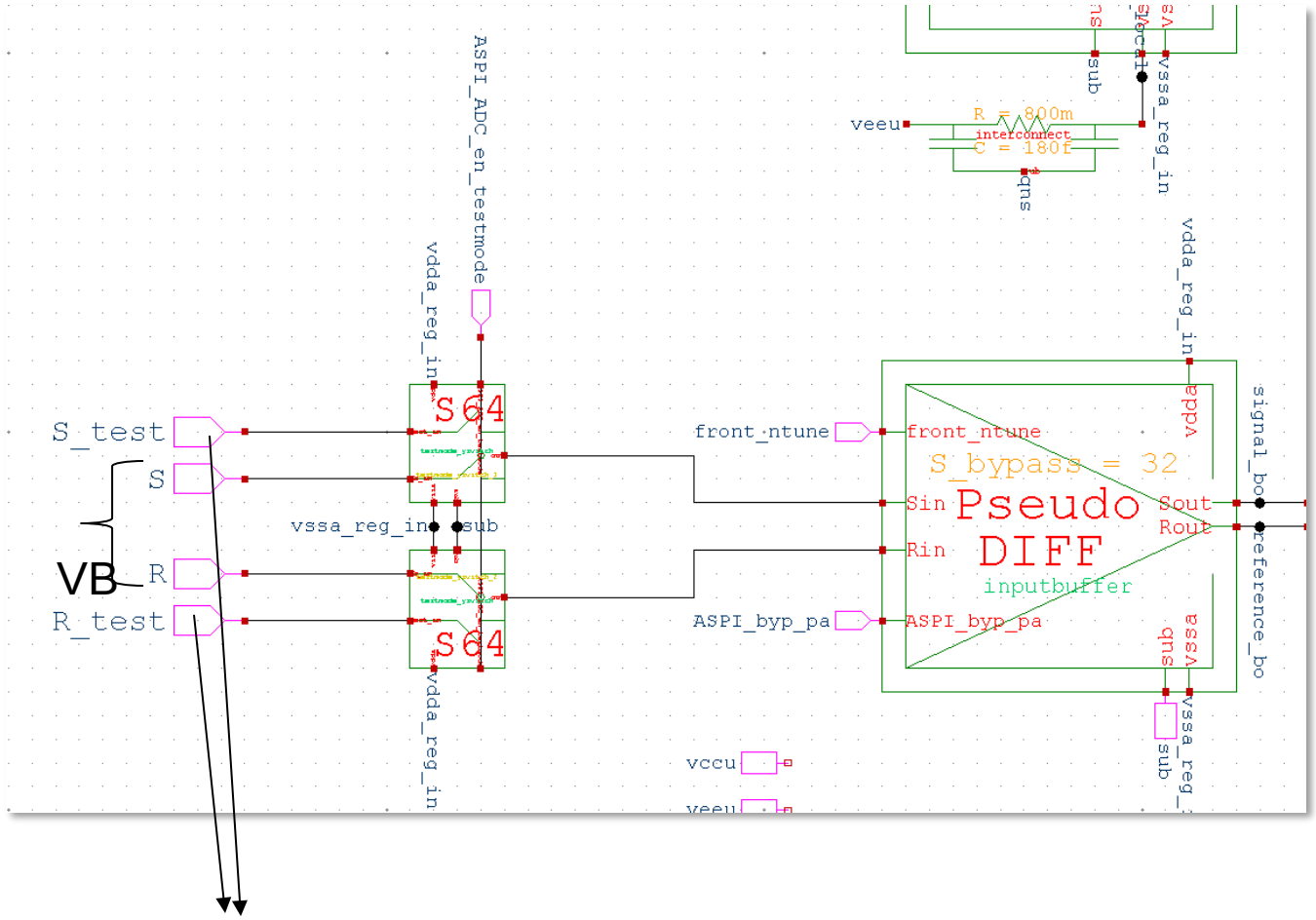
Programs interleaving of the 8 ADCs

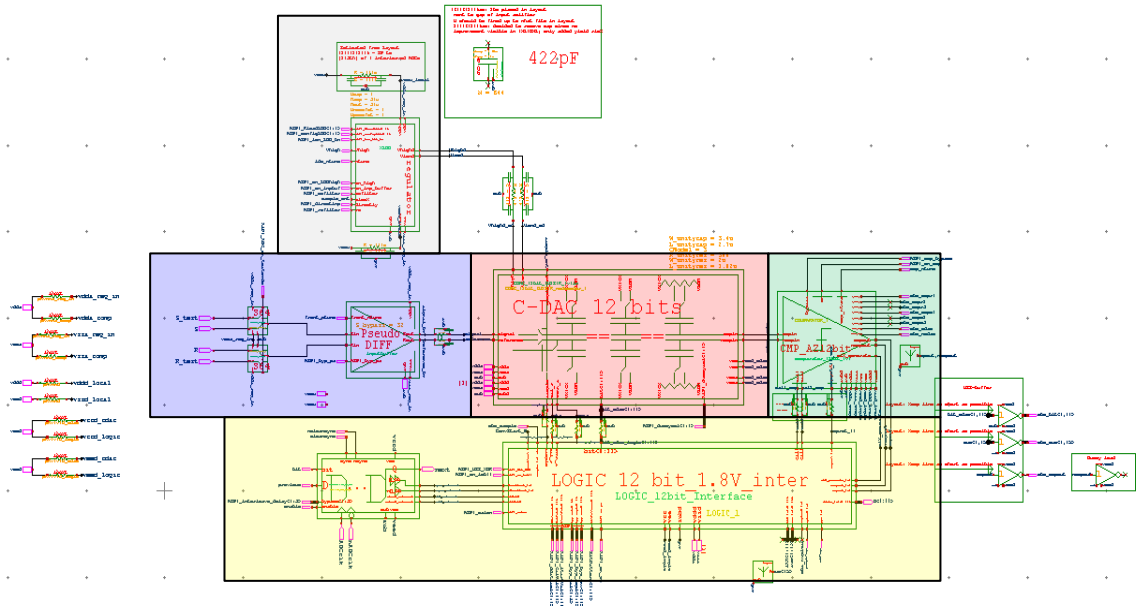
- Default: 8 clocks



Interleave_timing<0:3>	SPI Register	Data Bit	Function
0b1111	95	12:15	16 clock cycles per conversion (8-bit mode)
0b1011	**95**	**12:15**	**24 clock cycles per conversion (12-bit moden nominal)**
0b0011	95	12:15	32 clock cycles per conversion (14-bit mode = 4x oversampling)
0b0000	95	12:15	48 clock cycles per conversion (16-bit mode = 16x oversampling)

SAR ADC input





In some cases 8 separate pairs.
In other cases all 8 shorted together [S,R]
In some cases available as test points towards the “BP_ADC_test_reset and BP_ADC_test_signal”

CDAC concepts

CDAC hierarchy

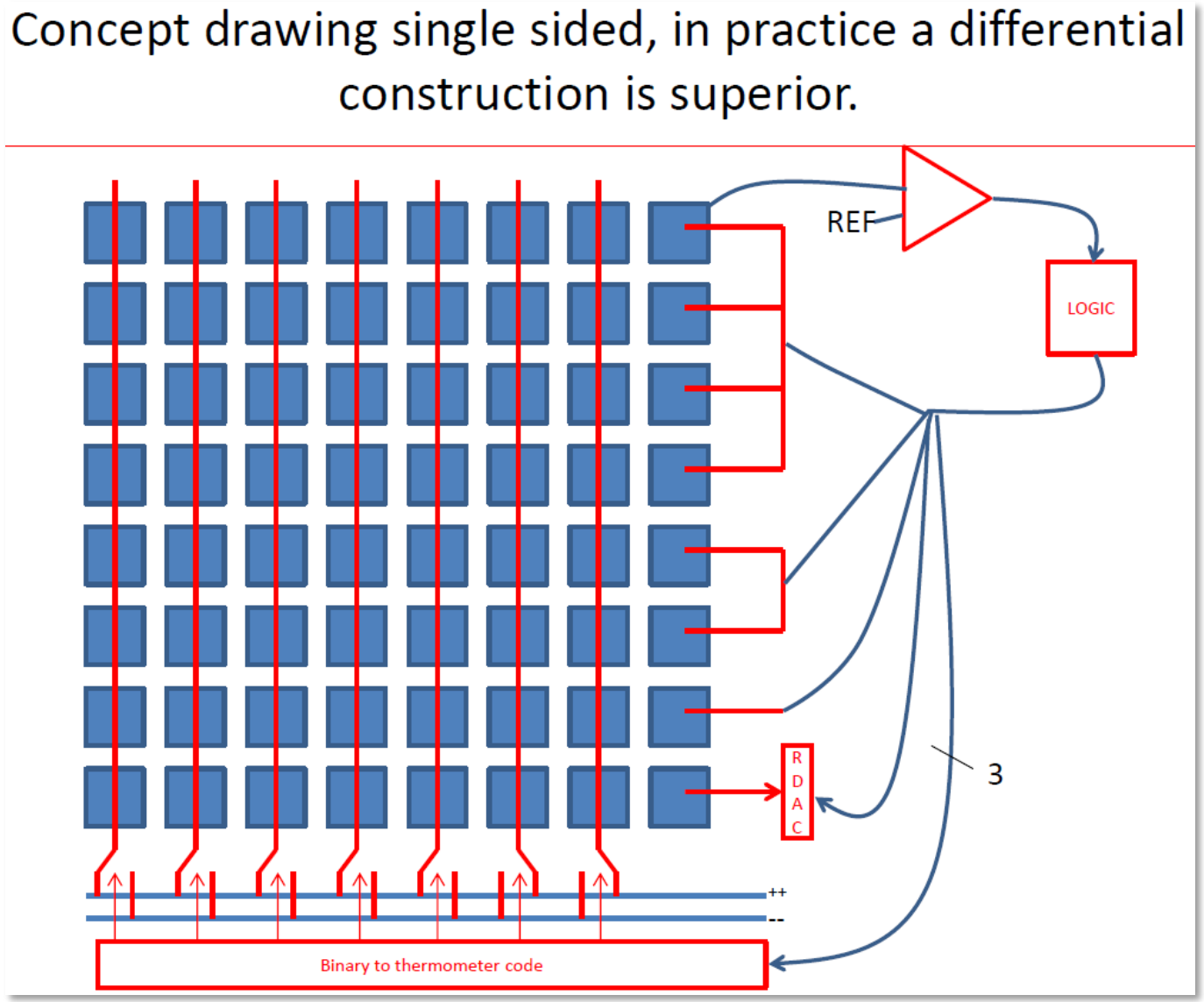
caeleste

Original concept from Notebook 0022→

The most significant bits are a “unary” = “thermometer code driven” capacitor bank

The medium bits are a classic “binary” capacitor bank.

The least significant bits are a single unit capacitor driven by a RDAC.



The “unit capacitor” approach

caeleste

The CDAC is built on unit capacitors

- They must be all matched, however the “unary architecture” relaxes this requirement.
    - For layout matching there is a single ring of dummy unit capacitors around
- The number of unit capacitors.
    - In a U6B2R4 (Unary 6 bits, Binary 2 bits, RDAC 4bits) the number of unit capacitors is 64*4, not counting the dummy capacitors
    - The total input capacitance are for each input, seeing this number all in parallel.
    - One desires a total ADC input capacitance
        - Not too high, for power, crosstalk, area
        - Not too low, so as to keep kTC noise low.

Example: In ATON they were 2x2µm², “POD” type, having a value of 20fF per unit cell. The ADC pure input capacitance not including parasitics and dummy capacitors is then (64*4*20fF=) 5.1 pF per input.

(not) equal speed in unary/binary/RDAC

Ideally they should all have the same settling speed.

Pro: fastest in “normal” (raw) operation

Con: short glitch when crossing a boundary double conversion.

Problems

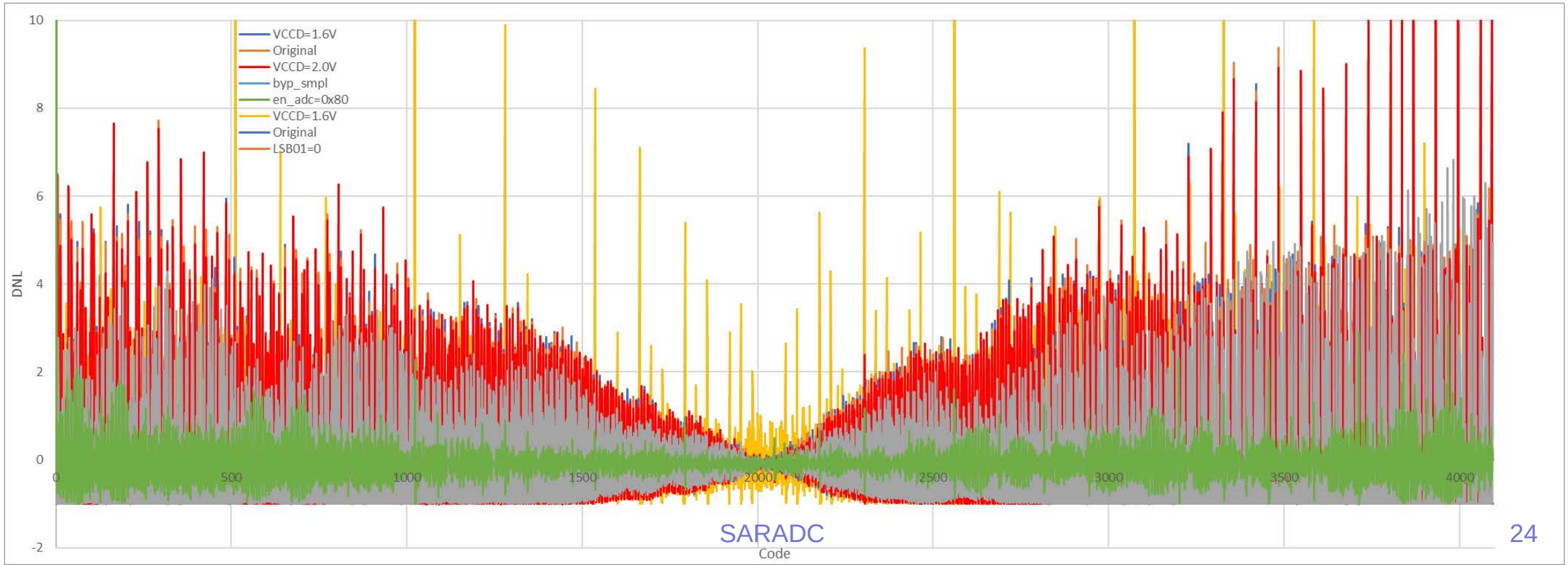
- Switches see unequal loads in binary.
- Extra gate delays in unary versus binary versus RDAC
- Realistic (fast, compact) thermometer decoders have unequal speeds for different transitions.

Noise and DNL minimum phenomenon

Noise (rubbish of all kinds) that is present on the difference between HIGH/LOW reference voltages is minimal at midscale.

The reason is that at midscale the HIGH/LOW difference impacts each of the differential comparator inputs in an equal way.

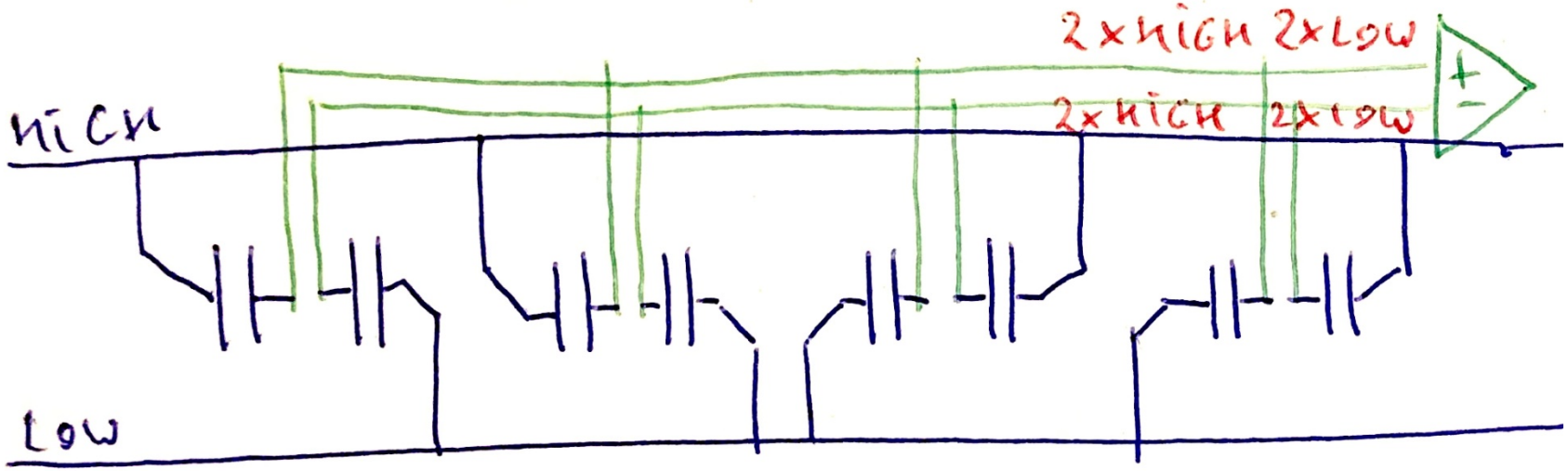
Measured on ATON 202207



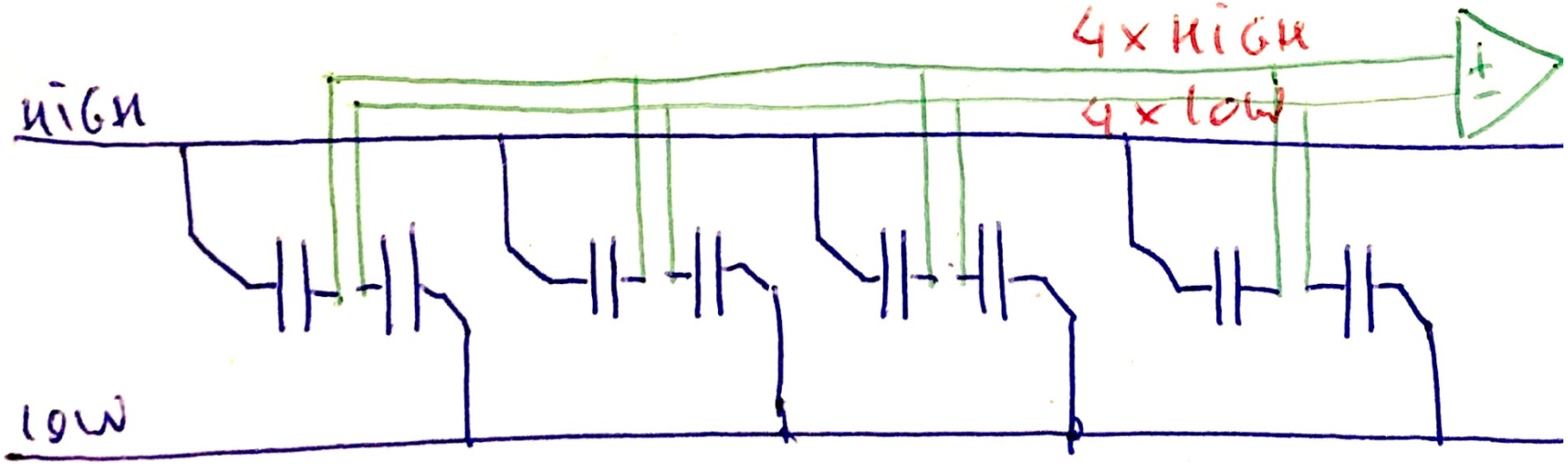
Origin of the noise minimum

caeleste

At midscale, an equal number of unit capacitors hangs on HIGH and LOW, as seen by each comparator input.



Towards scale extremities, each comparator input node sees capacitors connected all to either HIGH or LOW.



Noise and DNL minimum *shift*

caeleste

Noise (rubbish of all kinds) that is present on the difference between HIGH/LOW reference voltages is minimal at midscale.

The reason is that at midscale the HIGH/LOW difference impacts each of the differential comparator inputs in an equal way.

One can make use of that: add extra capacitance asymmetrically to the comparator input nodes. In Balor and Aton, one used the dummy capacitors for that.

- this method has a drawback: the input capacitance increases.
- Other drawback: the CDAC adjustment range drops below HIGH-LOW.
- *A further variant method ("split capacitor") does not have these drawbacks. See further.*

- In Aton one observed this shift of the minimum, but the minimum did not improve the noise level of the baseline. Possibly the improvement was masked by another larger effect.

Comparator

(Not) clocking the comparator (except its latch)

The ATON comparator was made programmable to be used both as

- Preamplifier and gain stages clocked = reset prior to conversion
- Preamplifier and gain stages not clocked = continuous time comparators

The end latch is always clocked.

Pro clocking:

1. should settle faster when large polarity change.

Pro not clocking:

1. eats up analog settling time (hypothesis not proven)
2. Clocking undos the calibration (offset cancellation) as the equilibrium of reset is not the same as equilibrium of calibration.

In ATON was experimentally verified that when not clocking, there were significantly more glitches.

(from Notebook NCR07 Aton debugging)

Conclusion: **keep the programmability** to chose between clocking and not clocking. As tradeoff not fully understood.

V_{HIGH} and V_{LOW} regulation

CDAC's HIGH and LOW reference

caeleste

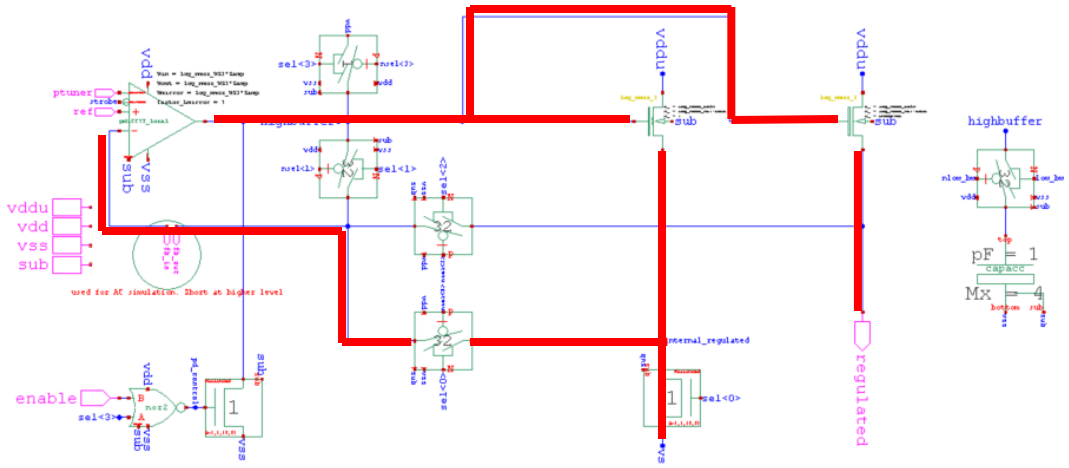
In BALOR, ATON, ELFIS2, LARADIS, on-chip regulation options exist per elementary ADC

The unregulated VDDU can be regulated down to a HIGH, the VSSU is then used as LOW.

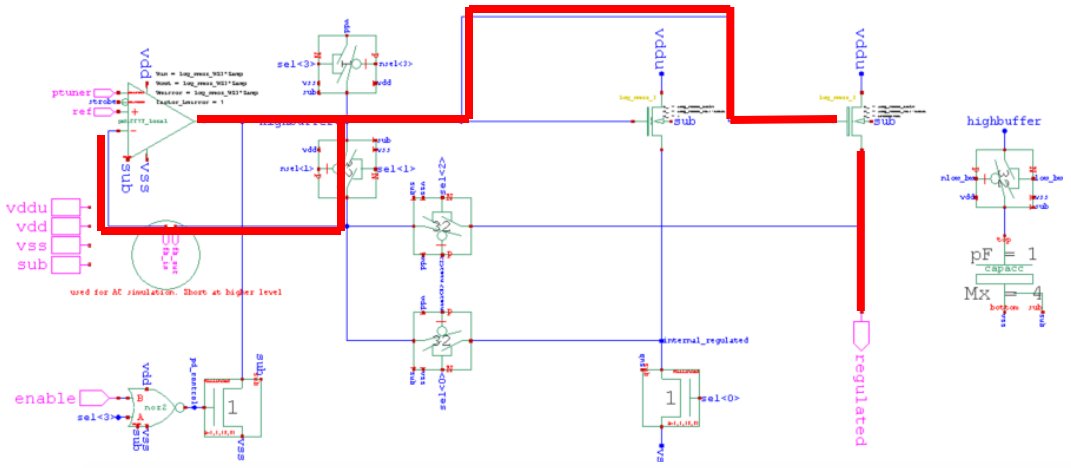
Following programmable modes exist:

1. Bypass, i.e. the VCCU is used as HIGH reference, via a switch in each unit ADC.
2. Feedback SF regulation: the local HIGH is made equal to a reference by feedback.
3. Feed Forward (FF) regulation, referencing a dummy SF regulator stage. It does not feedback, but it compensates for temperature etc.
4. The SF is directly driven from a DAC voltage. Same as FF, but there is no temperature and process compensation.

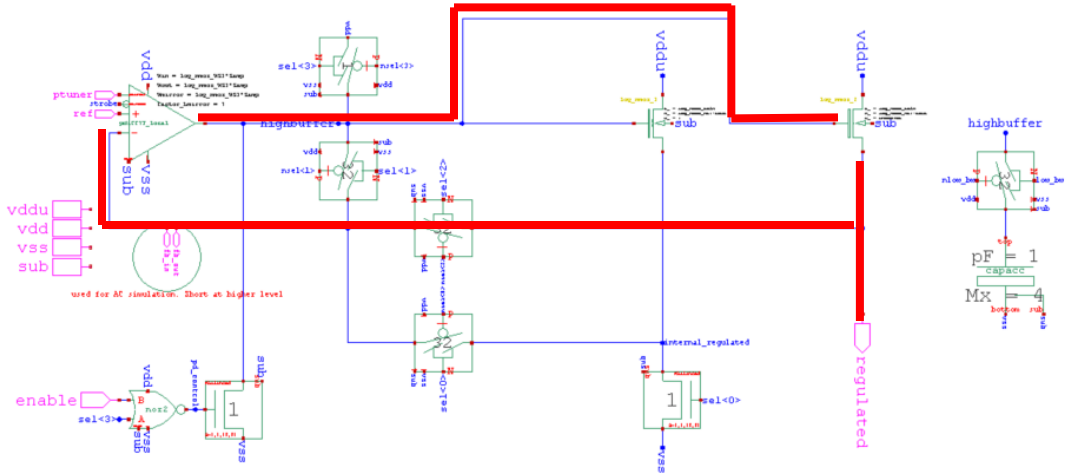
Regulator Modes of operation



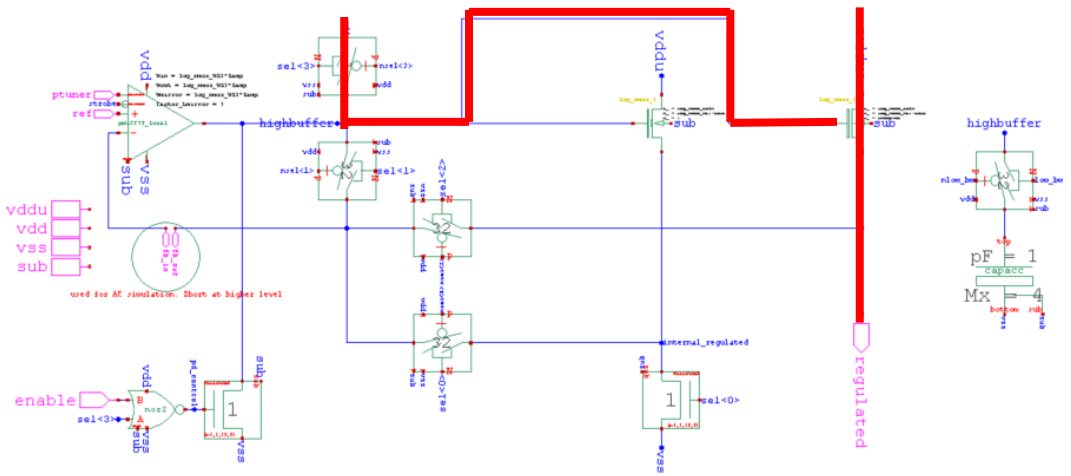
Feedforward mode (FF)



DAC + buffer drives the SF gate (UG Buffered)



Feedback over SF (Full Regulator)



Bypass (SF as switch fully on)

VDDU / VSSU network

caeleste

In order to limit the “IR drop” generated crosstalk, two viewpoints:

- VDDU/VSSU (or VCCU/VEEU) wires are separated per [interleaving] phase (i.e. elementary ADC) and joined at the bondpad (as in Balor, Laradis…)
- Have a single low impedance track (as in Aton). A simple design, and an unquantified risk on phase to phase crosstalk.

Furtheron we will discuss possible further improvements to limit phase to phase crosstalk.

Double conversion and postcorrection

What is “double conversion”

caeleste

Double conversion is an “unsupervised error correction mechanism”.

Suppose that the ADC makes a (dynamic or static) error at some bit. Subsequent lower bits can never fully correct the error, by nature of the *basic* SAR principle.

Our solution is to allow (a number of, statically programmed) bits to act twice, each time having a conversion that can eventually result in twice that bit being 1 or 0 in the converted digital value.

This will be realized as an adder that receives the bit value twice, either in upward counting or downward counting mode.

The method allows to cross the major bit boundary by lower bits.

The method is intended to reduce large conversion errors.

At run time one can program the ADC to have such double conversion on any bit position. Of course, if you do this for every bit, the conversion time will almost double. In reality, one will choose the bit position where double conversion is foreseen carefully.

What is postcorrection

caeleste

Postcorrection is a double conversion on the LSB. It can happen multiple times.

Postcorrection is also a non-supervised error correction method.

After the regular (e.g. 12bit) ADC conversion cycle, one might not effectively have reached the ideal ADC output value. If this happens at random output codes, we speak about noise. If this is systematic for certain output codes, we call it “large DNL” or “code gap”.

Such code gap is solved after the regular conversion by executing a series of comparisons, and acting by adding or subtracting one LSB to the end result. This solves code gaps and should improve DNL and noise, as far as it is not due to mismatch inside the CDAC.

The odd/even effect of postcorrection

caeleste

As postcorrection adds or subtracts a +1 or -1 to the regular ADC result, it will have **only odd** or **only even** results, depending the number of postcorrections, only depending if the regular result was odd or even.

Also the regular ADC converted result (without postcorrection) suffers from this odd-even effect. This is a major FLAW of our ADC.

This increases the random noise of the end result.

It also degrades the result if one would use the series of postcorrected results for oversampling.

A number of tricks has been proposed to remove the odd even effect AND reduce the associated discretization noise.

The odd-even effect, countermeasures

Our SAR ADC in simple operation (raw mode without double conversion and without postcorrection) has an odd-even effect

the solutions implemented and preferred (since Balor) are

- Alternating between up/same and same/down
  - For the last conversion, and for each subsequent postcorrection
- A better root cause solution was implemented since Generation 8 (2021 ELFIS2) (is that true?)

Q: what is the baseline? (Cfr Nitara / Amir?) please document here.

caeleste

Oversampling

Oversampling concepts

caeleste

Actual concept, used in BALOR a.o.

A separate 16 bit adder accumulates 4 or 16 values coming out the ADC while postcorrecting.

Assuming that postcorrection results are uncorrelated results of the AD conversion (which they are not!), the average has less noise.

Alternate concepts, not elaborated:

- Register and readout the last N comparator bits, so that off-chip the intermediate values can be reconstructed.
- Other methods see under future improvements

int2float

A subcircuit that converts in a combinatoric fashion a 14bit integer to a 12bit float

INT2FLOAT

caeleste

The ADCs of LARADIS and ELFIS2 both have a INT2FLOAT encoder acting on the 14 MSB of the 16 bit oversampler.

See Notebooks 0226 and 0328, also 0571

Project SENSATION

In project SENSATION the Laradis ADC was tested in 4x oversampling (14bit mode) and 16x oversampling (16bit modes).

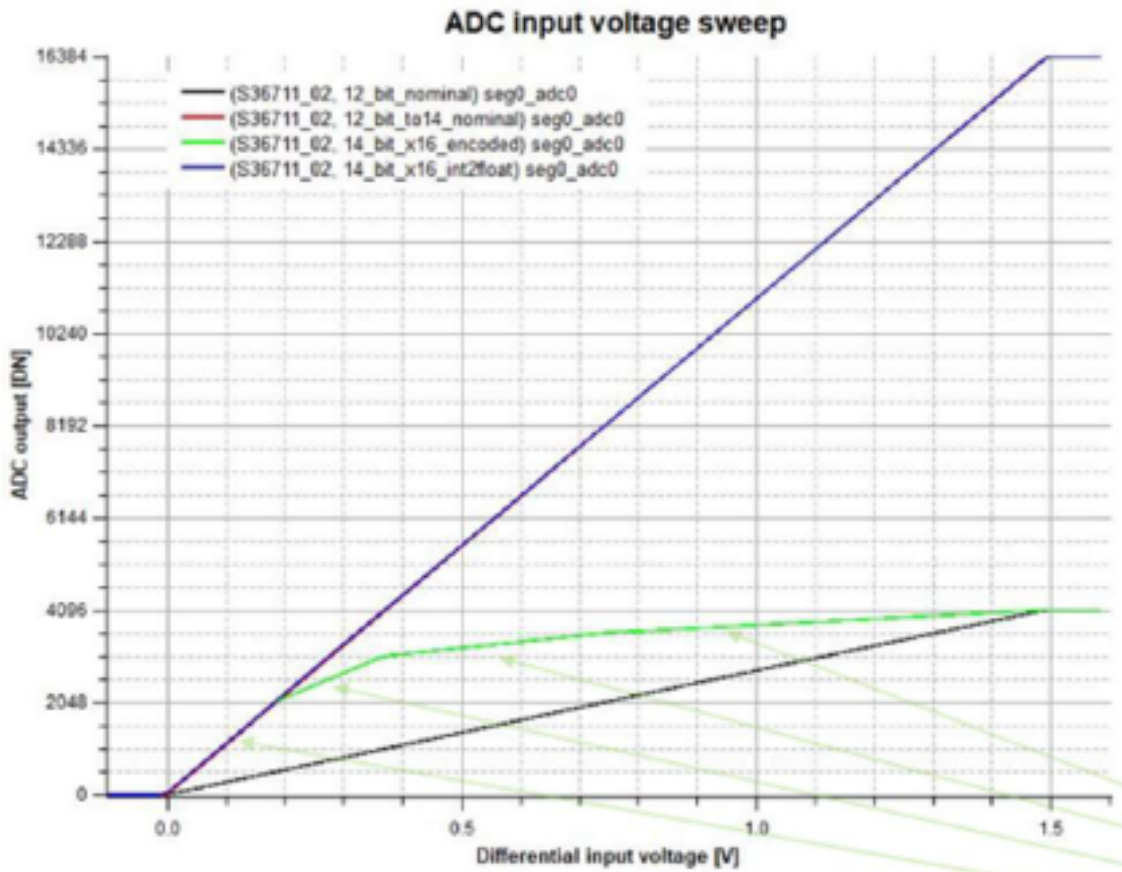
In both cases the 14 highest bits are encoded to 12 bit words and expanded at the computer side to the original 14 and 16 bits. Results were reported in the *Sensation WP3, D3.6, Final evaluation report, Version 1.0, 17-06-2021*

14bit mode

From report

4.4 14 bit mode, 30 MHz

In 14 bit mode, the ADC inherently compresses the data into a 12 bit word. This is allowed and considered lossless as the ADC is designed for image sensor application where photon shot noise inherently leads to a minimal noise level at each level of illumination.



14bit operation, encoded in 12 bits, decoded back into 14 bits

14bit operation, encoded in 12 bits

12bit operation

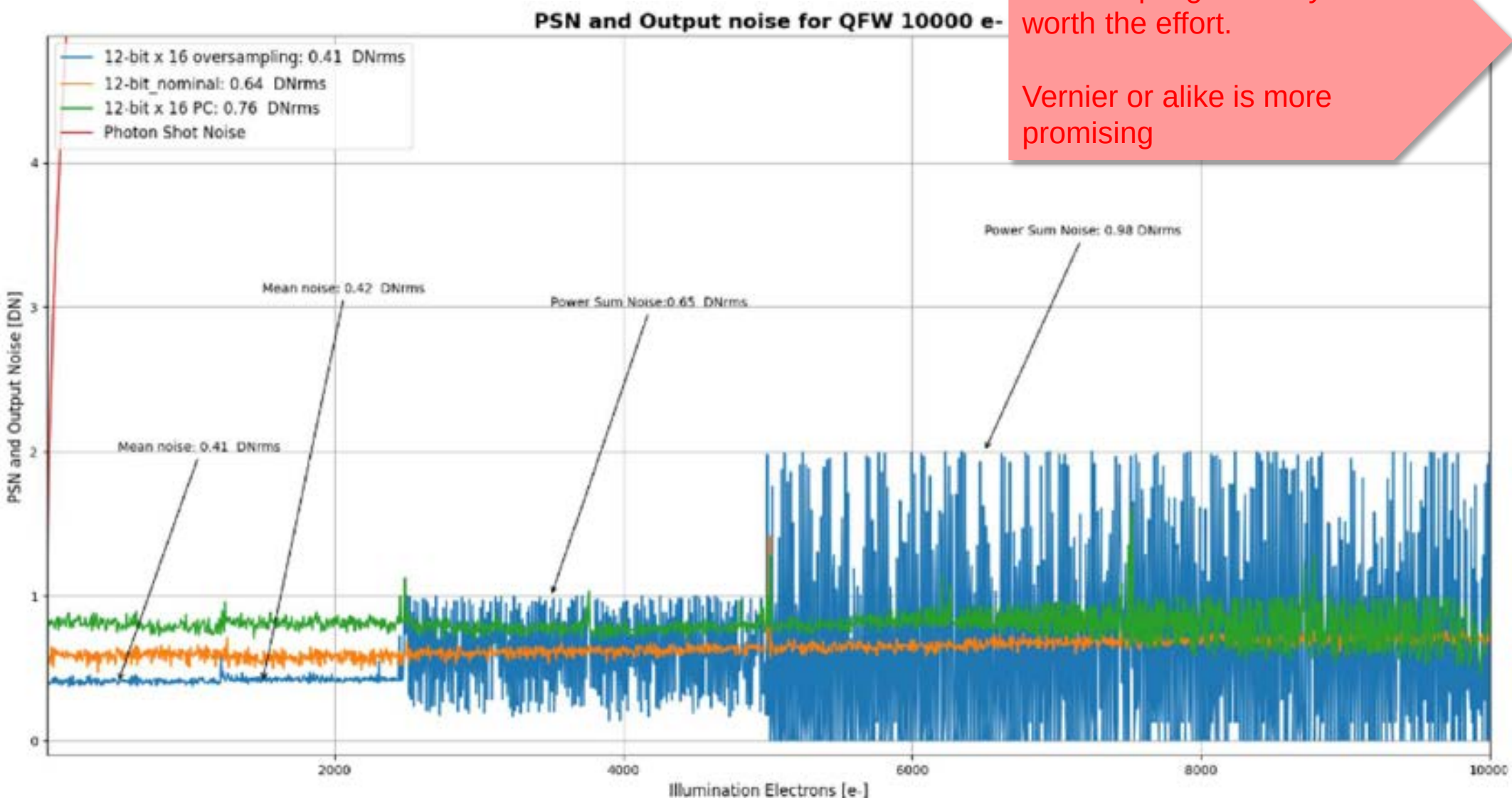
ABCDEFGHIJKLMN LOGIC	RESULTING BIT WORD	#bits error
If A == 1	**111**BCDEFGHIJ	4
If B == 1	**110**CDEFGHIJK	3
If C== **1**	**10**DEFGHIJKLM	1
if C==**0**	**0**DEFGHIJKLMN	0

Figure 4-8 ADC gain curve (green=encoded, blue=decoded, black= reference 12 bit operation)

16x oversampling, read out via int2float

From report. Oversampling results in baseleve noise going from 0.64DN (nominal) to 0.41DN.

This is a very modest noise reduction. The discretization limit would be 0.23DN, the effect of perfect uncorrelated oversampling would result in 0.16DN



Conclusion: (this way of) oversampling is barely worth the effort.

Vernier or alike is more promising

Array issues

Array issues

caeleste

In practice, the issues encountered with image sensor on-chip ADCs are related to the fact that there are many more than one in an intensily connected network.

→ A single ADC may work fine, in an array of such, inter-ADC crosstalk results in significant noise and artifact increase

→ This problem applies to all types, such as Ramp ADCs, SAR ADCs…

→ Most root causes are related to the common nodes (supplies, substrate, biasing)

→ The other causes may be due to “plain” crosstalk between e.g. digital parts and analog parts.

Arrays issues (general concepts)

caeleste

<<< insert here relevant parts of notebook 0124>>>

Relevant Parameters

Agressor circuit:

- Non-synchronous peak current of one agressor circuit
- Note: synchronous peak current due to clock is likely not an issue

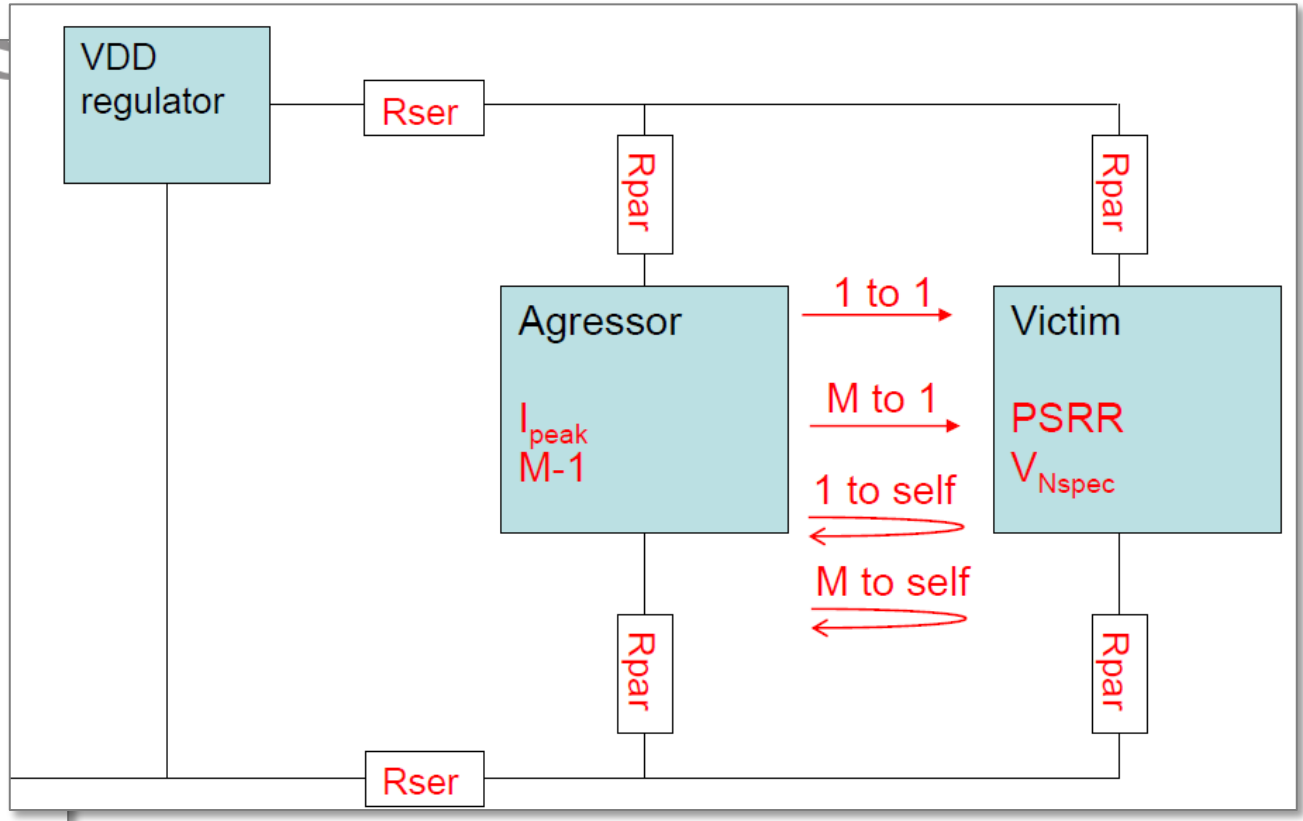
Supply routing

- Supply wire resistance for IR drop; both VDD&VSS
- Configuration of the supply routing: series/parallel
- "M" Number of elements in the array

Victim circuit

- PSRR of the victim circuit
- Noise spec of the victim circuit as "V_{Nspec}"
- Crosstalk from itself or from other array elements
- *the "equal elements" are both victim and agressor, at the same time or shifted in time*

30 November 2015 Caeleste Confidential 3



Not all crosstalk is adverse

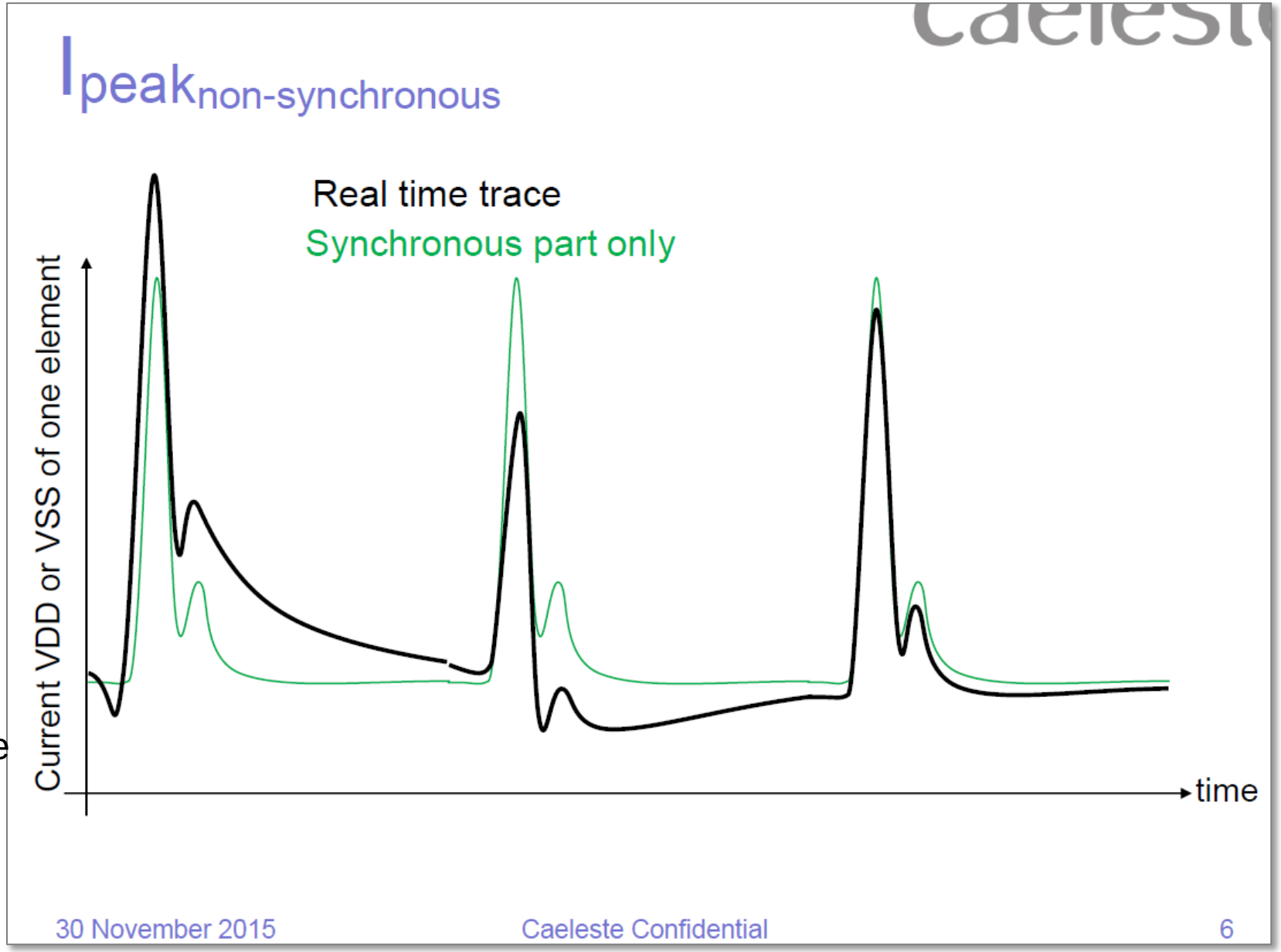
There is a fundamental difference between

harmless
1. Synchronous, reproducible crosstalk

hurts
2. Signal dependent crosstalk

noise
3. Random crosstalk

Understand this difference in prioritizing counter measures.



Radiation tolerance

Radiation hard

caeleste

The same approach as in image sensors is adhered.

Approach

- TID hard layout
- DICE in the static ASPI registers
- Not DICE nor TR for the fast in-ADC registers
  - Thus an event in one of these might result in a one-code error, similar to a large conversion error.
  - If really wanted, one can equip also the internal fast registers and latches with DICE, making these slower.

Possible future improvements

In the binary SARADCs

Advise to do in next generation

Should not do

Update RDAC

caeleste

To be done for Flames

Bottom switch NMOS only, this simplifies the design of the decoder.

Advise: make all switches same nature as for the unary&binary CDAC.
UNLESS split capacitor method used which demands FAST switches

BUT
Will definitely not do if split capacitor is done.

Timeconstant of RDAC

caeleste

The RDAC in series with a unit capacitor should be given a timeconstant as short as the other switches.
If not, one will see a vulnerability to VHIGH-VLOW noise.

Advise: reduce R of RDAC
proper
Parallel resistor (MOSFETs)
does not make sense.

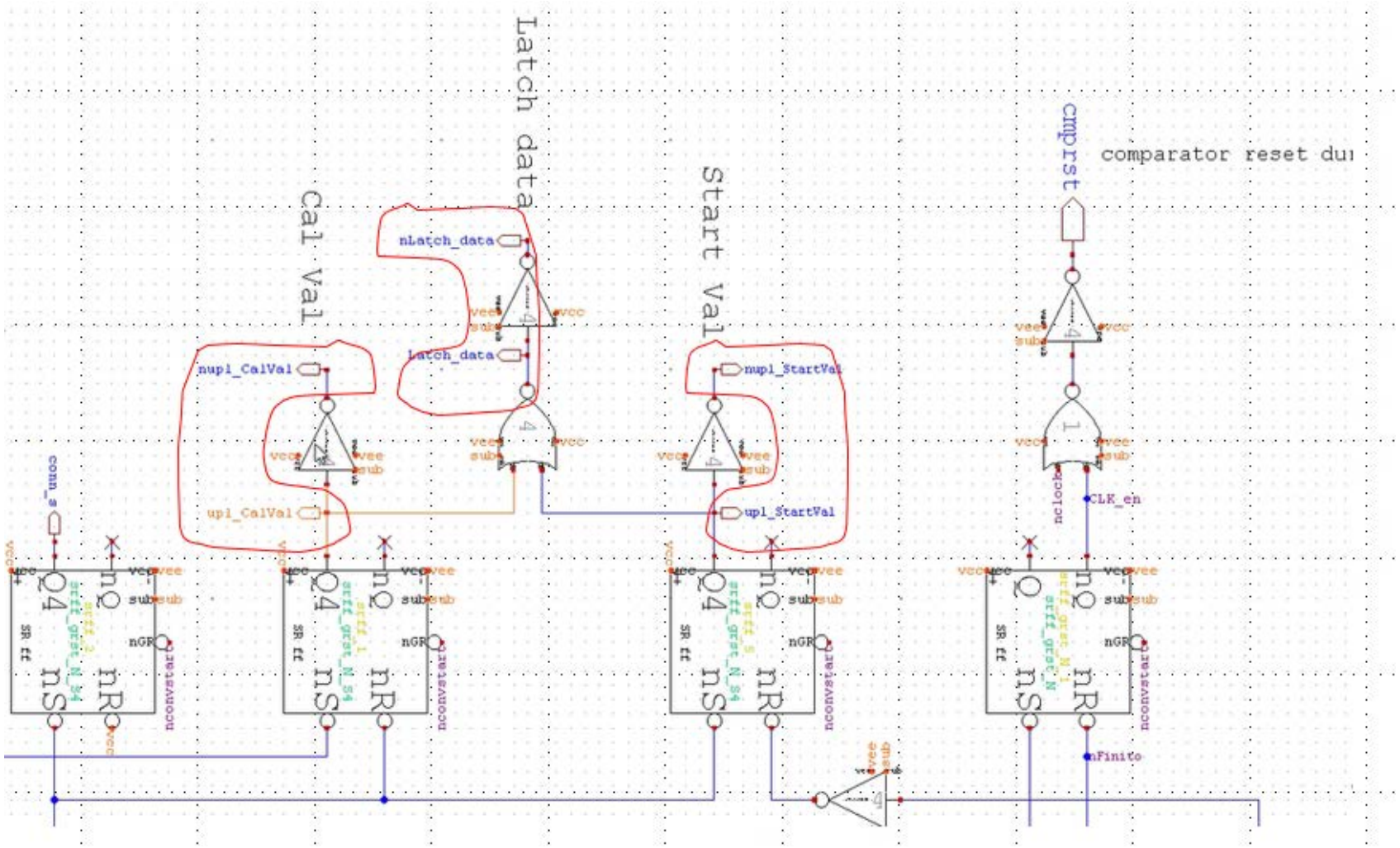
In sequencer: symmetric controls

caeleste

Make these internal controls symmetric by using invninv_core instead of just inverting.

(from Notebook NCR07 Aton debugging)

To be studied if really necessary.
If so, as circuit cost is marginal, consider in next generation



Dummy caps used as decoupling

caeleste

Should not load the output of the regulator, or at least be disconnectable.

Should not load the input nodes of the comparator, or at least be disconnectable.

Advise: use *single* band of dummy connected to ground. Do not use for decoupling.

also

caeleste

do

- Hardwire startval on 100000000000

do

- Overload calval with teachcodes

HIGH and LOW reference regulation

caeleste

- This is about interleaving ADCs (when not interleaving, some aspects change)
- On-chip per ADC regulation options
  - Bypass, i.e. the VCCU is used as HIGH reference, via a switch per ADC
  - Feedback regulation.
  - Feed Forward (FF) regulation, which compensated for temperature etc.
  - DAC to SF, same as FF, but SF is directly driven from DAC voltages.

  When doing so, VCCU and VEEU
  - Can be separated per [interleaving] phase and joined at the bondpad (as in Balor, Laradis…)
  - Have a single low impedance track (as in Aton)
- Collective regulation options
  - All HIGH and LOW are shunted, pretty low impedance, decoupled at bondpad
  - LOW is de facto equal to VEEU
  - There may be a regulation VCCU to HIGH
    - On-chip, at the HIGH bondpad
    - Distributed over all ADCs

Abandon in-ADC regulation

Conclusion
In general a preference to move completely to “split capacitor” as Notebook 0621

Notebook 0621: Compensation of glitches in HIGH reference

As in notebook 0621→

This makes sense if the regulation is on-chip and imperfect.

This drawing leaves open

- HIGH not shunted amongst ADCs
- HIGH shunted amongst ADCs

The last case resembles the previous slide.

What about the RDAC?

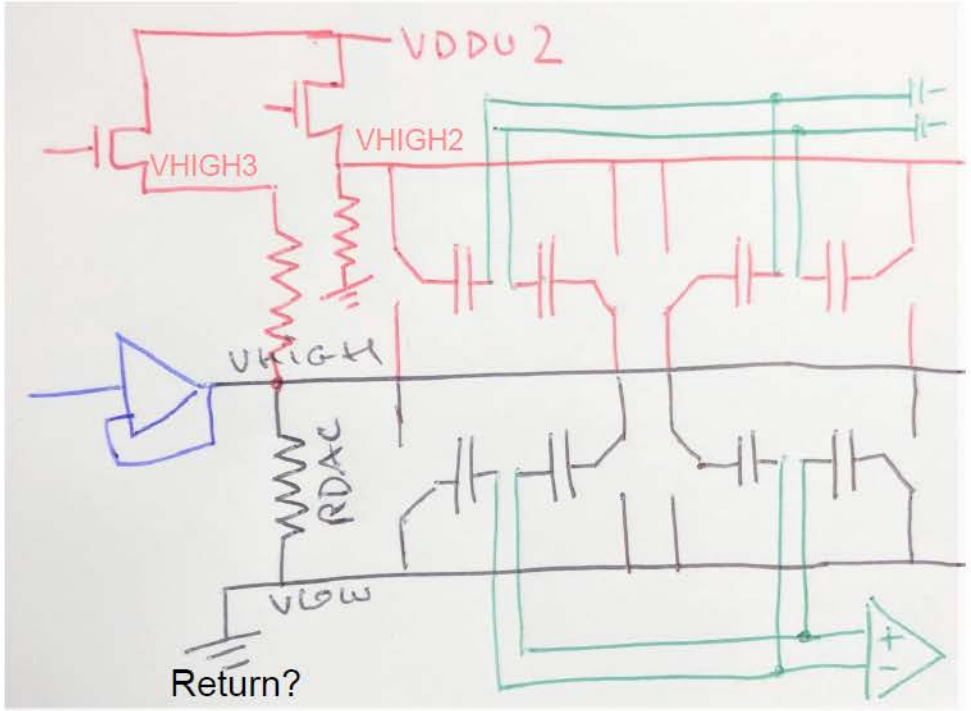
The RDAC of the dummy CDAC should not be tied to the disturbed VHIGH2 but to a clean VHIGH3.

In that case, the midpoint VHIGH can be made clean by construction.

Driving from a simple DAC is sufficient (to be proven).

"Return2" is the current return path of VDDU2 in this scheme.

The dummy comparator should only be a representative input capacitance.



21 September 2022 confidential 15

Reduce reference voltage sensitivity by *split capacitor*

Above: concept drawing of differential capacitance on the input wires of the comparator

Below: each capacitor is split in two, of which

- One half is tied to LOW (GND)
- One half is tied to a new “HIGH” reference, that has 2x larger voltage.

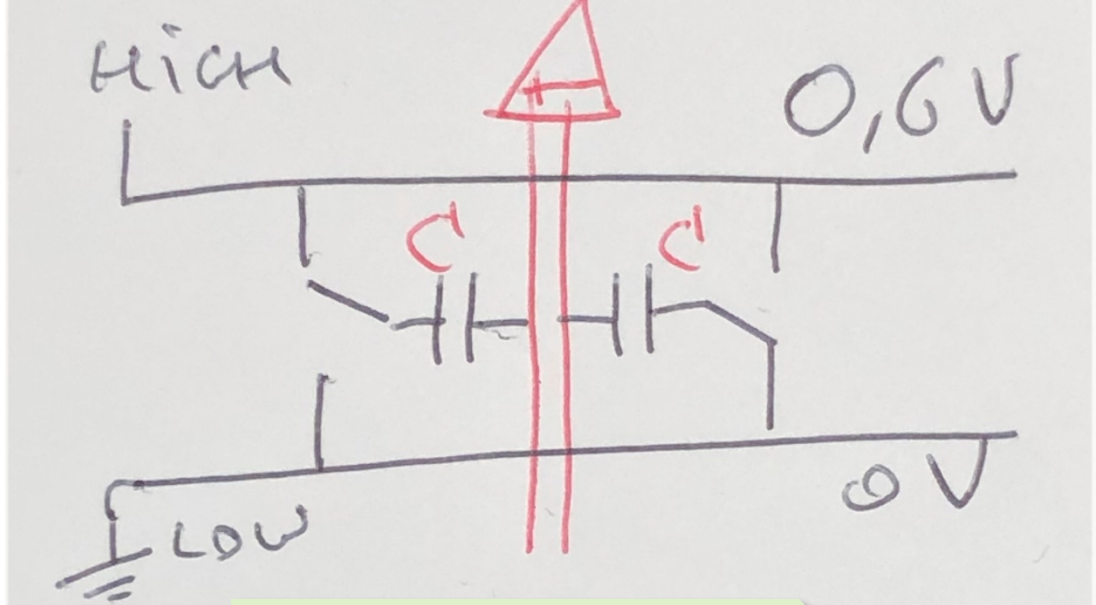
This construction makes the impact of noise/rubbish on the HIGH reference twice as low.

Can do that if HIGH-LOW is indeed low.

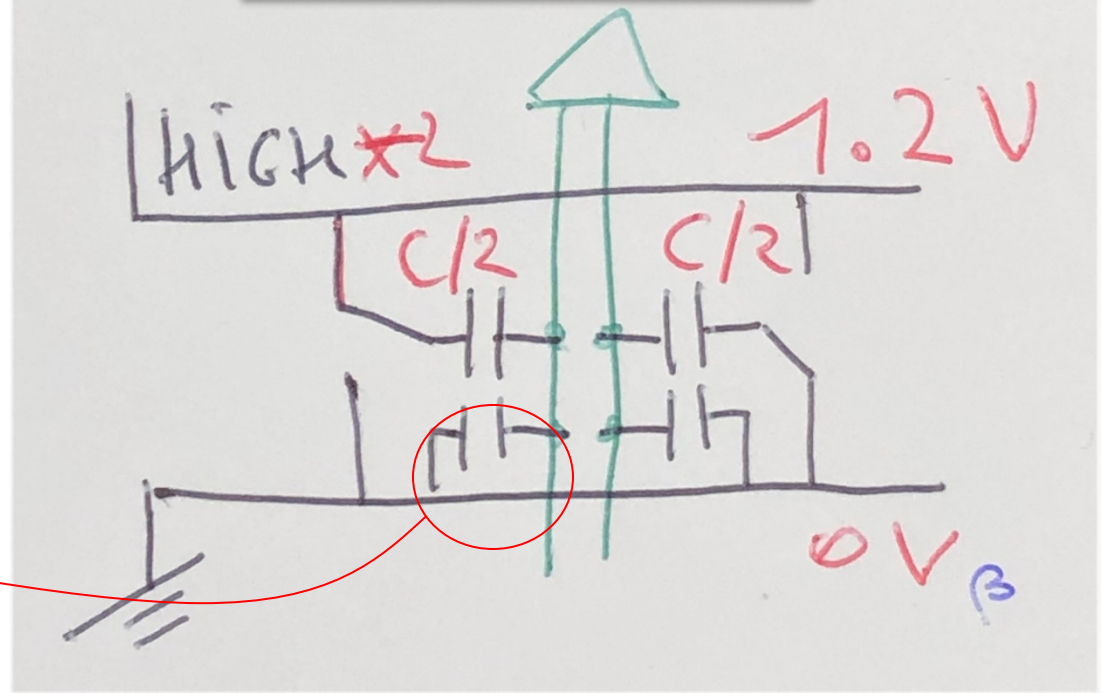
Why not: 3x0.6V=1.8V

To try out: can this be combined with the dummycap asymmetric connection for noise minimum without its disadvantage. Next page

Can be combined with collective regulation



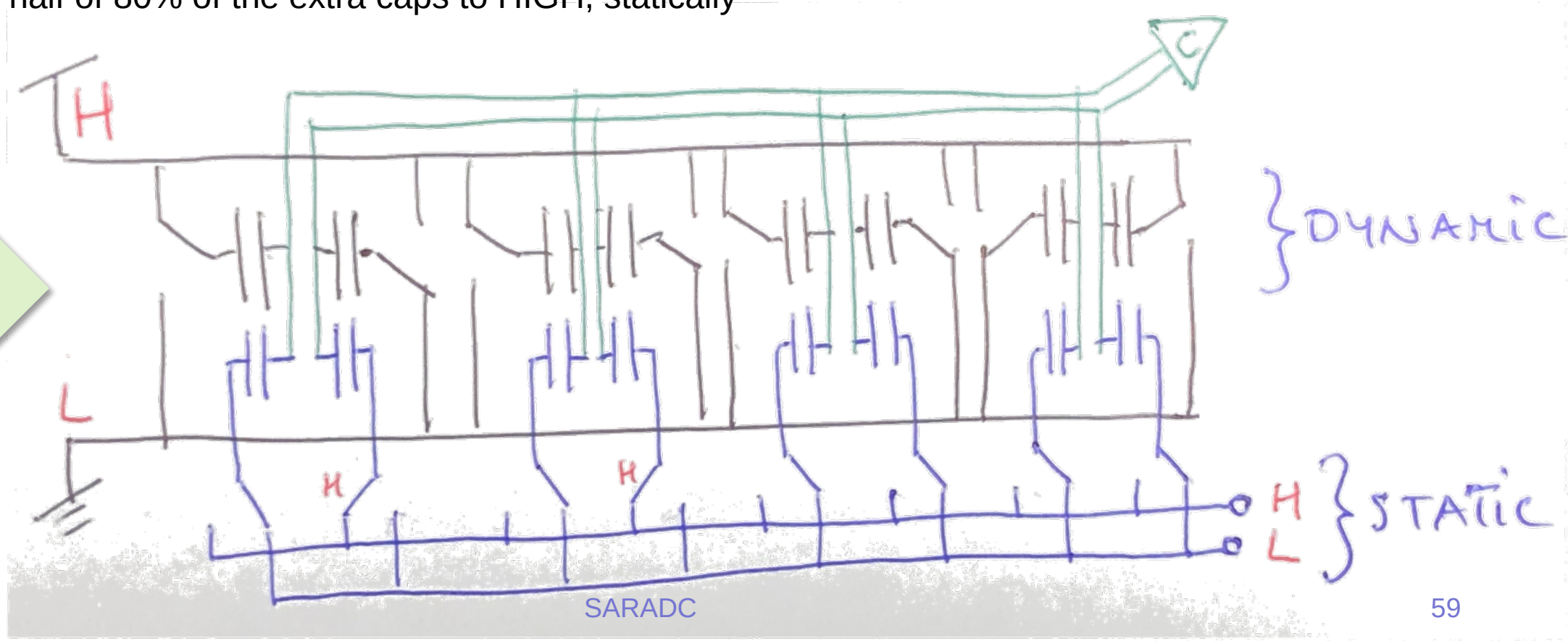
Propose to use in next generation (+next page)



Combination of split capacitor and noise minimum shift.

This is a variant on the *noise minimum shift*. In this embodiment, the disadvantage of increased capacitance and less adjustment range is not present. None or one of the split capacitor pairs is connected to HIGH or LOW reference in a static way. If you want the noise minimum to happen at 10% of range, one needs to tie half of 80% of the extra caps to HIGH, statically

Propose to use in next generation



Further evolution of the previous see Notebook 0635

- Apply 1.8V VHIGH, directly on bondpad
  - Perhaps one can put a regulator on the bondpad. Perhaps just rely on outside regulated quality
- Downconvert to 0.6V range by adding the {1/3 normal} + {2/3 parallel} C on the comparator input nodes.
- Allow noise minimum shift by having bottom plates to HIGH or LOW reference programmable.

And also

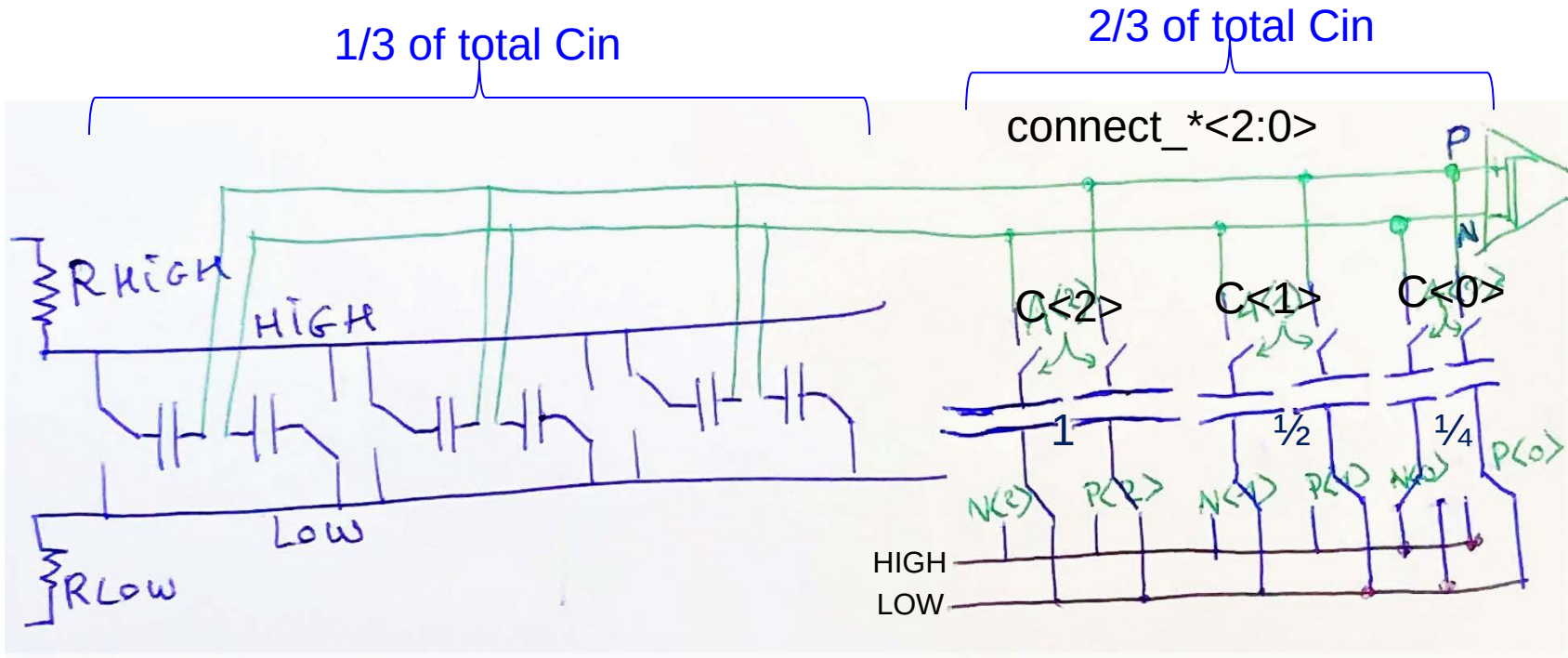
- Make the downconversion programmable by making the extra caps connectable in binary fashion

← Less downconversion (lower gain) goes with lower input capacitance (faster, noisier) and lower resilience against HIGH-LOW rubbish.

Caveat also

- This ressembles the dummycapacitor network as present in generations 8, 9. Do not make same mistakes.

Will use in non-binary



20221226ba last update

Single direction CDAC switching

As suggested in Notebook 0621

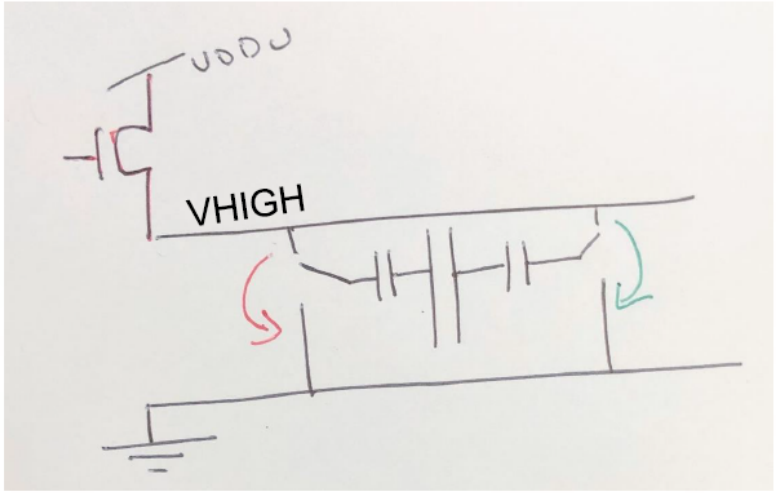
Countermeasure?

(avoid victim situation)

Avoiding the glitch by not having a low-high switching.

In the start position all capacitors are ties to VHIGH.
After comparison chose either 10 or 01

Weak point: the comparator common mode will drift.



Will do this in non-binary

PS: this reminds Notebook 0045, slide 25

caeleste

Toplevel register loops

Rethink the toplevel, and improve the imperfections
of the two critical loops digital / analog

Rethink the two register loops

caeleste

(from Notebook NCR07 Aton debugging)

1 dff = 2 latches

In fact we have 3 latches in series.

In classic "robust" registered logic, the blocks of combinatoric logix are separated by one DFF, thus 2 latches.

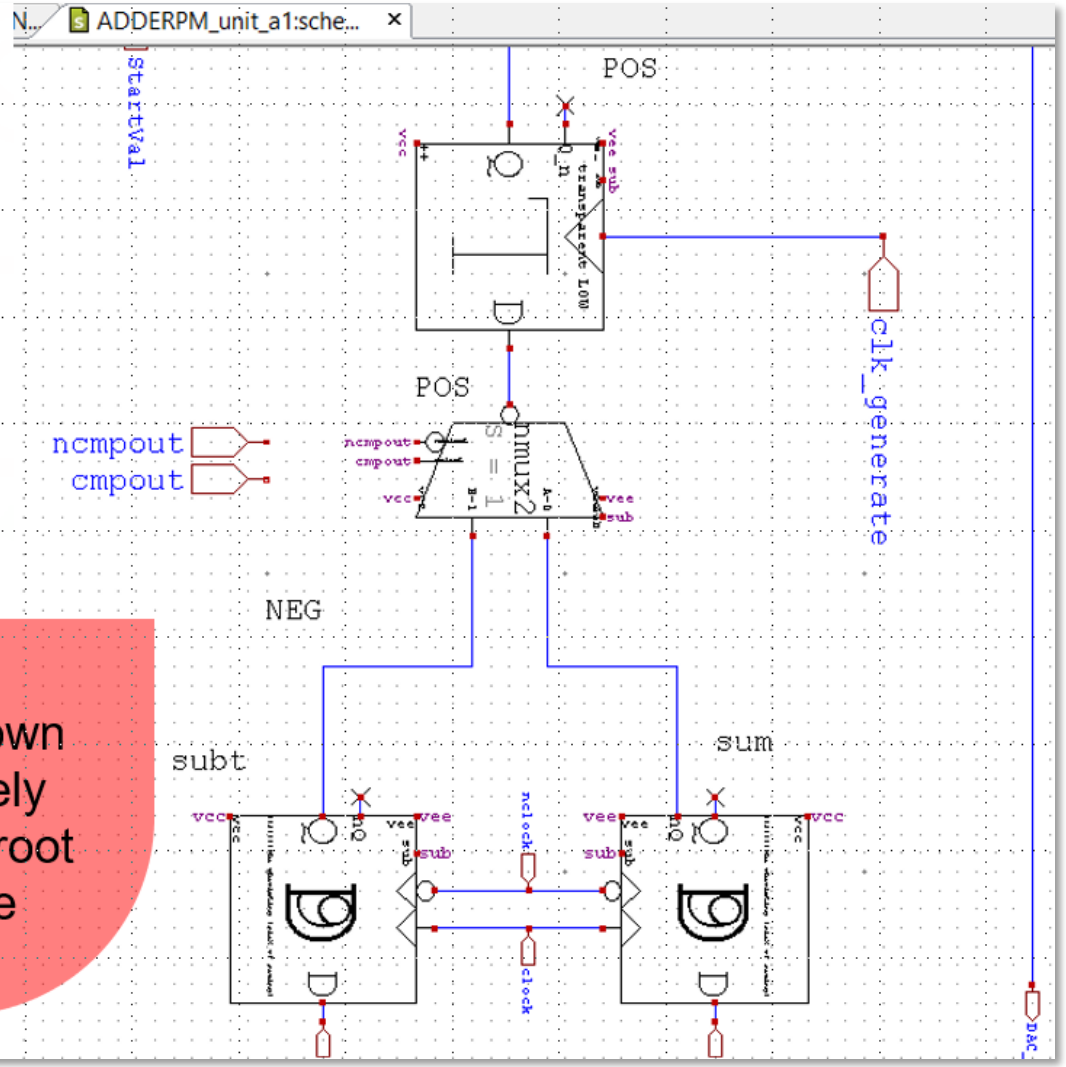
Hypothesis: the propagation time allotted to the loops

- The adders
- The CDAC+comparator

Is insufficient

Not known but likely not the root cause

22 September 2022



Avoid making/using clk_generate

from Notebook NCR07 Aton debugging→

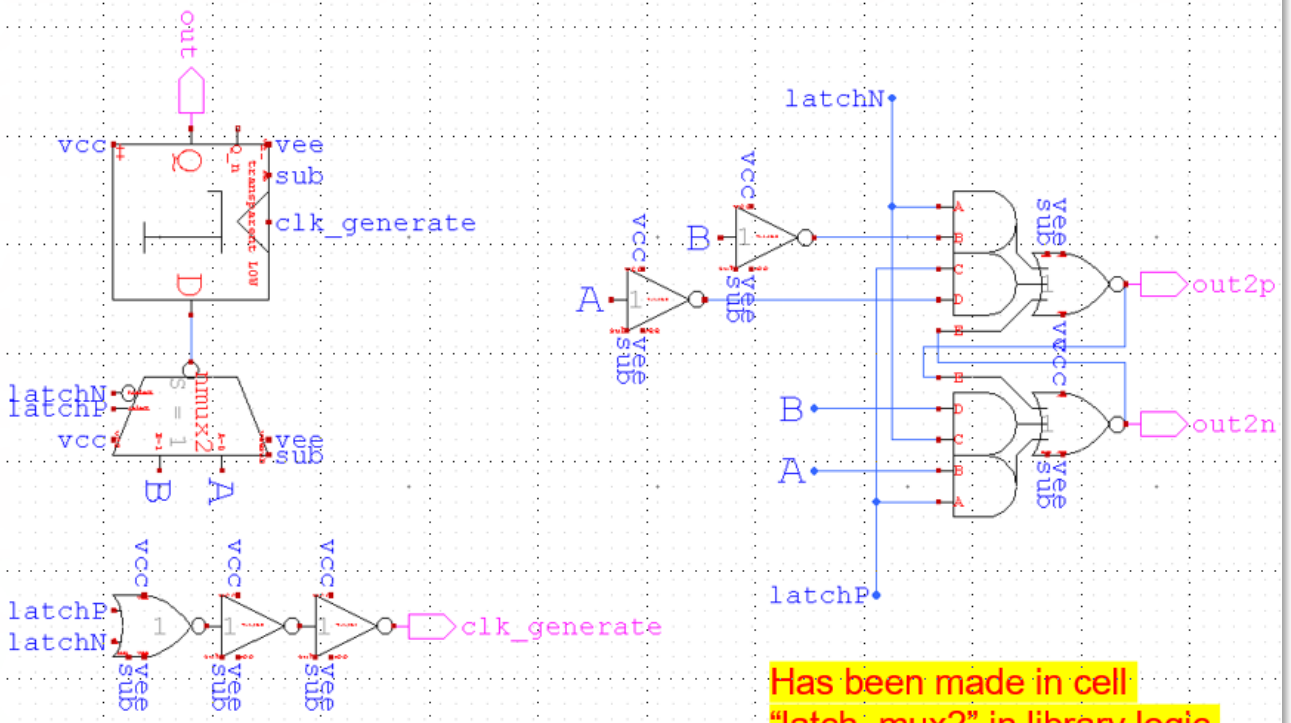
See also Notebook 0624

Concept imperfect as failure modes possible. I.c. if pulse is very short, not all bits may be updated.

Probably need to gate latchN/P with the comparator clock.

Functional equivalent: circuit

The right circuit is mux2 embedded in the cross-coupled SR latch.



Has been made in cell "latch_mux2" in library logic.

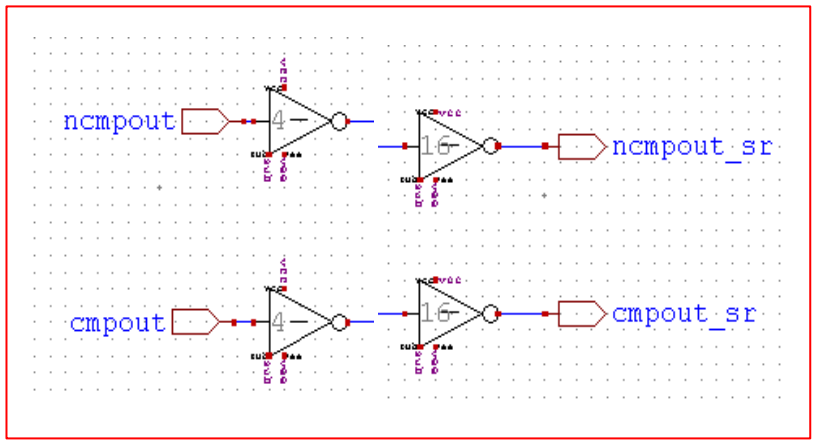
Simplify this part?

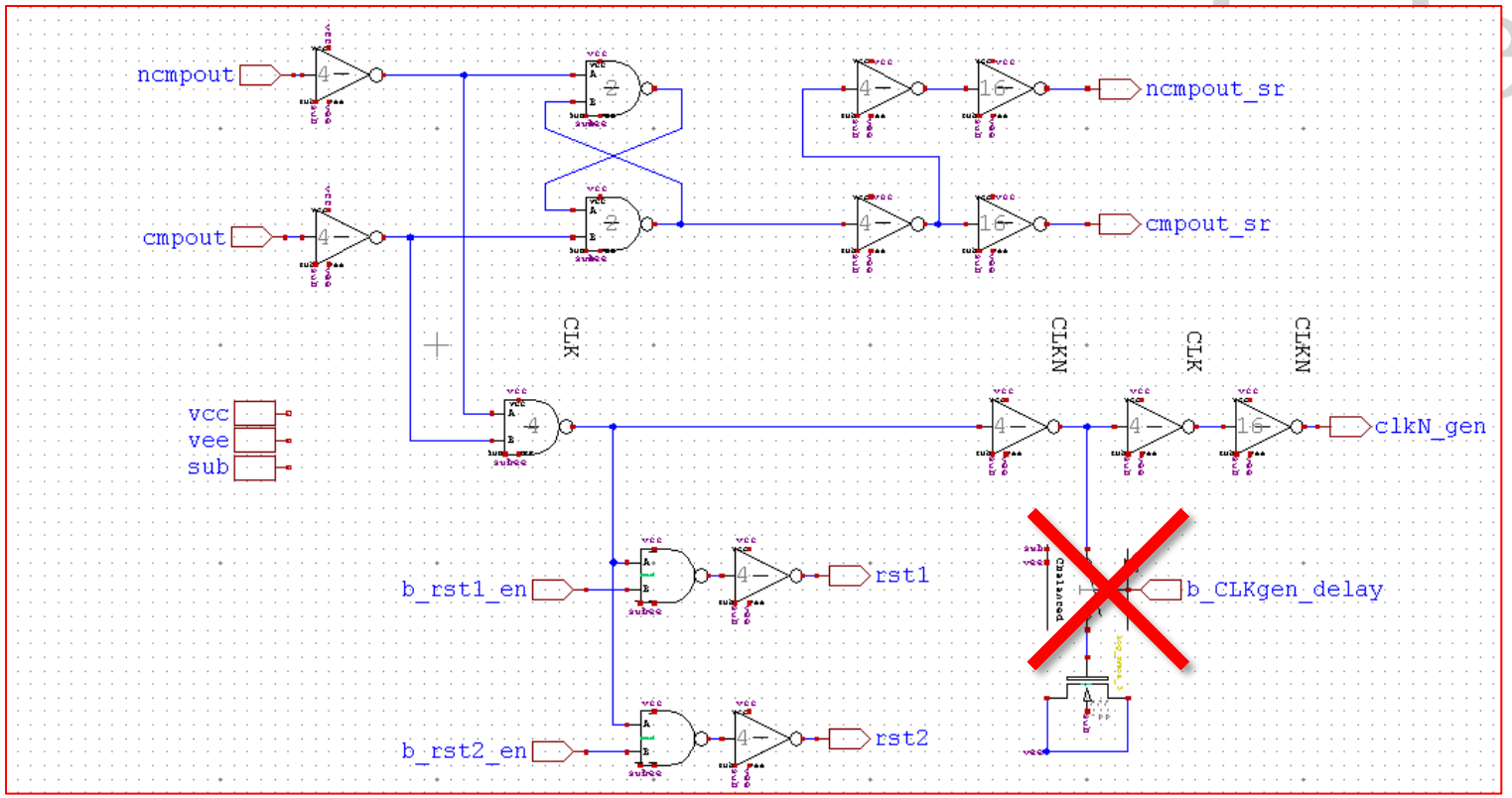
"conditioning" following the comparator latch.

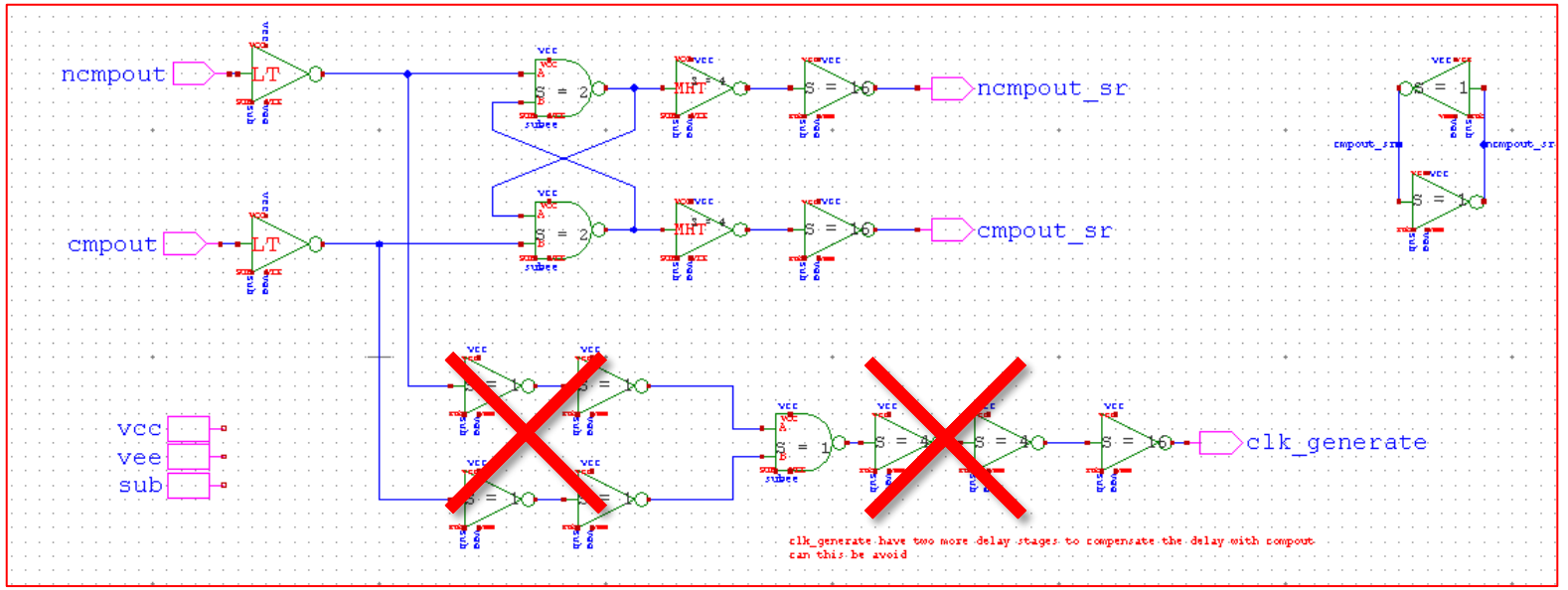
The above is ATON, the below is BALOR. Speed improvements indicated with x

As the relative timing does not matter, why not even the below?

Irrelevant in comparison to other propositions







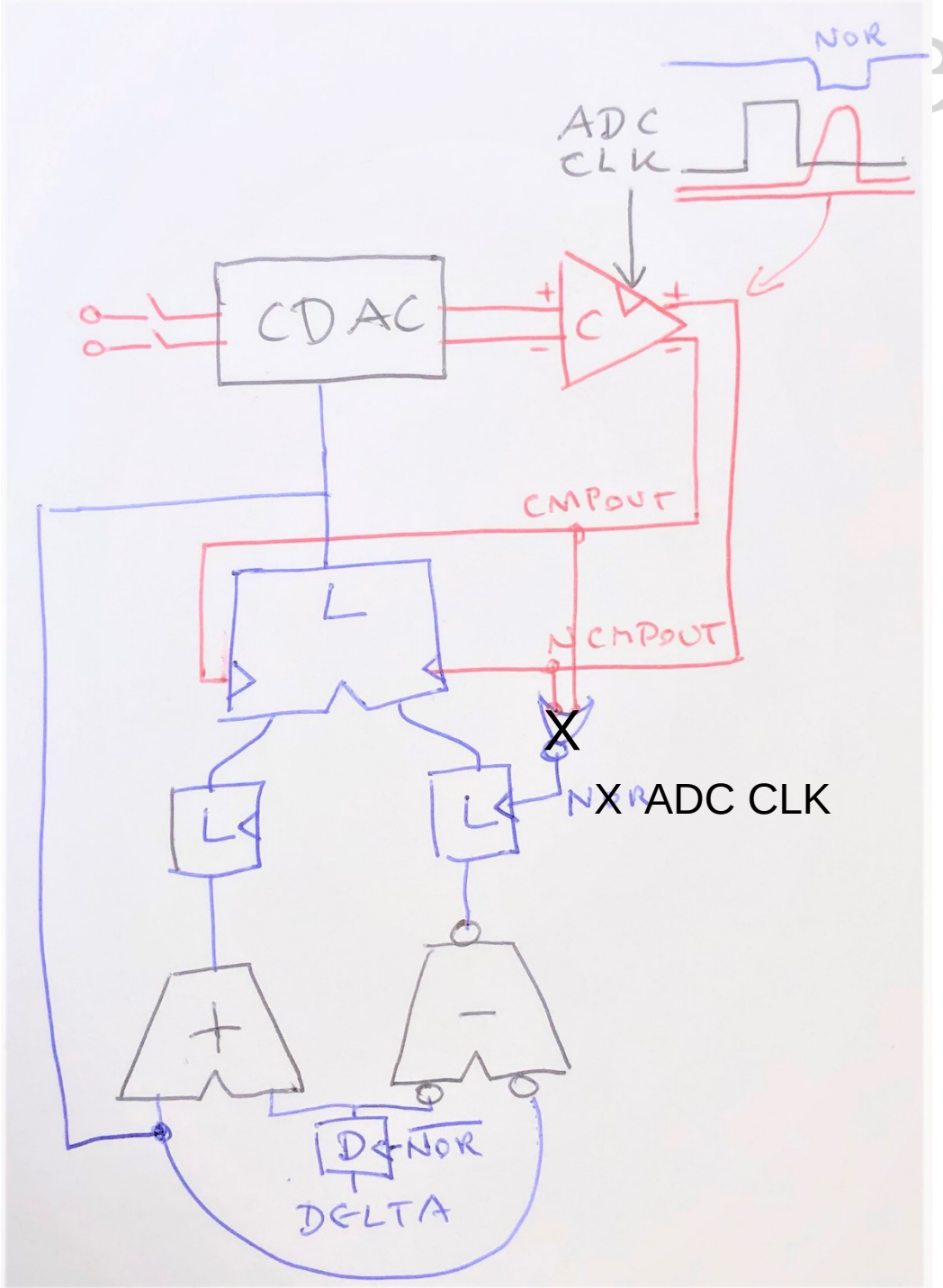
Topology with 2 latches in each loop.

From Notebook 0624:

Configuration where both analog and digital loop have two latches.

Here a small modification, both the Latch after the adder and the comparator latch receive the bare ADC clock.

First prove by simulation that it is really better, then use in next generation



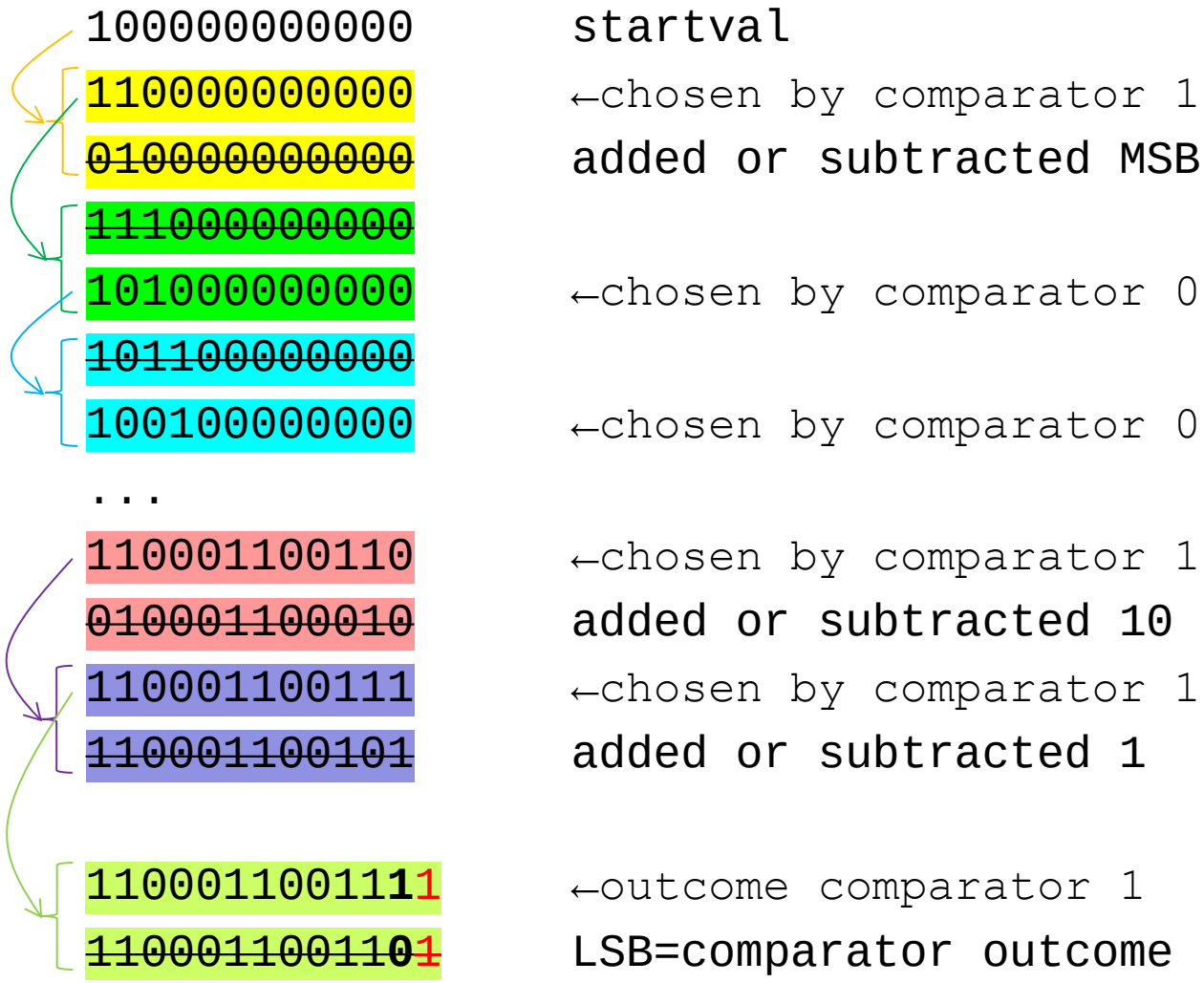
Odd/even effect

Hypothesis, countermeasure

What happens? And *first* countermeasure

One needs 11 comparisons to reach the LSB, and then?

Each next bit is incremented with +1 or -1, and one chooses the one corresponding to the comparator.



The LSB value would need an addition starting at an imaginary LSB+1 (1/2 bit value)

However, instead, one simply can copy the comparator output into the LSB, not needing an addition.

Acknowledgment: this is an older idea by *NN*

The same with postcorrection

caeleste

We start at LSB-2 conversion and then continue with a number of postcorrections.

Yet for ADCout we keep the previous value and put the comparator output on LSB position.

```
110001100110  ←chosen by comparator 1, added 10
010001100010
110001100111  ←chosen by comparator 1, added 1
110001100101
110001101000
110001100110  ←chosen by comparator 0
110001100111  ←chosen by comparator 1
110001100101
110001101000
110001100110  ←chosen by comparator 0
110001100111  ←chosen by comparator 1
110001100101
```

```
110001100110
110001100111
```

```
110001100110
110001100111
```

This seems as well an odd-even effect, but it is not.

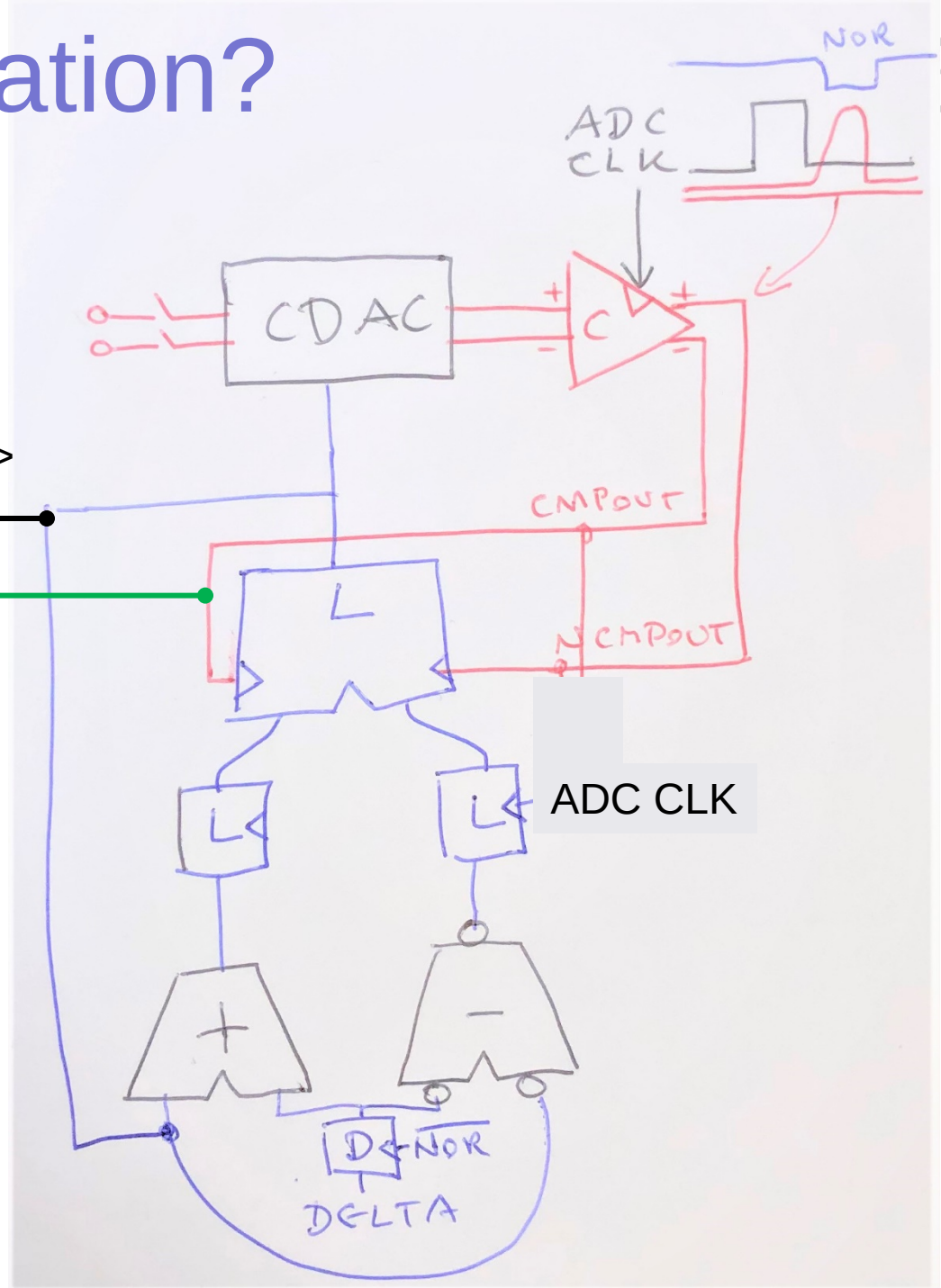
Countermeasure implementation?

With minimal changes in the present ADC one could do:

- Upload the previous value of the adder/subtractor.
    - May need extra dff tapping the CDAC input.
- Yet replace the LSB with the comparator output for the next change.

This may be unconditional, just do it always, either for raw conversion or postcorrection.

BTW in any normal operation, one always reaches the LSB, so no special precautions for non-LSBs are required.



Countermeasure valuable?

“old” postcorrection means LSB increment or decrement

LSB½ increment/decrement is assumed a better way.

Hypothesis: the same result is obtained by copying the comparator output into the LSB?

```
110001100111   ←previous value
110001101000   ←result LSB+1
110001100111   ←result as if LSB½+1
110001100111   ←LSB=comparator outcome
```

```
110001100111   ←previous value
110001100110   ←result LSB-1
110001100101   ←result as if LSB½-1
110001100110   ←LSB=comparator outcome
```

Assuming that the LSB½ is the “best” or “most accurate” approach, then

Old postcorrection gives the same result in half the cases, 1 LSB difference in the other cases.

And just as well copying the comparator output gives the same result in half the cases, 1 LSB difference in the other cases.

```
110001101000   ←previous value
110001101001   ←result LSB+1
110001101001   ←result as if LSB½+1
110001101001   ←LSB=comparator outcome
```

```
110001101000   ←previous value
110001100111   ←result LSB+1
110001100111   ←result as if LSB½-1
110001101000   ←LSB=comparator outcome
```

Thus, apart from removing the odd/even, this countermeasure has no added value.

Solve odd/even in extra ½ conversion

onlyup/onlydown

Shown here: after the “normal” +/-1LSB conversion,

We do one next comparison, but allow only the LSB to increment, or stay the same.

```
110001100111   ←previous value
110001101000   ←result LSB+1
1100011001111  ←result as if LSB½+1
110001100111   ←LSB=comparator outcome
110001101000   ←result LSB+1
```

The result is equivalent to +/- LSB½ (apart from 1 LSB offset). (PS in fact only decrement would be better)

```
110001100111   ←previous value
110001100110   ←result LSB-1
1100011001101  ←result as if LSB½-1
110001100110   ←LSB=comparator outcome
110001100111   ←result LSB-1 not decrementing
```

Implement in next generation: Do one extra comparison after +/-1LSB and allow only to decrement.

```
110001101000   ←previous value
110001101001   ←result LSB+1
1100011010001  ←result as if LSB½+1
110001101001   ←LSB=comparator outcome
110001101001   ←result LSB+1
```

If postcorrection is kept, do it alternating onlydown / onlyup / onlydown as was already in place.

```
110001101000   ←previous value
110001100111   ←result LSB-1
1100011001111  ←result as if LSB½-1
110001101000   ←LSB=comparator outcome
110001101000   ←result LSB-1 not decrementing
```

Thus, in total for a 12bit ADC there are **13** comparisons.

Thus the LSB is affected TWICE, in different ways!

Amended onlyup or onlydown

caeleste

Standpoint/hypothesis: the best odd even removal acts “as if” there is an extra LSB/2 being converted.

The “onlyup” or “onlydown” approaches as presently use are good, however in a noise-free situation, one does not use that extra conversion to home in on the final result, and are thus conceptually worse than as if there is a LSB/2.

Assuming that we are in a situation where the next conversion typically overshoots

because of postcorrection

because of non-binary

Then it is statistically better to do

onlyup when the previous conversion was downward

onlydown when the previous conversion was upward.

Sounds good idea,
however, it is not clear
how to make the circuit.
Only do this when
limited overhead.

caeleste

Alternative ways to oversample

In software, off chip

How to average extra postcorrections

example

Filling a codegap noise

Suppose that the final ADC longword is NNNNNNNNNNNN+1+1+1+1+1+1+1+1+1-1+0-1+1+0-1

The first 9 bits ABCDEFGHI are +1
Initially NNNNNNNNNNN is then corrected to NNNNNNNNNNNN+9

Thereafter, 6 new results are made, and including the corrected itself we can make the average of 7 results:

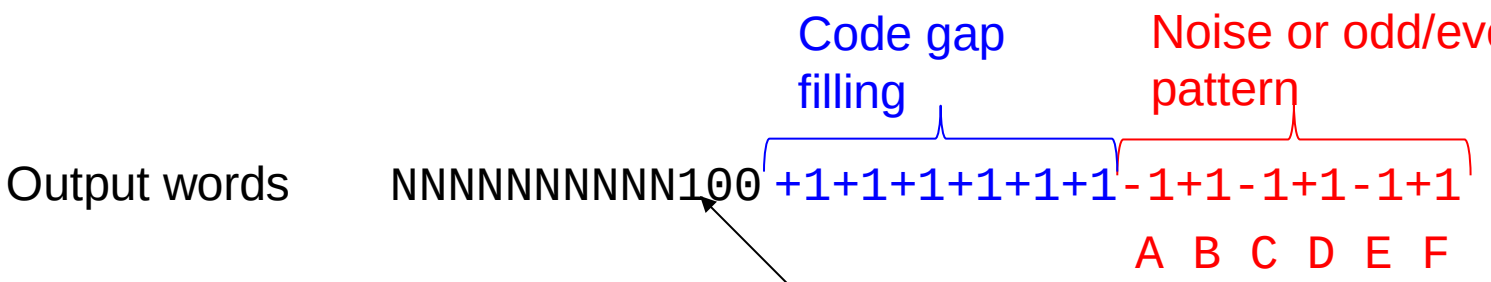
NNNNNNNNNNNN+9
NNNNNNNNNNNN+9-1
NNNNNNNNNNNN+9-1+0
NNNNNNNNNNNN+9-1+0-1
NNNNNNNNNNNN+9-1+0-1+1
NNNNNNNNNNNN+9-1+0-1+1+0
NNNNNNNNNNNN+9-1+0-1+1+0-1

NNNNNNNNNNNN+9+(-1)*6/7 +(0)*5/7 +(-1)*4/7 +(+1)*3/7 +(0)*2/7 +(-1)*1/7

Little problem: division by 7

Off-chip mitigation of odd/even effect caeleste

The off-chip handling of postcorrection can be used to handle the odd-even effect as well.
This allows to have the pure +1/-1 comparator result to be readout as single bits.



Flaw: this does not improve noise below 1LSB??

Reconstructs last 6 codes

NNNNNNNNNN100	this is the final result
NNNNNNNNNN010	minus F
NNNNNNNNNN100	minus E
NNNNNNNNNN010	minus D
NNNNNNNNNN100	minus C
NNNNNNNNNN010	minus B
NNNNNNNNNN011	average

Should be an even number so that a pure odd/even pattern results in half value.

Do not
Vernier is a better concept

Better postcorrection?

Better postcorrection

caeleste

Left: postcorrection as if from an extra LSB-1.

Right: present postcorrection

The better postcorrection gives you one bit more.

```
      -1001.111
1000      -
1100      +
1010      +
1001      -
1001.1    -
1010.0    +
1001.1    -
1010.0    +
1001.11
```

```
      -1001.111
1000      -
1100      +
1010      +
1001      -
1010      +
1001      -
1010      +
1001      -
1001.1
```

```
      -1010.101
1000      -
1100      +
1010      -
1011      +
1010.1    -
1011.0    +
1010.1    -
1011.0    +
1010.11
```

```
      -1010.101
1000      -
1100      +
1010      -
1011      +
1010      -
1011      +
1010      -
1011      +
1010.1
```

```
      -1011.001
1000      -
1100      +
1010      -
1011      -
1011.1    +
1011.0    -
1011.1    +
1011.0    -
1001.01
```

```
      -1011.001
1000      -
1100      +
1010      -
1011      -
1100      +
1011      -
1100      +
1011      -
1011.1
```

```
      -1010.011
1000      -
1100      +
1010      -
1011      +
1010.1    +
1010.0    -
1010.1    +
1010.0    -
1010.01
```

```
      -1010.011
1000      -
1100      +
1010      -
1011      +
1010      -
1011      +
1010      -
1011      +
1010.1
```

vernier

Alternative for postcorrection and oversampling

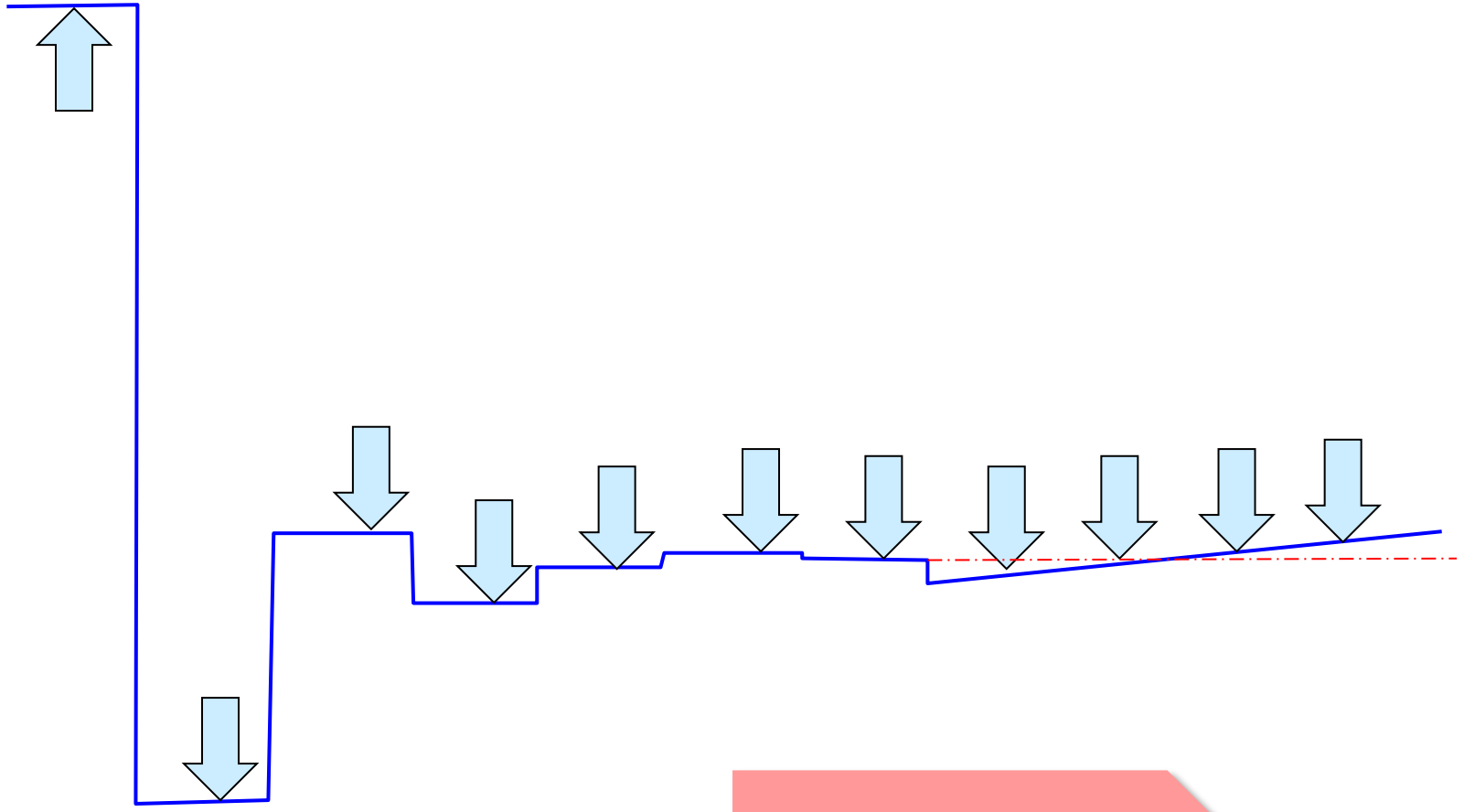
Postcorrection with a vernier

Idea:

- Apply 12 extra comparisons
- Do not recalculate the SAR code (or…?)
- Vernier (sweep) or Dither (pseudo random) an offset to the CDAC.
- Record the 12 extra comparison bits.

May/should span more than one LSB.

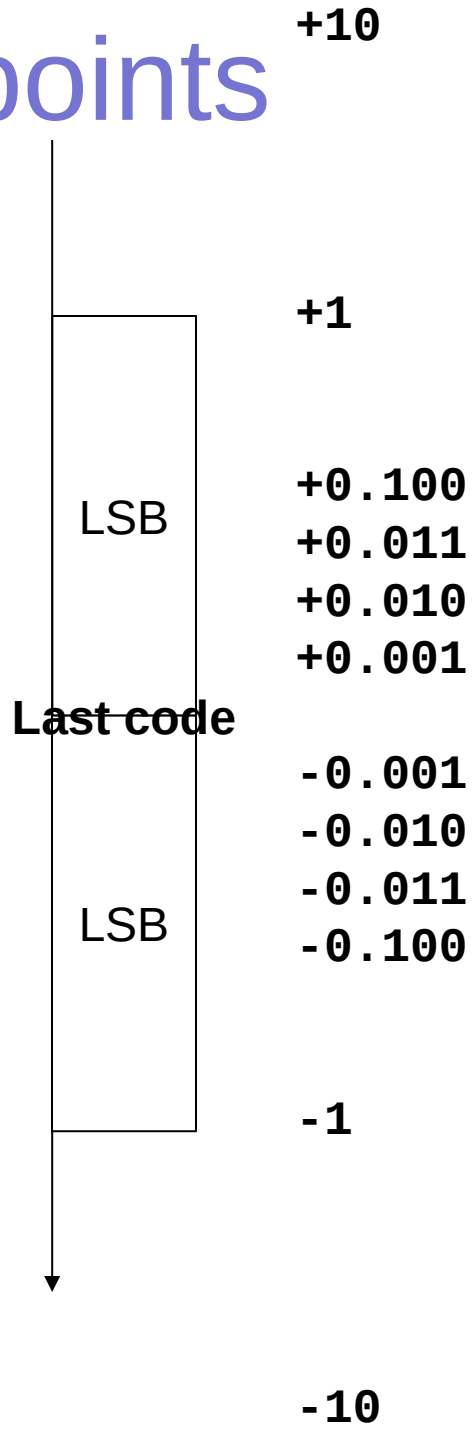
Should not even be equally spread.



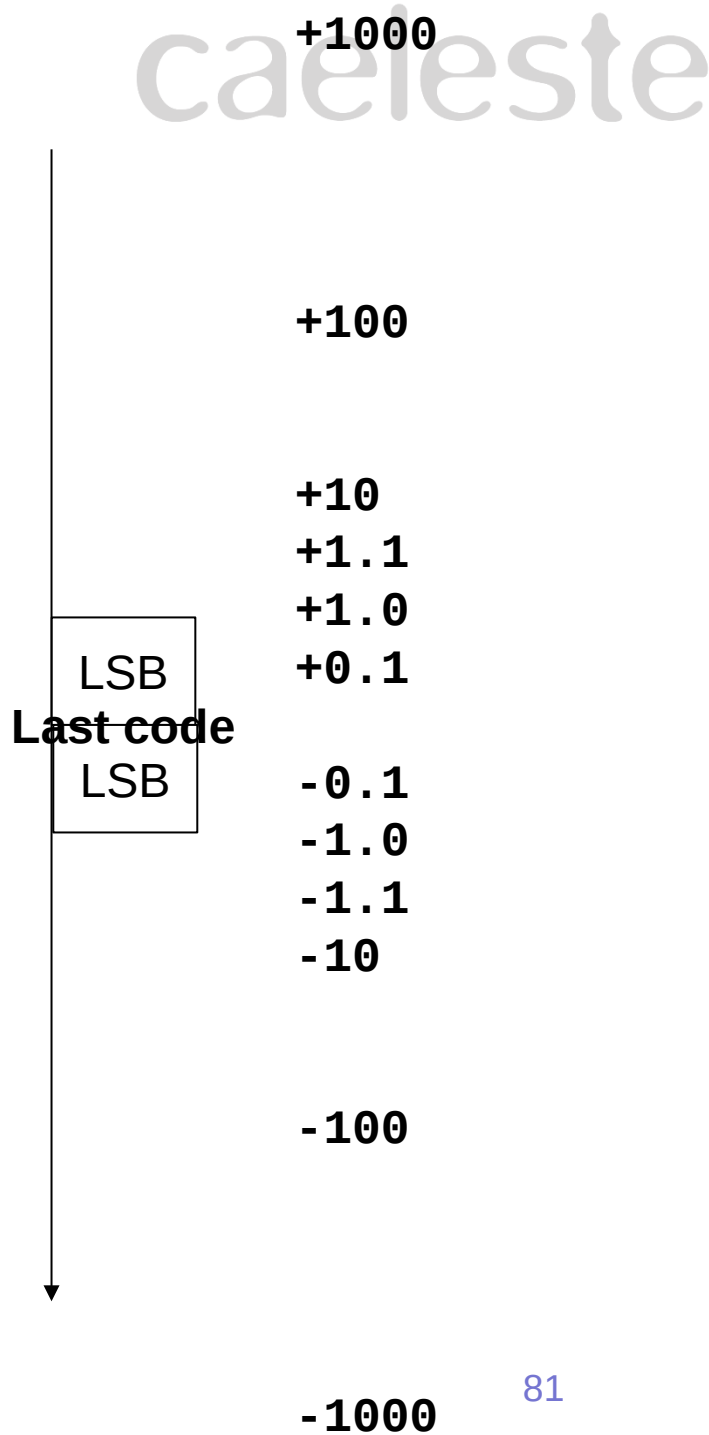
A vernier with 8 positions spanning one LSB, gives you ideally 3 extra bits.

Vernier 12 points

Very optimistic, assuming that the nominal ADC reaches <0.5 LSB error



More realistic, assuming that the nominal ADC reaches ~2 LSB error



Compressing the Vernier output.

As the Vernier is [ideally] a Thermometer code, it can be encoded as a binary number.

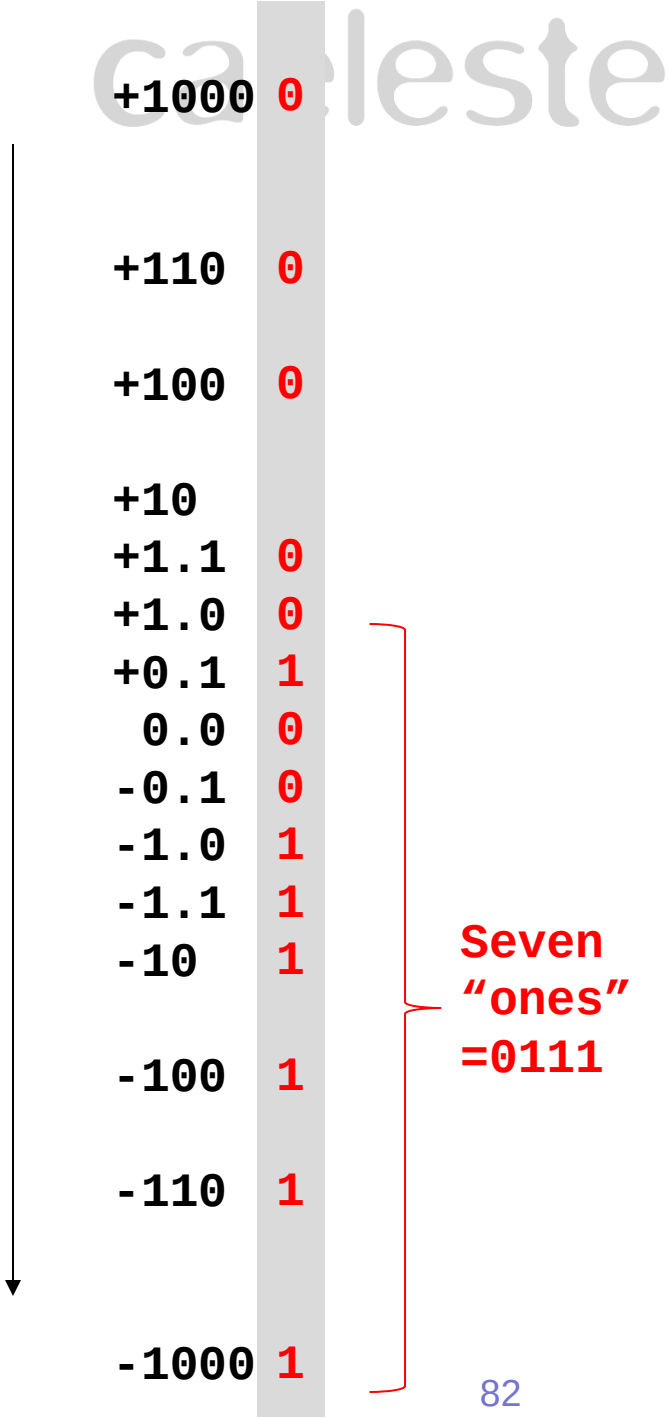
A 15 bit Vernier can be encoded with 4 bits.

The embodiment can be just a 4bit counter.

In “Vernier mode”, these 4 bits are then appended to the original 12 bit.

Not in next generation.
Good idea for later if the
Vernier has proven its quality

Offchip software should interpret
1111 or 0000 as “error reading”



caeleste

Alternative to double conversion

Non-binary variant of the present SARADC

Using steps slightly smaller than 2

Alternative to double conversion

caeleste

Basic SAR:

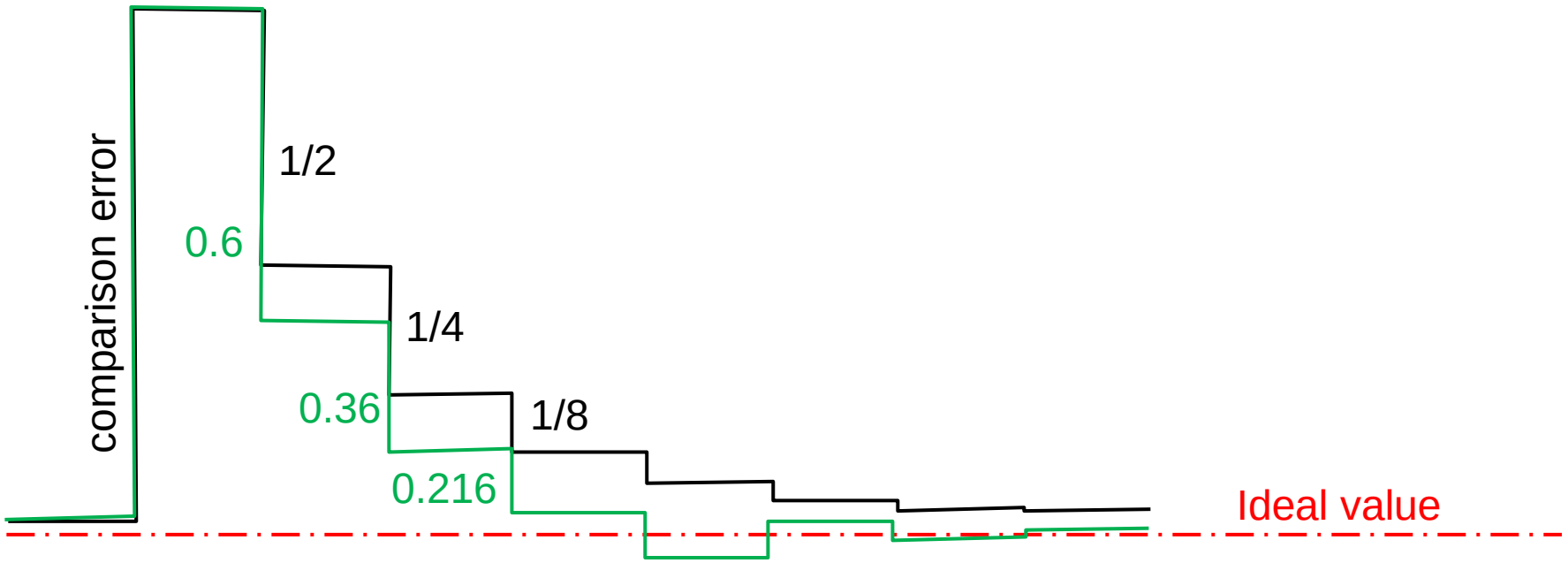
- each next step is half the previous

Double conversion:

- as basic, but some steps are repeated

Alternative:

- next step is larger than half the previous



Alternative to double conversion

Basic SAR:

- each next step is half the previous

Double conversion:

- as basic, but some steps are repeated

Alternative:

- next step is larger than half the previous

How to implement: a simple lookup or combinatoric layer between sequencer and adder

10/23/2024

Factor	2048	100000000000
2	1024	10000000000
	512	1000000000
	256	100000000
	128	10000000
	64	1000000
	32	100000
	16	10000
	8	1000
	4	100
	2	10
	1	1

This is the basic SAR sequence: steps decreasing with a factor 2. the number of SAR steps is 12 for a 12bit ADC.

	…	…	…	…
factor	128	10000000	10000000	128
1,5874	81	1010001	1010000	80
0,6666	51	110011	110000	48
	32	100000	100000	32
	20	10100	10100	20
	13	1101	1100	12
	8	1000	1000	8
	5	101	101	5
	3	11	11	3
	2	10	10	2
	1	1	1	1

↑Here the step decreases with a factor $2^{2/3}$. The number of SAR steps becomes 18. In red simplified/rounded

↓for factor $2^{4/5}$, #SAR steps = 15

	…	…
factor	128	10000000
1,41	91	1011011
	64	1000000
	46	101110
	32	100000
	23	10111
	16	10000
	12	1100
	8	1000
	6	110
	4	100
	3	11
	2	10
	1	1

Here the step decreases with a factor SQRT(2). The number of SAR steps doubles to 24

factor	10000000000	2048
1,741101	10010000000	1152
0,8	10100000000	640
	11000000000	384
	11100000	224
	10000000	128
	1001000	72
	101000	40
	11000	24
	1110	14
	1000	8
	101	5
	11	3
	10	2
	1	1

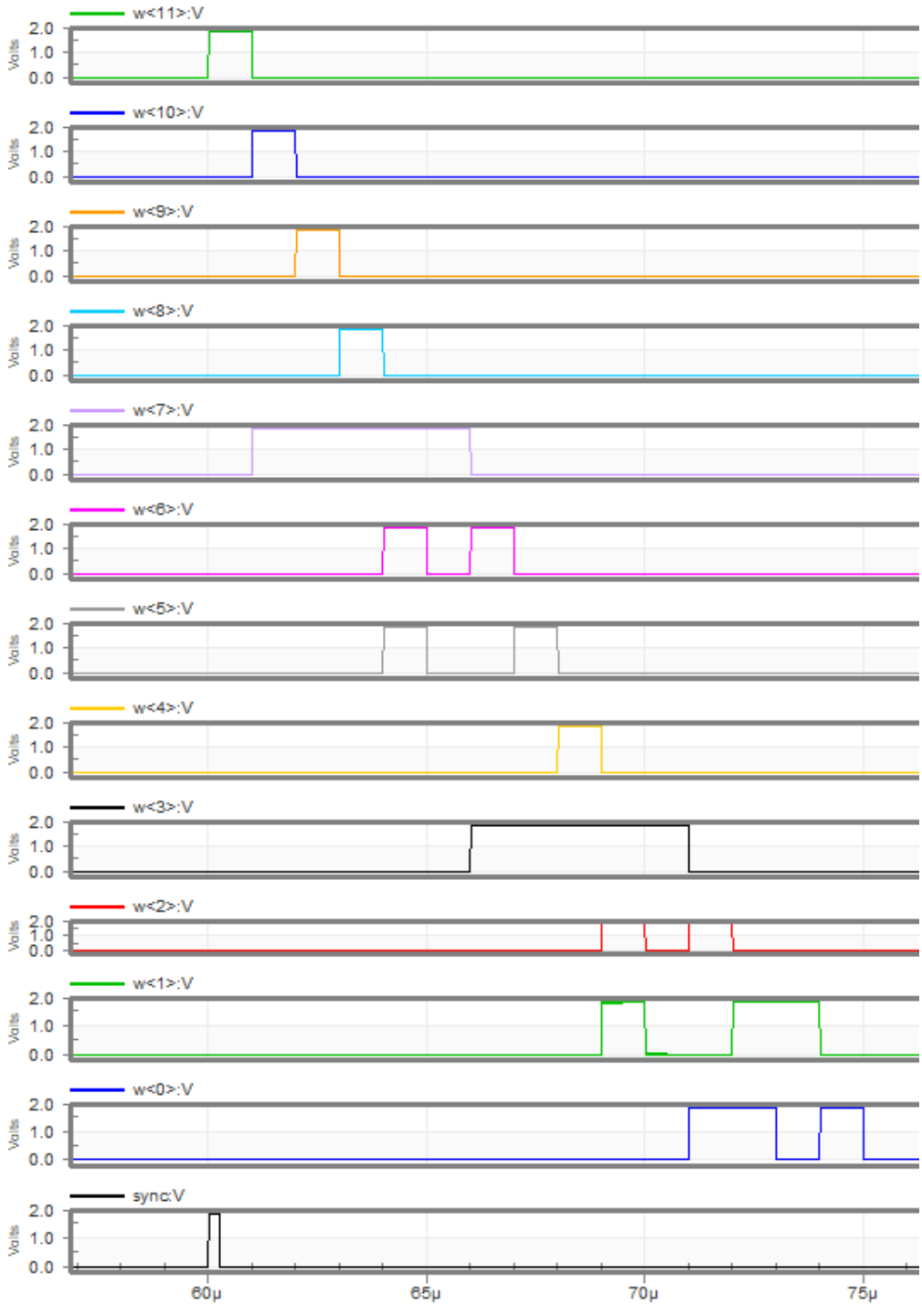
Looks like nice to have

Circuit and simulation

This circuit makes the coefficients for the *divide by 1.741* case. It can be built on top of the shiftregister of the sequencer.

See also Notebook 20221202ba “non-binary SARADC”

	W<11:0>	
factor	10000000000	2048
1,741101	10010000000	1152
0,8	10100000000	640
	11000000000	384
	11100000000	224
	10000000	128
	1001000	72
	101000	40
	11000	24
	1110	14
	1000	8
	101	5
	11	3
	10	2
	1	1

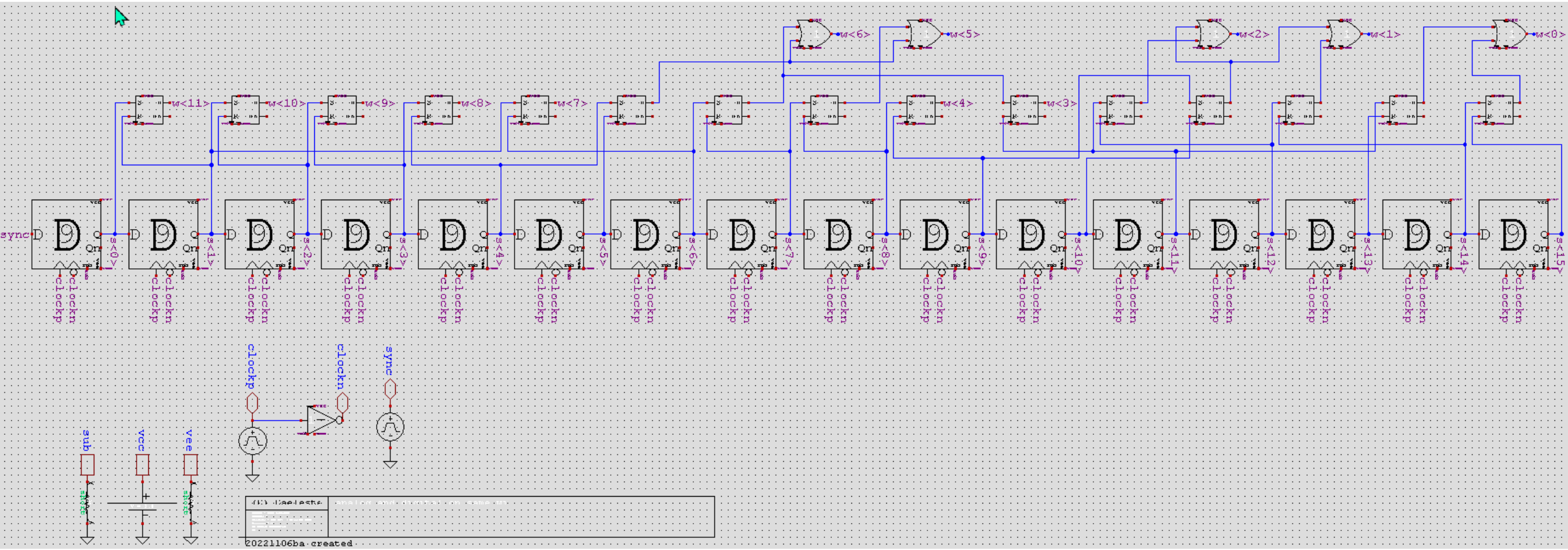


Circuit SB_saradc_factor1741

caeleste

Scib library “SB” Cell “SB_saradc_factor1741”

Uses 15 SRFF and 5 OR2 gates, on top of sequencer shift register.



caeleste

Radical non-binary

See Notebook 20221202ba “non-binary SARADC”

And second section of this course

caeleste

Not interleaving?

Driving many parallel ADCs from one sequencer

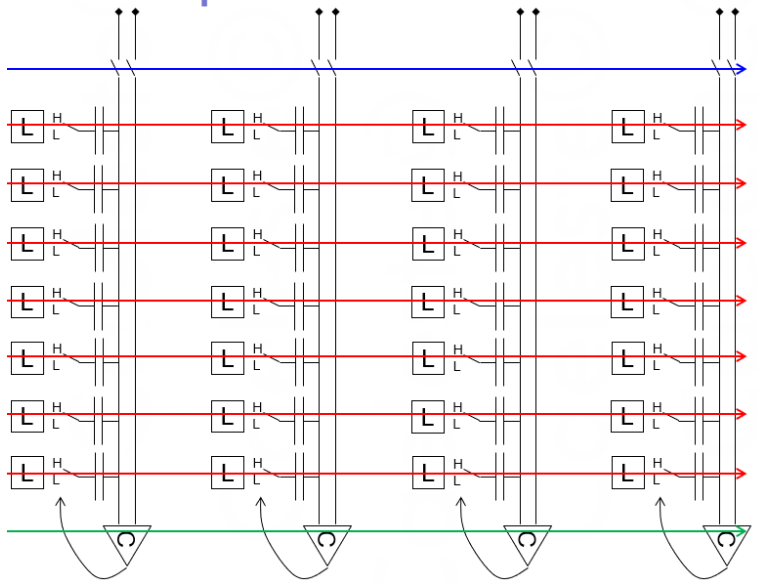
One serializer feeding on many parallel or interleaved ADCs

A videobus can be interleaving or fully parallel.

Driving many ADCs in parallel?

Plain parallel has advantages:
- ✓ adjustability of each period in the field
- ✓ Can skip modes (e.g. skip double conversions or postcorrection)
- ✓ All ADC settle simultaneously and are all quiet when converging.

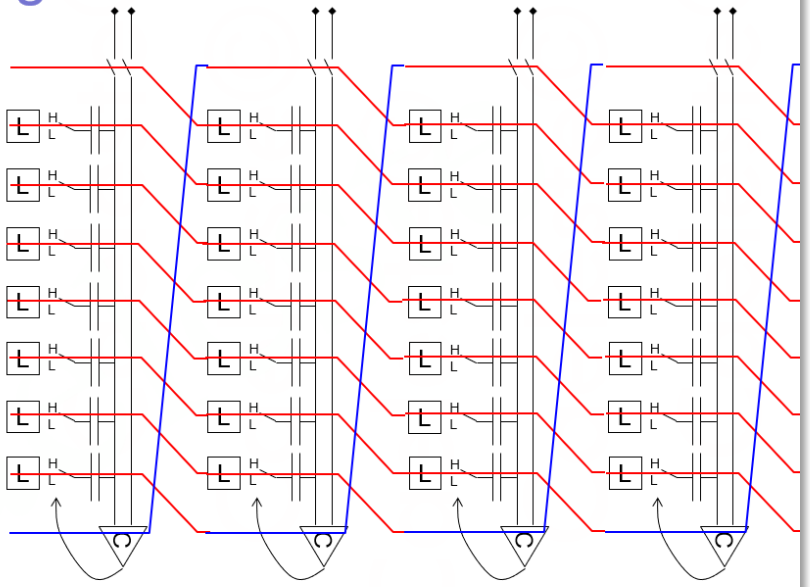
Disadvantages:
- ✓ Input data ready is all at the same time, thus not easy to analog multiplex.
- ✓ Also output data ready all at the same time, making serialization difficult.



This, interleaving?

Interleaving has following advantages
- ✓ Same as present approaches
- ✓ Smoothing of supply current peaks
- ✓ Easy multiplexing of one input signal to many ADCs
- ✓ Easy serialization

Flagrant disadvantages:
- ✓ Fancy rotating routing and wasted clock cycles
- ✓ looses adjustability and skipping
- ✓ Quiet converging ADCs are disturbed by others that are early in the cycle.



How to solve this?

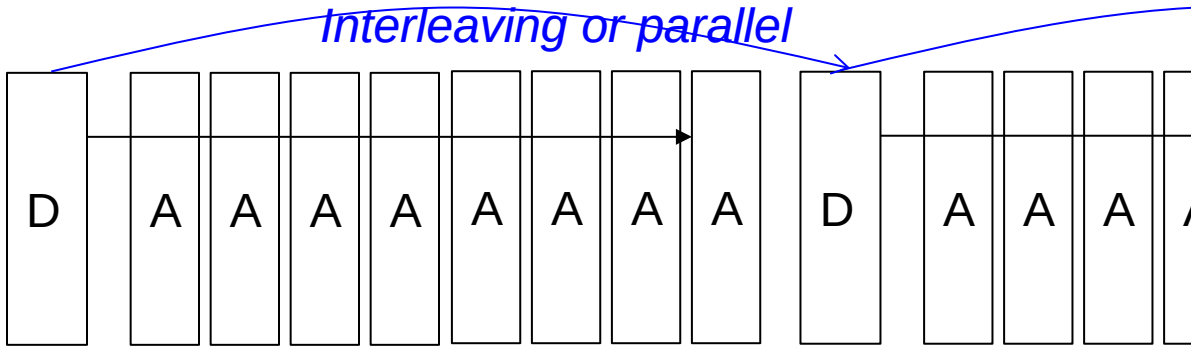
←Notebook 0516

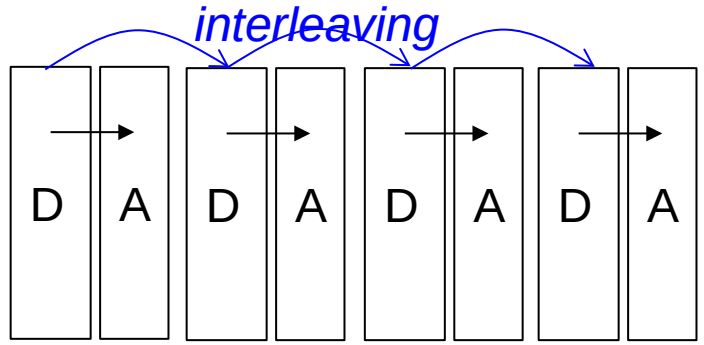
Separate the analog “A” and digital parts “D” of the ADC.

For a large array, one D drives many A, as DAAAAAAAA.

At larger scale, such groups may interleave.

For a small array, or single interleaving ADC, the ADC is a group of D-A pairs.





Single ADC

3 subcircuits: modular approach

caeleste

ADC core

inputs

- Analog input sampling
- Comparator clock
- Adder/subtractor value

output:

- Comparator bits

Sequencer

inputs

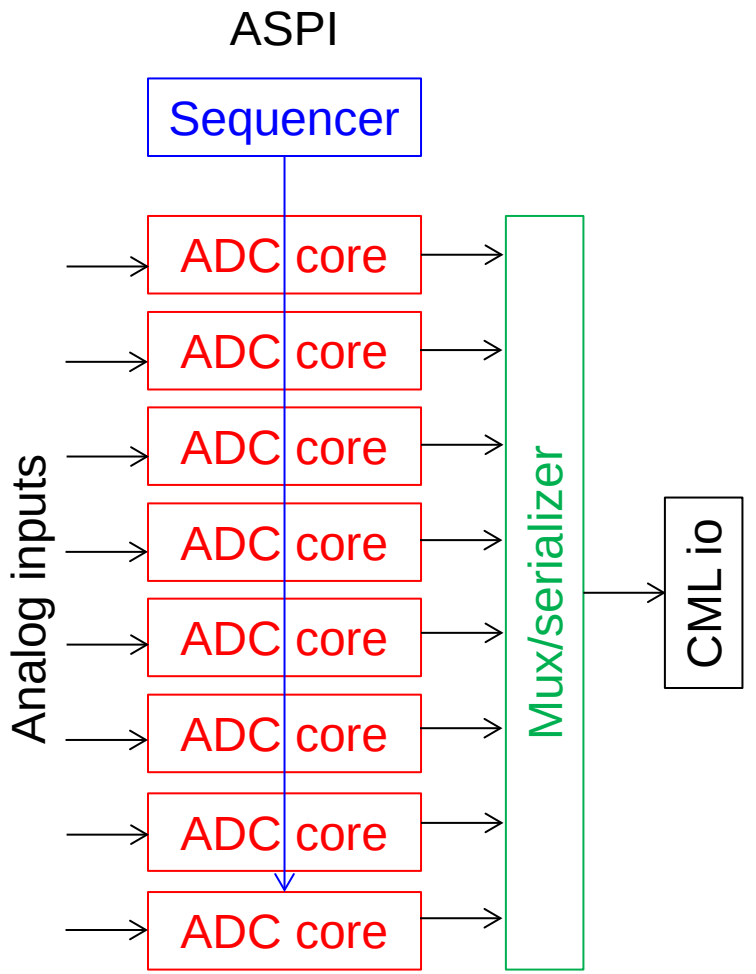
- ADC clock/sync and ASPI bits

outputs

- Adder/subtractor coefficients
- Controls towards ADC core

Serializer

- Inputs: AD converted values and LVDS clock/sync
- Outputs: CML sender (or JESD204?)



ASPI and analog inputs are considered external to the circuit.

VCC/VEE and VHIGH/VLOW are considered externally supplied.

Define and design these three blocks separately with precise interfaces

Serializer should be a 12 (or 16) bit SER

caeleste

Fully 1.8V

Fully operating @ 1.8V?

caeleste

- Can be done
- Already today the digital logic is “LOG” logic, VCC=1.8V
- Also the CDAC switches are LOG
- Today 3.3V is only used in
    - The regulator
    - The comparator input preamp and the tail current sources of the other preamps.
    - The input buffer if any.

- Potential issues:
    - What when migrating later to a LOG VCC=1.2V or lower?
    - What about the ADC might have an input voltage > 1.8V.

caeleste

Alternatives to “SAR”

Adaptive step

caeleste

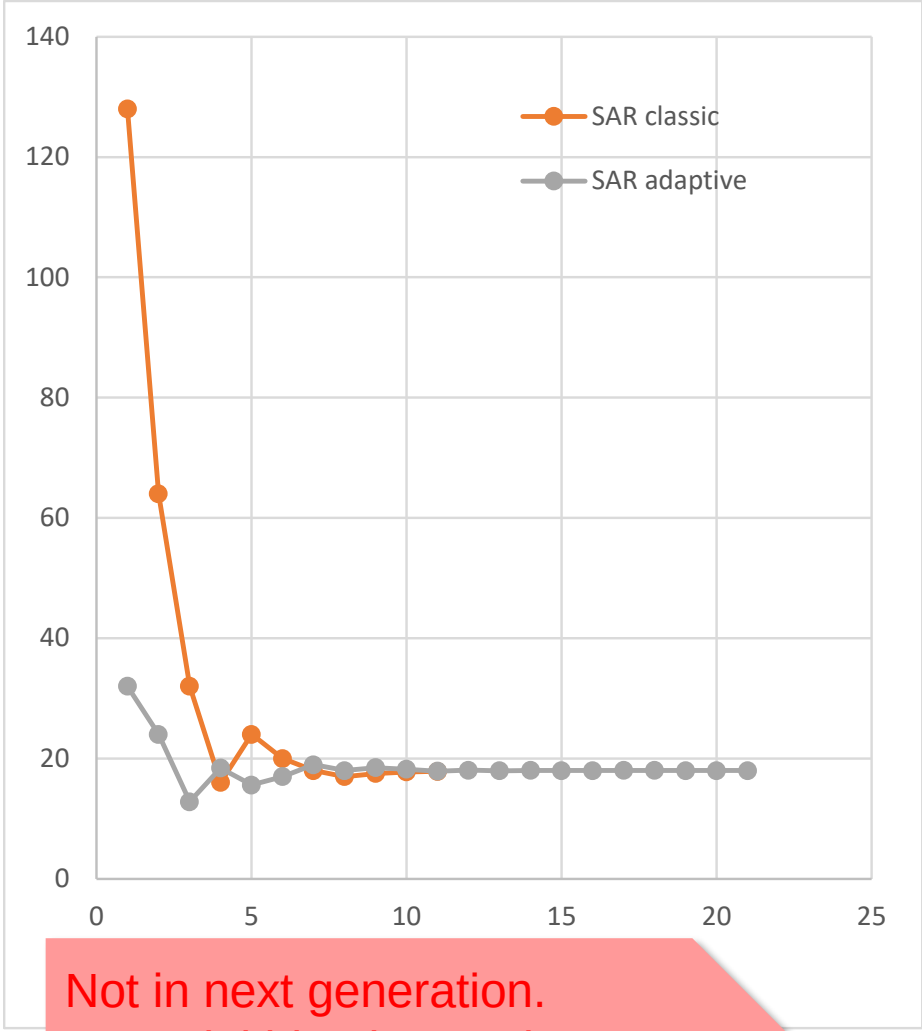
- Present SAR has equal weight/precision, irrespective the code you are at. One does not exploit a need for higher precision in the dark and a relaxed precision in the light, with higher PSN.

One idea:

- Start near the dark (e.g. 0000010, with a limited step height.
- Loop:
  - After comparison, step in the direction of the comparison
  - Find a next step height:
    - If the step direction reverses, decrease the step height with a factor [2x]
    - If the step direction is the same as before, increase it with factor [1.41x]
  - When step height < 1, end loop.

This or similar algorithm is convenient in the adder/subtractor topology

The values 2 and √2 are a sweet spot as these are easy to approximate from the series … 1100 1000 110 100 11 10 1



Not in next generation. Potential idea but we have to let it mature. Probably not compatible with non-binary

Change of FLAMES compared to ATON

Change of FLAMES compared to ATON.

- Input offset cancellation (also as Nova and similar to FPN cancellation of ELFIS2)
- Starting point of group of 8
- Disconnection the dummy CAPs, removed form the comparator inputs.
- Comparator always without 3rd stage. Simulation show that it is always better to remove it.
- Small difference in connection of regulator VSSA->VSSU
- Always new DFF9
- Carry out S=16 instead of S=4.

caeleste

S&H

For low/minimal kTC noise

Knowing the result of Notebook 0653

Foresee a slow switching S&H, eventually an active feedback S&H

section

2. Non-binary SARADC

SARADC10

Goal of this course

caeleste

Understand why we chose the non-binary SARADC concept

- Why differential? Why pseudodifferential?
- Why the modular approach with split between Sequencer and Core ADC
- Why and how to combine the modules in interleaving and full parallel?

Understand the CDAC concept, how to construct this CDAC

- Why non-binary
- Why the series capacitors

How to calibrate the non-binary

- On-chip versus off-chip nb2b
- How to obtain the coefficients for nb2b

Understand the comparator, clocked or not clocked, its auto-zeroing, its output signals.

Understand the array aspects

Understand and contribute to future improvements

- Improving speed / power
- Increase resolution
- Reduce large conversion errors, reduce noise, improve DNL/INL, improve PSR
- Strive to a set of modular cells with limited parametrization

SARADC10:
non-binary, short specsheet

caeleste

- Concept design: as “radical non-binary” in Notebook 0645 20221216ba non-binary SARADC.pdf
- Topology: non-binary
- Target number of bits: 12 to 14 when converted back to real binary
- Input: Accepts fully differential and pseudodifferential
- Internal number of (non-binary) bits: 16
- Non-binary factor “radix”: ~1.8
- Conversion non-binary to binary:
    - on-chip with a 16bit/16word adder
    - In-line (firmware) or off chip (software), i.e. acting on the non-binary word
- Can easily run with less bits by simply having shorter SAR sequence and not serializing the extra bits.

The true purpose of “non-binary” is to solve conversion errors and code gaps. It acts on every SAR conversion.

Also difference form earlier topology:

- Modular separation sequencer from core ADC: can easily make both parallel and interleaving
- #ADCs in an interleaved group: baseline 8, yet also 16 (24, 32..) envisaged.

SARADC10: non-binary

caeleste

Comparator

- Topology strongly influenced by Millipede ADC design
- Comparator keeps 3.3V capability, as we cannot guaranteed low input voltages.
- 1st stage remains the same. Experiment to add autozero in first stage as a programmable option not satisfactory and not maintained.
- 2nd stage is bypass-able.
- 2nd stage calibration switch 1.8V NMOS replaced with a 3.3V one. Avoids issues with high common-mode and speeds up the calibration. (is this justified)
- Reset of the 1st and 2nd comparator stages is generated as in Millipede.
- 3rd stage removed as Monte Carlo simulations showed no offset cancellation benefits.

CDAC

- Fully non-binary. No unary part, no RDAC part, no guardrings.
- Subranging with capacitive division
- “split capacitor” section to allow Vhigh-Vlow=1.8V, enables noise minimum shift
- “offset capacitor” section to offset S versus R ranges when calibrating
- Start-from-the-middle (monotonic) switching: no odd-even artifact. Possible drawback is common mode drift.
- No design for matching as all mismatch compensation should be included in the non-binary

subcircuits: modular blocks

caeleste

ADC core

inputs

- Analog input sampling
- Comparator clock
- SAR bits

Sequencer

inputs

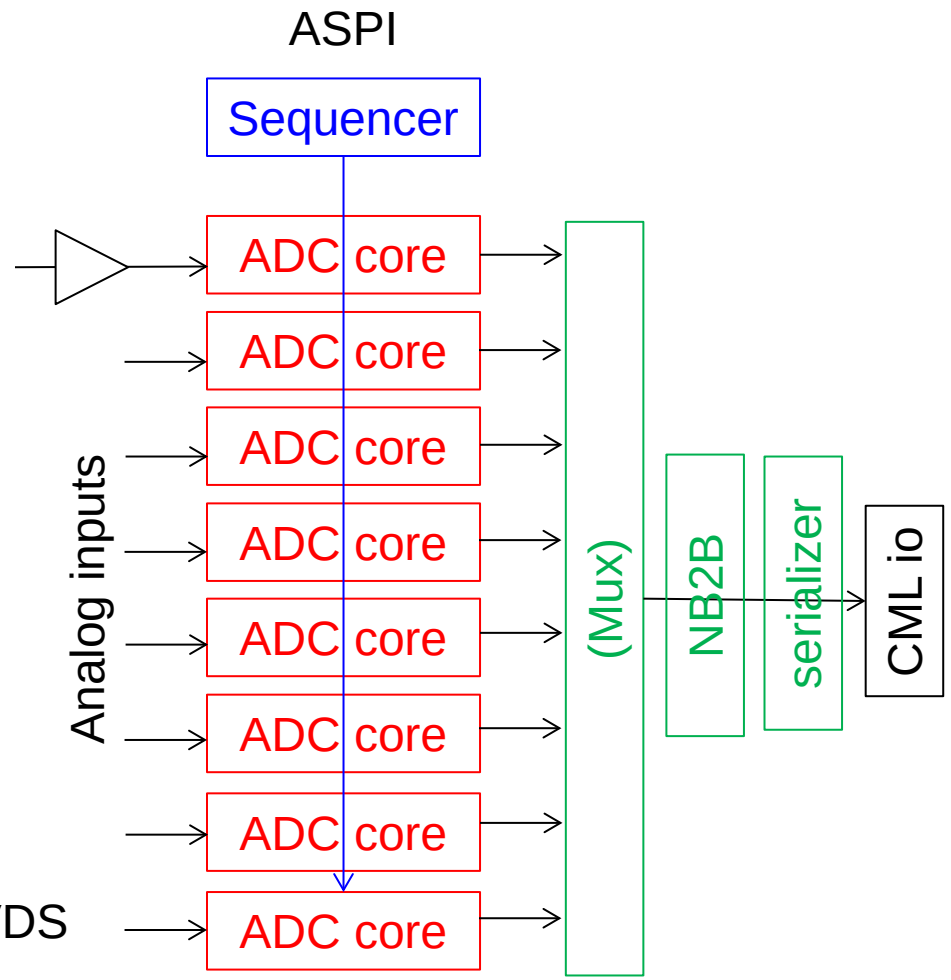
- ADC clock/sync and ASPI bits

outputs

- Controls towards ADC core

Serializer

- Inputs: AD converted values and LVDS clock/sync
- 16 word NB2B convertor
- Outputs: CML sender



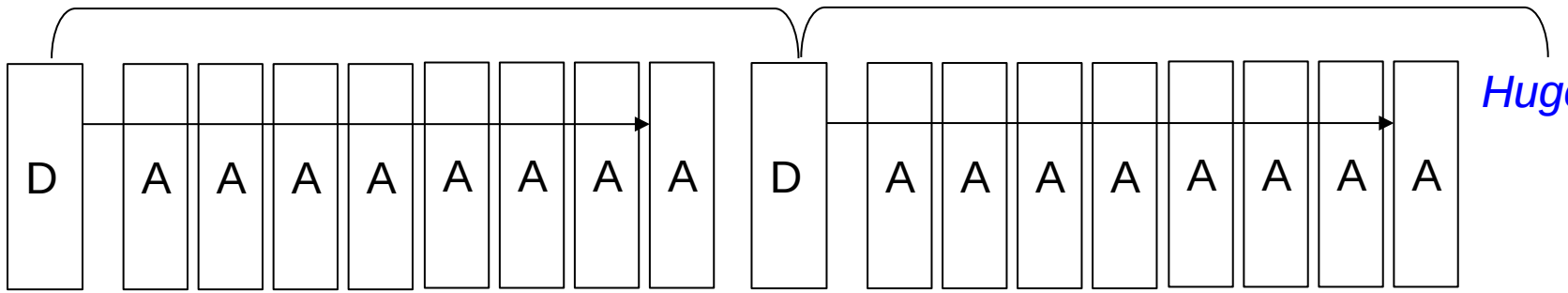
ASPI and analog inputs are considered external to the circuit.

VCC/VEE and VHIGH/VLOW are considered externally supplied.

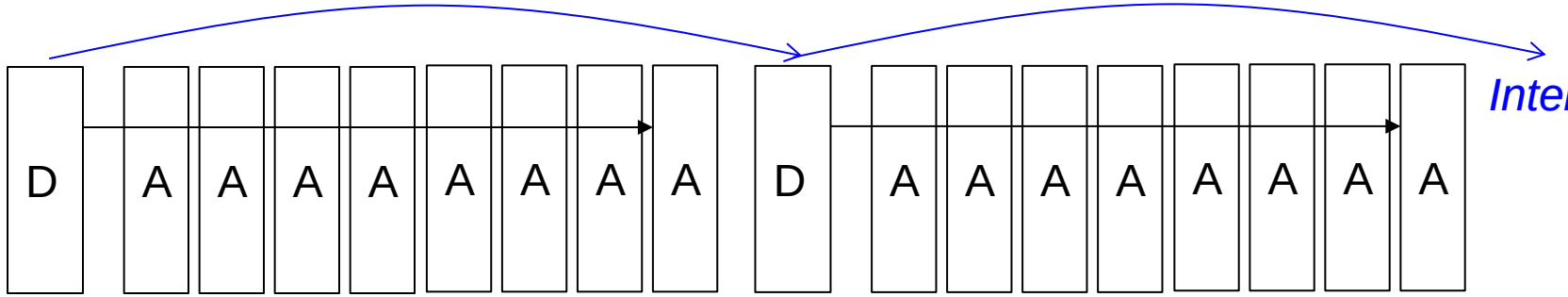
Serializer handles 12 (adjustable 8…16) bit words.
This word length depends on lvds_clock / lvds_sync

Suitable for parallel and interleaving

The modular blocks are (should be) suitable for many different parallel and interleaving floorplans.

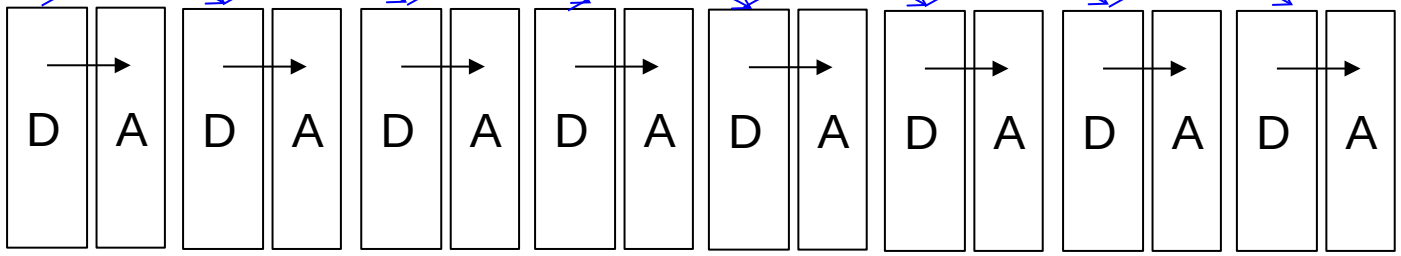


Huge parallel array of SARADCs



Interleaving of smaller parallel groups

Interleaving unit SARADCs



A: core ADC
D: sequencer

Towards a real standard cell (1)

caeleste

The modules should be considered as rigid standard cells, with no or very limited parametrization

- The sequencer
  - The sequencer is capable to drive both parallel and interleaved organizations, and also active sampling if needed. There is no variant schematic
  - The only parameter is S of the control outputs towards the ADC core. Advised S=1 (slow), S=4 (driving one core, interleaved) and S=16 (driving parallel)

- The input stages
  - There are two variants in library SARADC
    - inputbuffer: just a buffer with bypass
    - inputbuffer_feedback: the same buffer, with capability to receive the feedback from SAMPLE_active (active sampling)
  - Parameters:
    - S=4, 16, 64 or 256 variants of the pdiff_keepwarm (SARADC10: W=40 thus S=133=~128)
    - Internally S/2

  In principle the input stages must not be part of the ADC, except when using the “active sampling” feature

Towards a real standard cell (2)

caeleste

- The Sample & Hold part inside the CDAC as a whole
  - There are two variants of this cell, in library SARADC_CDAC
    - SAMPLE: the classic approach
    - SAMPLE_active: having active sampling and requiring access to the feedback inputs of the input stages
  - There should be a single parameter S (internally one should derive S_sample=S, S_conn=S/4, S_cal=S/8). Advised values are S=64 and S=16, congruent with parameters used in CDAC.

Towards a real standard cell (3)

caeleste

- The CDAC proper part is topology rigid.
  - It is conceived to have 16 conversions. As the LSB is small in size, we do not envisage to make N=14 or N+12 variants. Having less nominal bits is just a matter of having less ADC clocks
  - There are three parameters in the schematic cell
    1. The Cmodel, type of capacitor
    2. The strength of the switches. Parameter S applies to the MSB-switch of the nonbinary CDAC proper. All other switches (including those of Split and Level capactitors) refer to that. Proposed values are S=16 (SARADC10) and S=4
    3. Value of capacitors, now implemented as W and L, thus Area. The parameter given are Wmsb and Lmsb of the MSB capacitor of the non binary CDAC prpoper.

       These must be chosen so that C_{MSB}=1pF (nominal), 0.25 or 4pF. These determine the kTC noise of the ADC, hence:
       - C_{total}=7pF: nominal, assumed for 14 conversions thus about 12 equivalent binary bits
       - C_{total}=1.75pF: high speed/low power, for less than 12 equivalent binary bits
       - C_{total}=28pF: “low noise”, for 16 conversions thus about 14 equivalent binary bits

Towards a real standard cell (4)

caeleste

- The comparator
  - Should have a single parameter S. Advised values S=128 (present nominal) and S=32. Internally one derives S_preamp=S, S_offset=S/4, S_latch=S/16.
  - This S affects the 1/f and kTC noise of the comparator. Hence one should use S=128 for low noise and high resolution. S=32 (or even less) for higher noise / low power.
  - As the comparator is presently the dominant dissipator one must consider S=32 or even less in applications. Note that also the tuner setting affect this dissipation.

Notes:

1. In future designs, the comparator might evolve to a pure switching comparator.
2. Comparator dimensioning requires a thorough study , including the effects of calibration and auto-zero-ing on noise.

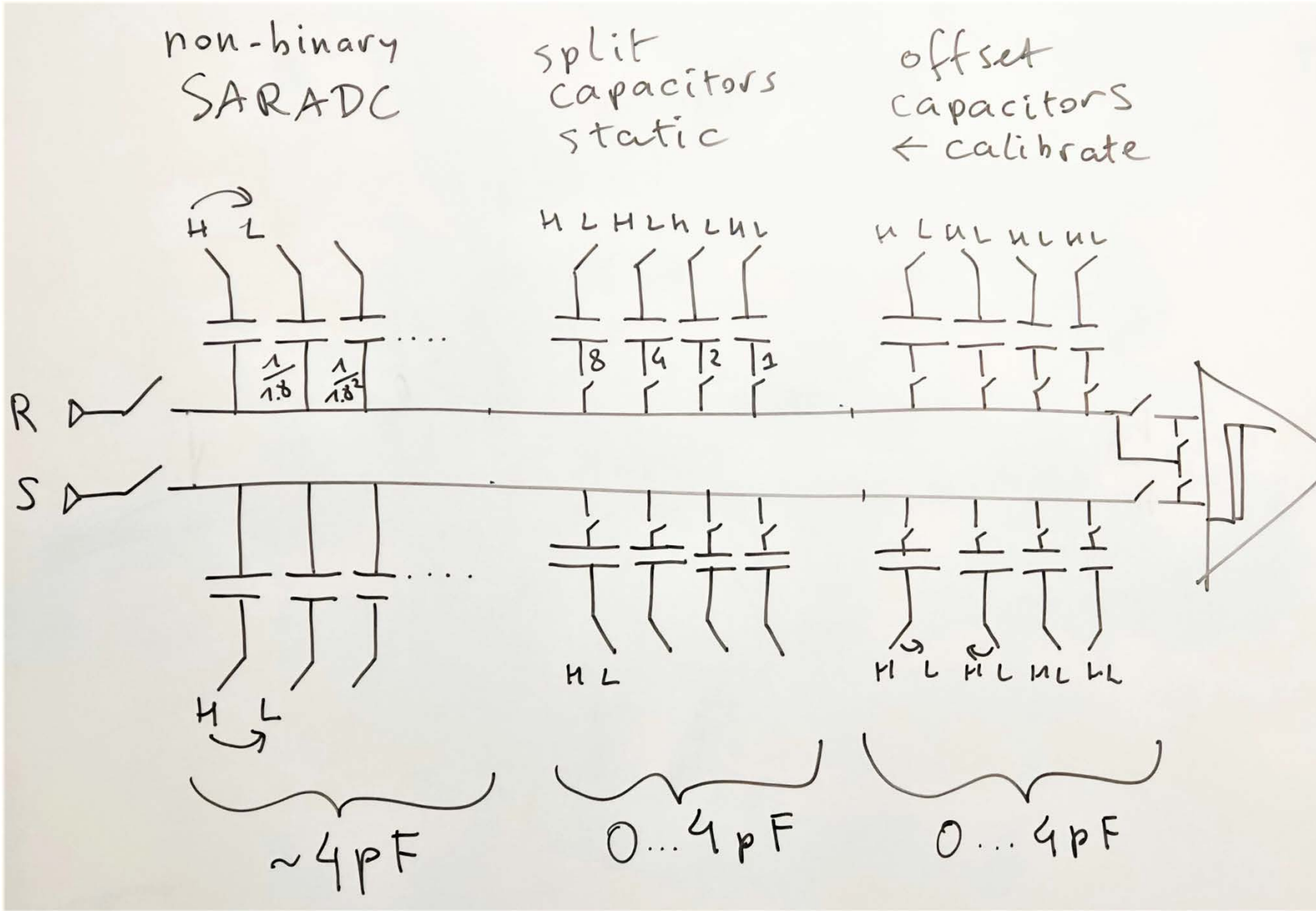
Non-binary CDAC

CDAC, overview

caeleste

The CDAC capacitors are in three groups of about equal total capacitance.

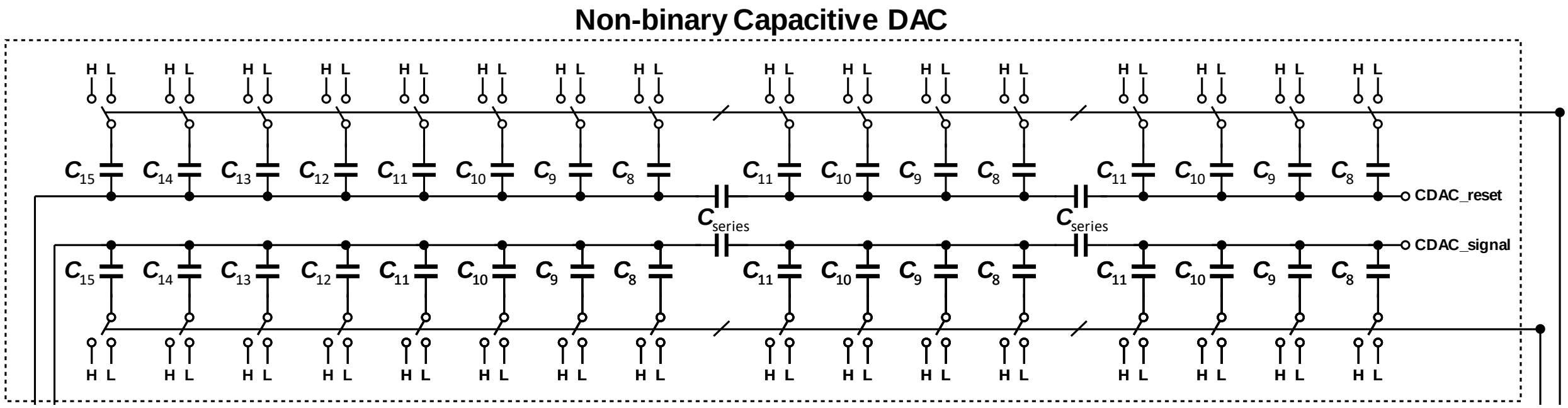
1. The **SARADC proper**, operating in a “SAR” fashion, monotonic and non-binary.
2. The statically biased **split capacitors**, serving gain reduction and the possibility to do noise minimum shift.
3. The dynamically biased **level shift capacitors**, serving to level shift R versus S inputs.



Non-binary capacitor array

caeleste

- The CDAC consists of a 16-bit capacitor array with <2 factors (radix). A radix of ~1.8 is used in SARADC10
- Series capacitors are used to reduce the ratio between the smallest unit capacitor(C_{unit}) and the largest capacitor(C_{MSB}).
  - Example: in brute force n=16, thus $C_{LSB} = \frac{C_{MSB}}{1.8^{15}} \rightarrow \frac{C_{MSB}}{C_{LSB}} \approx 6750$
  - When using C_{series}, n is limited to 8, thus: $C_{LSB} = \frac{C_{MSB}}{1.8^{7}} \rightarrow \frac{C_{MSB}}{C_{unit}} \approx 60$



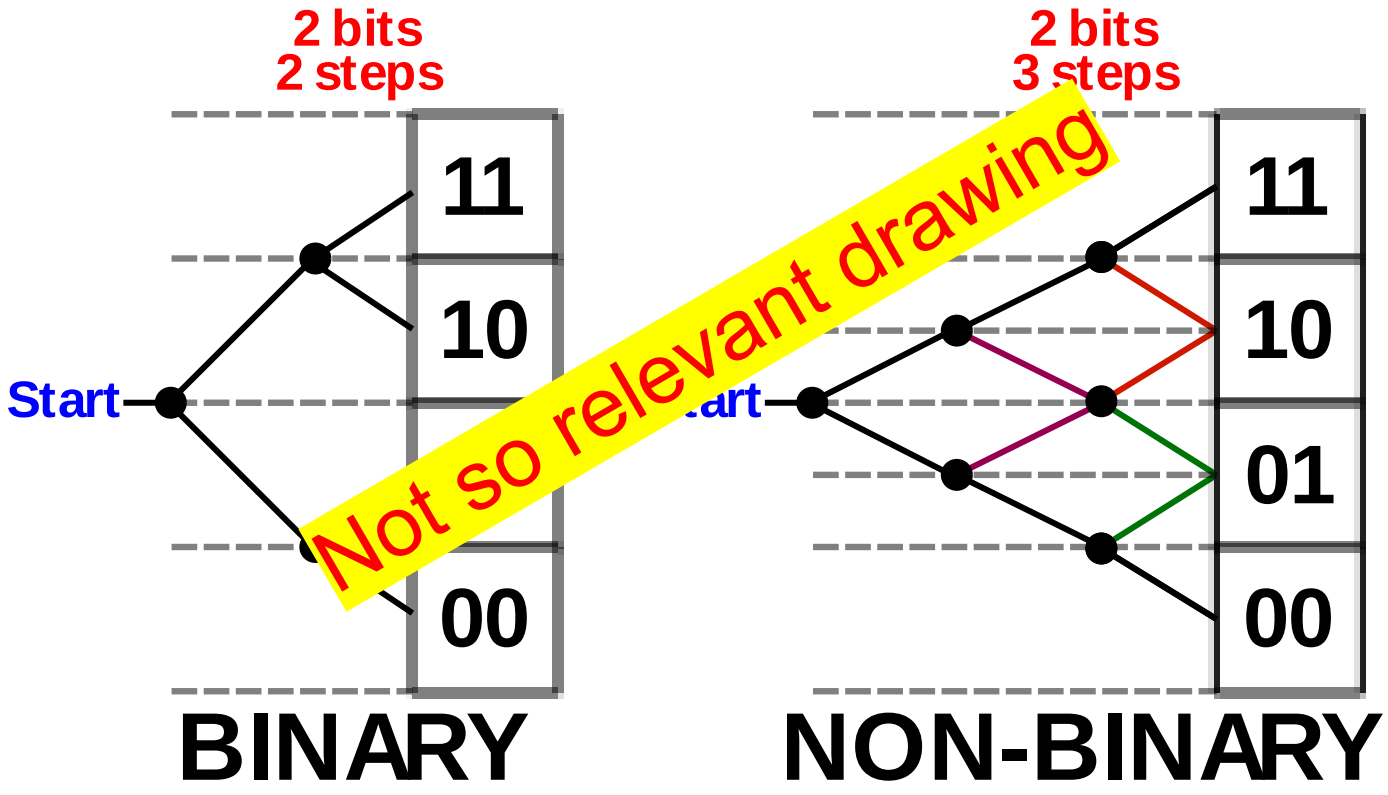
Non-binary capacitor array (Cont'd)

- A binary-weighted SARADC is vulnerable to both static and dynamic errors. Its conversion efficiency, where each subsequent step is exactly halved, means that any decision error cannot be recovered.
- A redundant, non-binary, architecture solves this problem by introducing overlap between the conversion range

- The radix (factor 1.8) value used depends on the ADC resolution and the number of conversion(CDAC) steps.
- The radix should also accommodate capacitor mismatch to avoid the scenario where the ratio between two capacitors exceeds 2 introducing irrecoverable decision levels.
- The effective number of bits can be estimated as:

$$ENOB \leq \log_2\left(\frac{\alpha^{N-1}-1}{\alpha-1}+1\right)$$

where α is the radix and N is the number of CDAC bits(conversion steps).



Equivalence of number of bits

the range of a non-binary ADC is larger than 2^N. For radix 1.8 it is up to 25% wider, or 0.3 bits

$$ENOB \leq \log_2\left(\frac{C_{total}}{C_{LSB}}\right)$$

$$C_{total} = \left(\sum_{i=0}^{N-1} \alpha^i C_{LSB}\right) + C_{LSB}$$

$$\approx \frac{\alpha^N - 1}{\alpha - 1} C_{LSB} + C_{LSB}$$

$$\Rightarrow ENOB^* \leq \log_2\left(\frac{\alpha^N - 1}{\alpha - 1} + 1\right)$$

* Liu, Wenbo, and Yun Chiu. "An equalization-based adaptive digital background calibration technique for successive approximation analog-to-digital converters." *2007 7th International Conference on ASIC*. IEEE, 2007. DOI: 10.1109/ICASIC.2007.4415624

Our own derivation, based on the relative magnitude of the LSB:

$$N_{binary} = 1 + \log_2\left(\alpha^{N_{nonbinary}-1}\right)$$

For large N these are ~equal. For small N, our formula feels more correct.

alfa	1,8		
$N_{nonbinary}$	Liu Equiv. N_{binary}	Our Equiv. N_{binary}	Account for wider range equiv. N_{binary}
1	1,70	1,00	1,15
2	2,34	1,85	2,05
3	3,05	2,70	2,94
4	3,82	3,54	3,81
5	4,62	4,39	4,68
6	5,44	5,24	5,54
7	6,28	6,09	6,40
8	7,12	6,94	7,25
9	7,96	7,78	8,10
10	8,81	8,63	8,95
11	9,65	9,48	9,80
12	10,50	10,33	10,65
13	11,35	11,18	11,50
14	12,19	12,02	12,34
15	13,04	12,87	13,19
16	13,89	13,72	14,04

RA

Non-binary capacitor array (Cont'd)

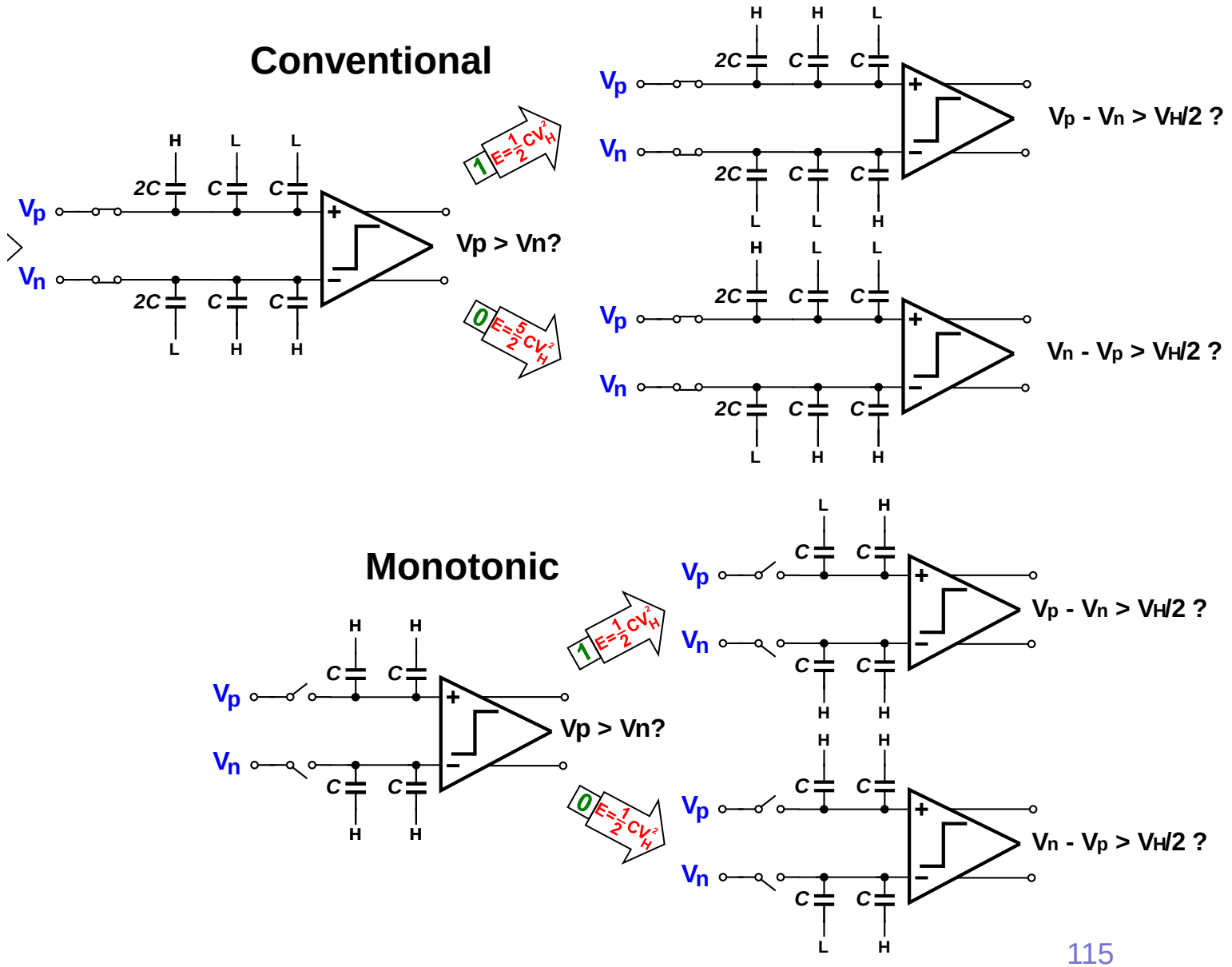
- Another advantage to redundancy is that the size of the CDAC capacitor array can be dimensioned noise-limited rather than matching-limited, leading to a reduction in size.
    - No requirement for exact matching, thus the design style based on a unit capacitor can be abandoned.
    - No requirements for guardings. Any mismatch due to a missing guardring is solved as well by the non-binary to binary conversion.

- A smaller capacitor array leads to:
    - Lower area
    - Lower power consumption
    - Faster settling

- Redundancy relaxes settling constraints as well since errors due to incomplete settling are recoverable.

Monotonic switching scheme

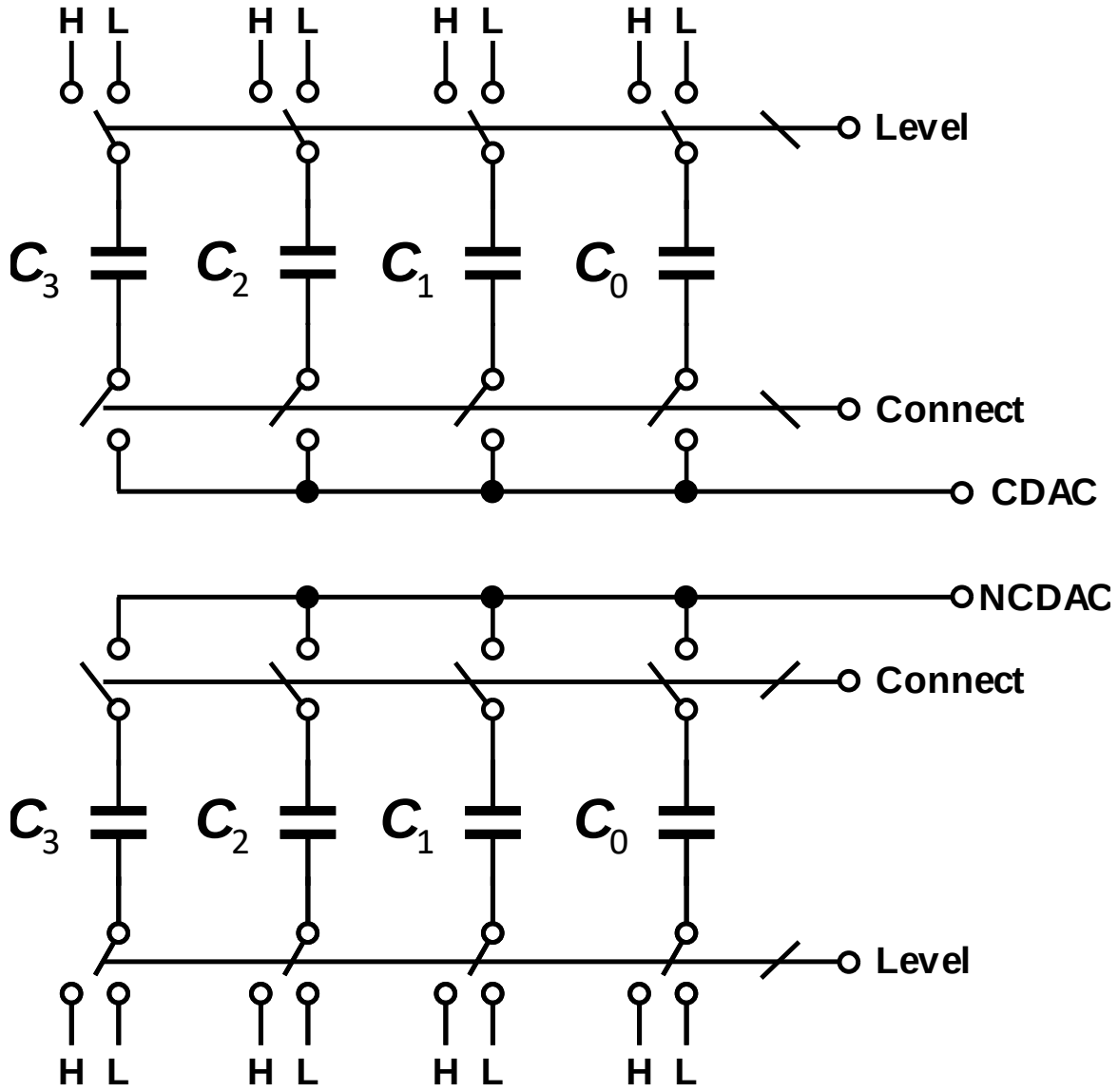
- In the conventional switching scheme of SARADC1…SARADC9, capacitor pairs toggle in opposite directions.
- In SARADC10 the monotonic switching scheme allows only ‘down’ transitions, meaning bottom plates go from $V_{High} \rightarrow V_{Low}$. At SARADC initialization, the capacitors are reset to V_{high}
- This scheme saves, on average, over 80% switching power compared to a conventional switching scheme in a regular binary-weighted CDAC.
- It reduces the disturbance on the referecen voltagens in an array of ADCs
- One disadvantage is the continuous drop of the common mode voltage.



Split capacitors

- A 4-bit binary-weighted capacitor array is connected parallel to the CDAC inputs. It has two functions:

1. Control the position of the noise minimum. [See Slides 22-24]
2. Control the sensitivity (conversion gain) of the ADC. Instead of using the full $V_{High} - V_{Low}$ range, by adding the *split* and *level* capacitors, we can choose to utilize only a fraction of it.



- The size of each of the capacitor arrays is equal to the CDAC capacitor array so that the voltage range is 0.6V ref to 1.8V V_{HIGH}-V_{LOW}

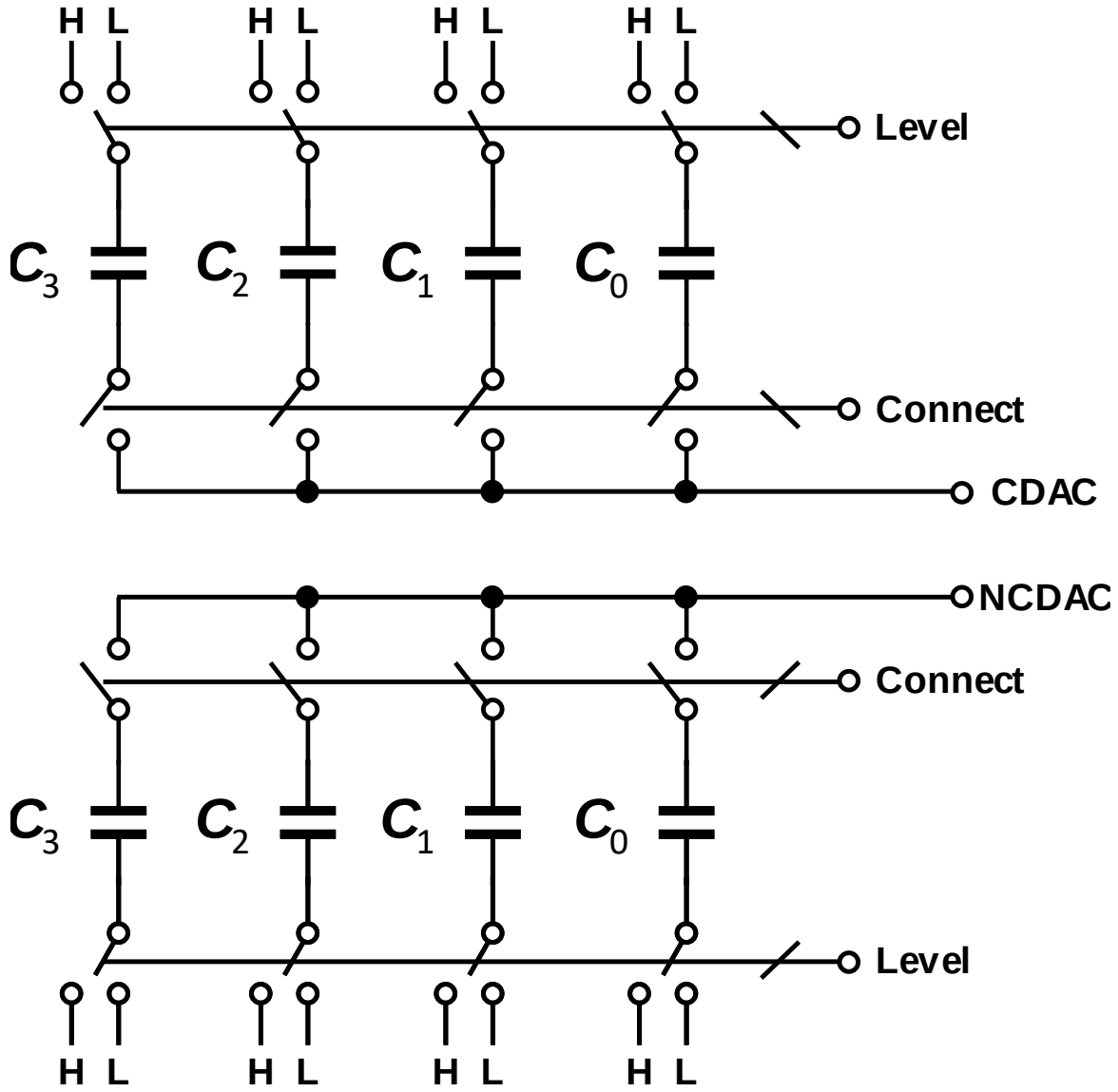
$$C_{CDAC} = C_{split} = C_{split} = \frac{1}{3} C_{total}$$

Level shift capacitors

- A second identical capacitor array is used for level shifting S versus R. Shifting the reset level prior to the MSB conversion in case of pseudo-differential signaling (Signal-Reset).

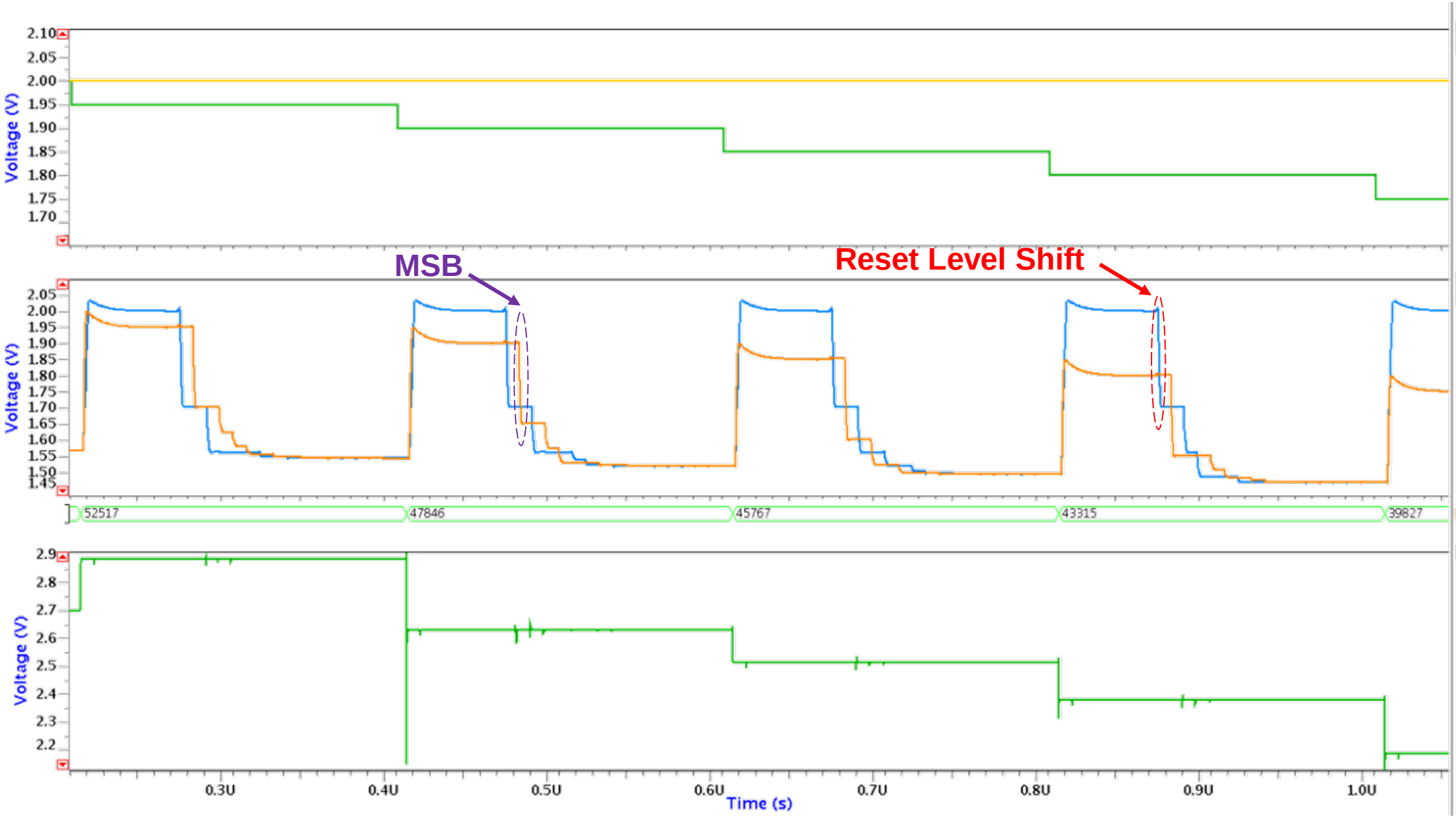
- The size of each of the capacitor arrays is conceived equal to the CDAC capacitor array.

$$C_{CDAC} = C_{split} = C_{level} = \frac{1}{3} C_{total}$$



Simulation Example@120MHz

caeleste



Input levels

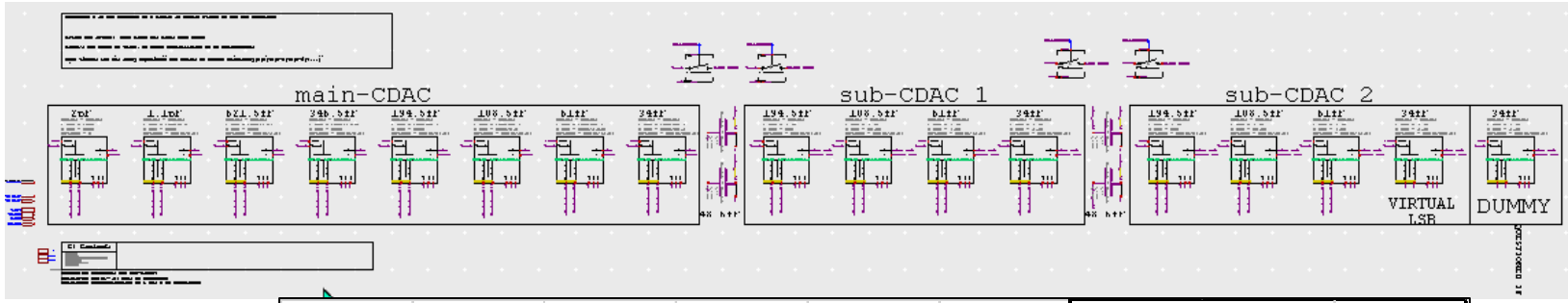
ADC Output

SPICE 16b DAC

Parametrization on the non-binary CDAC

The only parameters given to the non-binary CDAC are:

- **Cmodel**: flavor of capacitor
- **S:** strength of the switch of the MSB.
  - Lower switches are each time divided by 2 (indeed, this is imperfect, but OK)
- **Lmsb and Wmsb:** size of the MSB capacitor
  - Lower capacitors are obtained by dividing L or W by radix 1.8, see table
  - Coupling capacitor scale factor is empirically found. It is 1.42x larger than the previous capacitor (pls recheck)
  - The Capacitors behind the coupling C are copied from the 4 last capacitors before the coupling.



1,8	W	L	factor	Area	C	binary (split/level)		Area
MSB	15	40	1	600,00	1020,0	15	48	720
	15,00	22,22	1,8	333,33	566,7	15	24	360
	15,00	12,35	3,24	185,19	314,8	15	12	180
	15,00	6,86	5,832	102,88	174,9	15	6	90
	15,00	3,81	10,4976	57,16	97,2			
	8,33	3,81		31,75	54,0			
	4,63	3,81		17,64	30,0			
	2,57	3,81		9,80	16,7			
coupling	3,65	3,81	4,107042	13,92	23,7			
1,42	15,00	3,81						
	8,33	3,81						
	4,63	3,81						
	2,57	3,81						
coupling	3,65	3,81						
	15,00	3,81						
	8,33	3,81						
	4,63	3,81						
LSB	2,57	3,81						
area				1351,666		µm²		1350
total area				4051,666		µm²	Ctot/Cmbs	6,752777
C/area	1,7	fF/µm²						
Total Ctot				6887,832		fF		

Parametrization of the split/level capacitors

The parameters given to the split & level shift capacitors are:

- **Cmodel**: flavor of capacitor
- **S:** strength of the switch of the MSB.
    - Lower switches are each time divided by 2
- **Lmsb and Wmsb:** size of the MSB capacitor
    - The Split and Level C_{mbs} is 1.2x larger than the SAR (non-binary) C_{mbs}. In this way one makes that each of the SAR, Split and Level capacitor groups have the same total input node capacitance
    - Lower capacitors are obtained by dividing L by 2, 4, 8

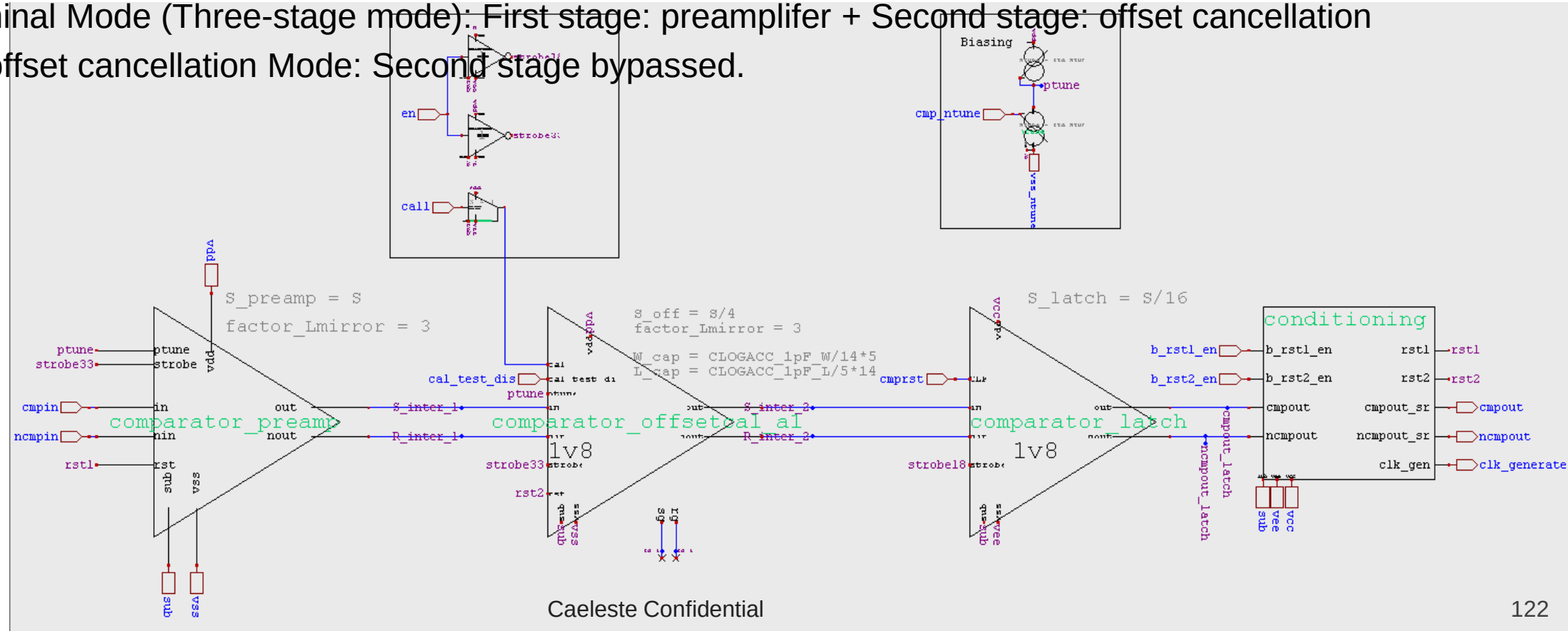
caeleste

Comparator

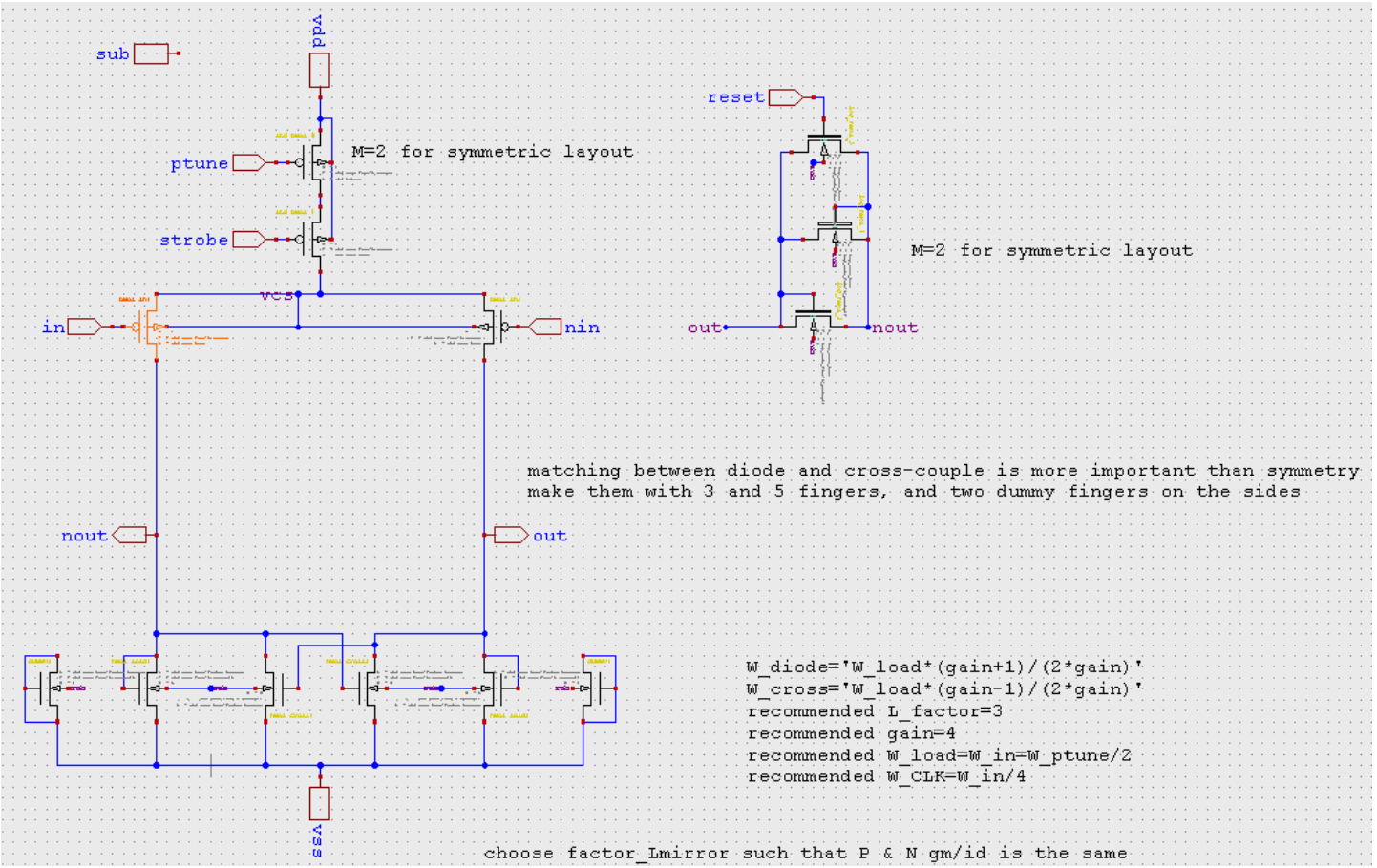
Comparator

caeleste

- The comparator is largely identical to the one of the binary SARADC. It consists of 3 stages:
    1. Preamplifier
    2. Gain stage with and offset cancellation
    3. Latch
- Two modes of operation:
    1. Nominal Mode (Three-stage mode): First stage: preamplifer + Second stage: offset cancellation
    2. No offset cancellation Mode: Second stage bypassed.



Comparator Stage 1 Preamplifier



- The preamplifier serves two purposes:
  1. Act as a low-noise gain stage.
  2. Accommodate a wide input common-mode range.

- Design Considerations:
  - Maintaining a similar bias (operating) point for the PMOS input and NMOS load, the sizes as a function of the gain can be approximated as:

$$W_{cross} = W_{diode} - \frac{W_{in}}{A}$$

where W_{cross}: width of the cross-coupled NMOS

W_{diode}: width of the diode-connected NMOS

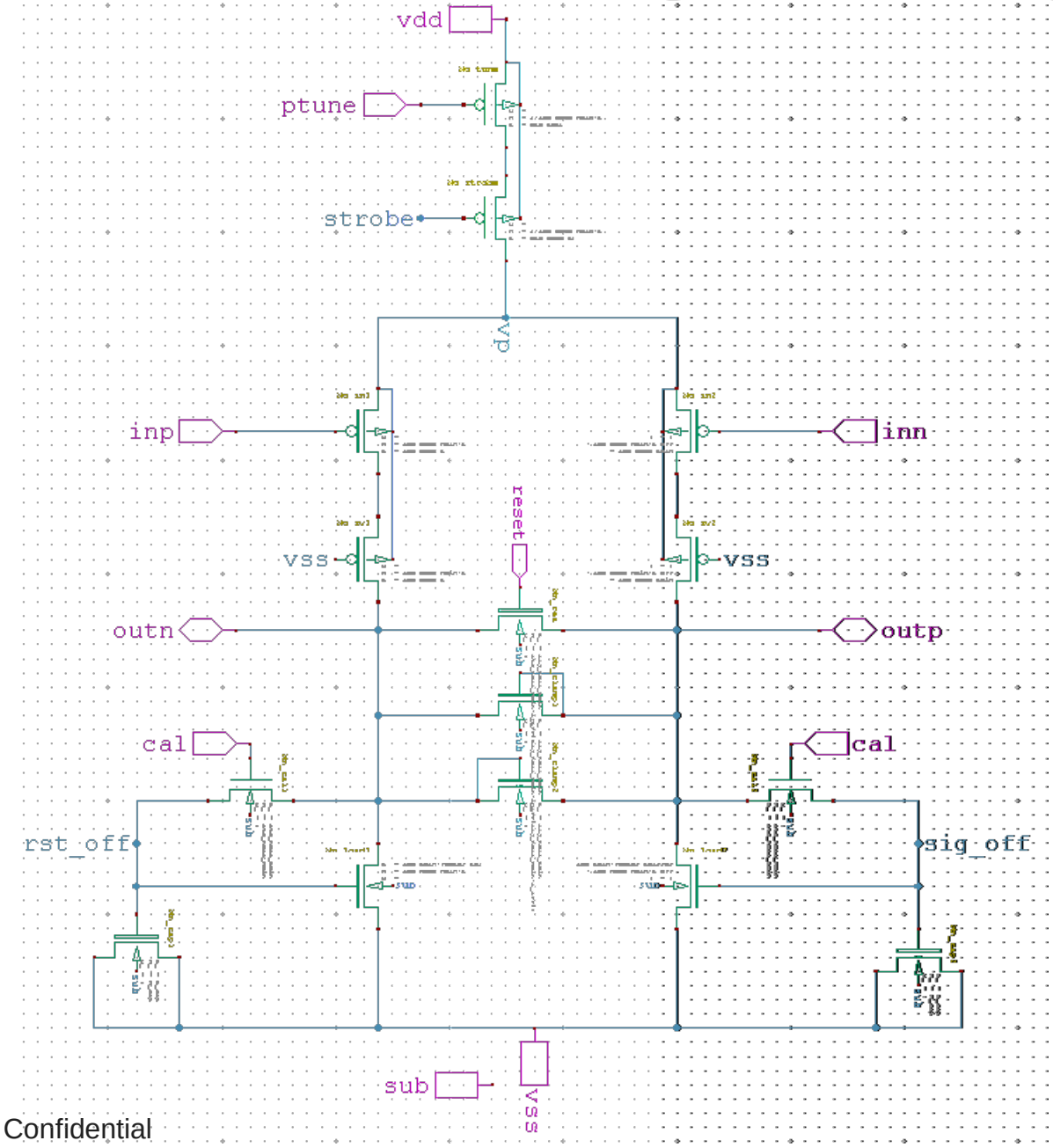
W_{in}: width of the input PMOS

  - A gain of 4 is usually used.
  - Dummy fingers (in earlier designs) removed as an accurate gain value is not a specification.

Comparator Stage 2: Offset Cancellation

- The second stage serves two purposes:
    1. Gain stage.
    2. Reduce the offset of the preamplifier as well as its own.

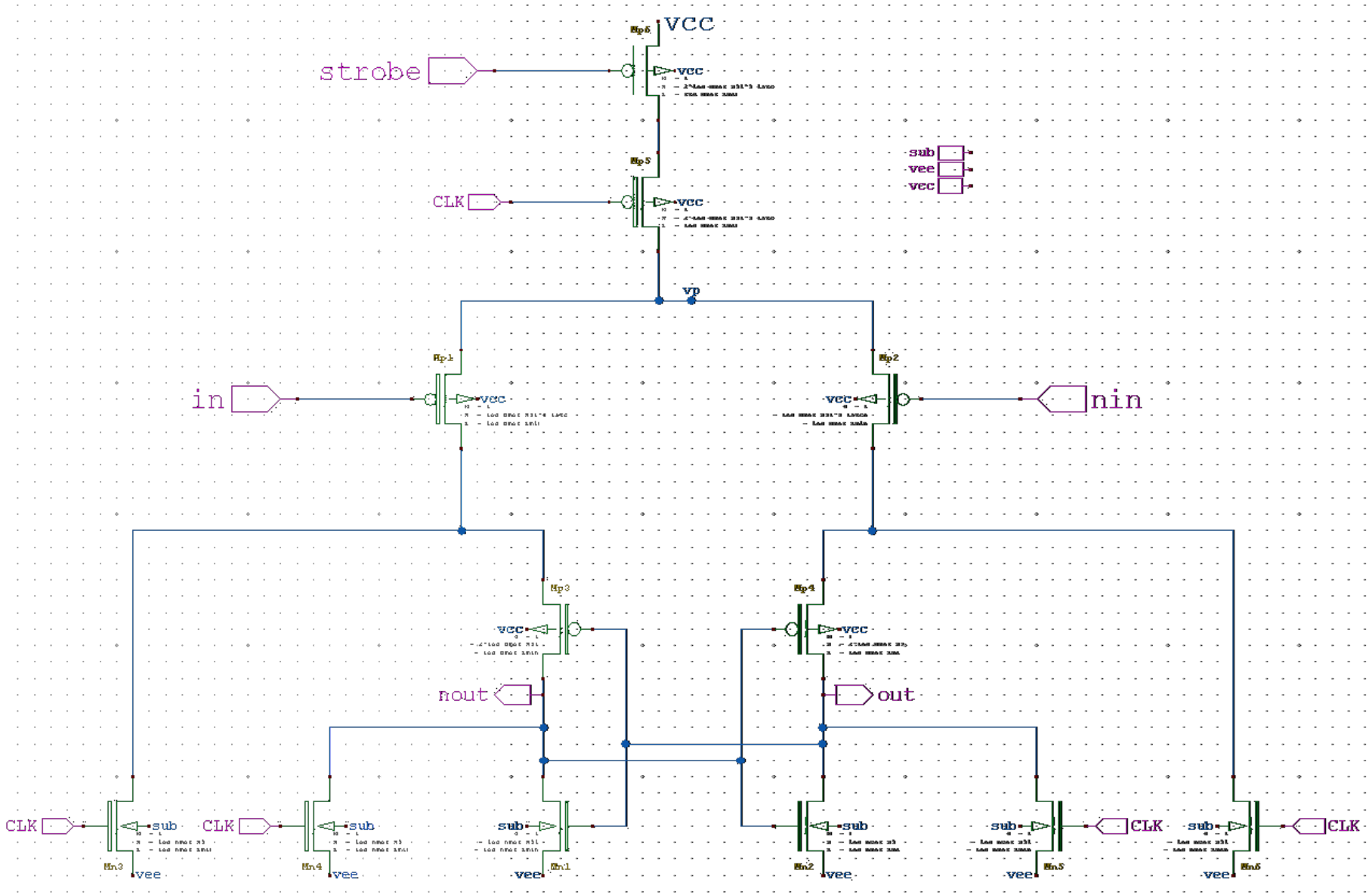
- It is possible to bypass the second stage. This is useful in case no calibration is used (you can do that by setting cal_time<1:0> to 0).



Comparator Stage 3: Latch

- The latch provides a rail-to-rail 1.8V comparator output.

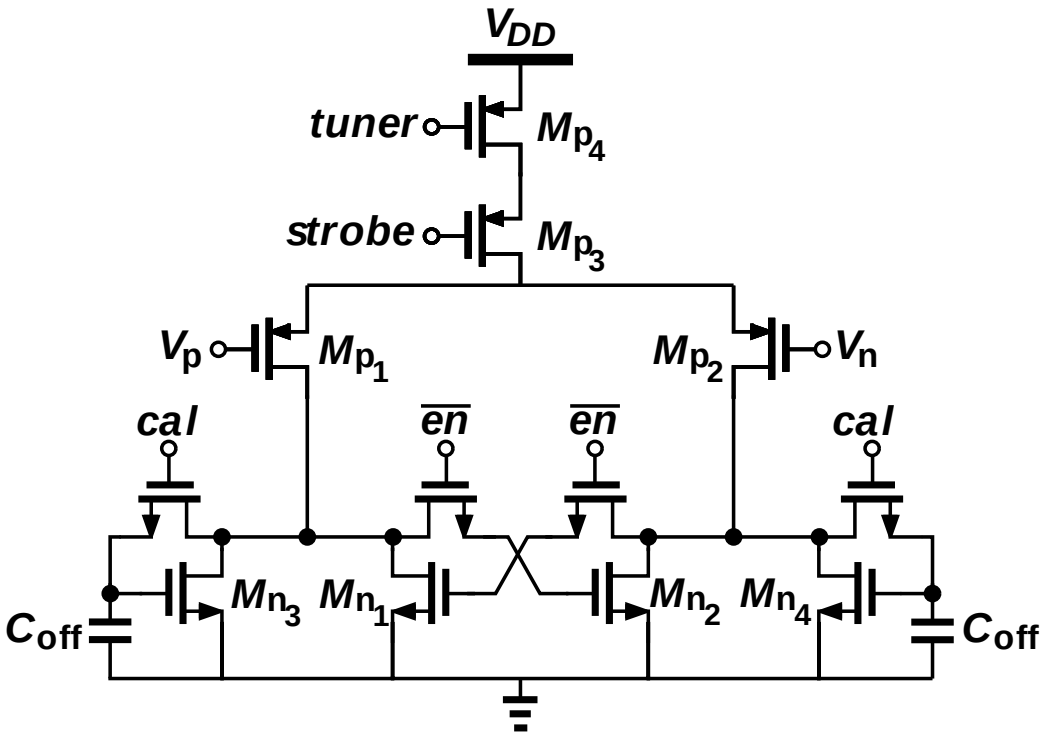
- Design Considerations:
  - Input, strobe and PMOS switch scale with the size of the latch.
  - Cross-coupled pair sizes are kept minimum.
  - Reset NMOS switches have minimum size as the comparator reset speed is inconsequential.



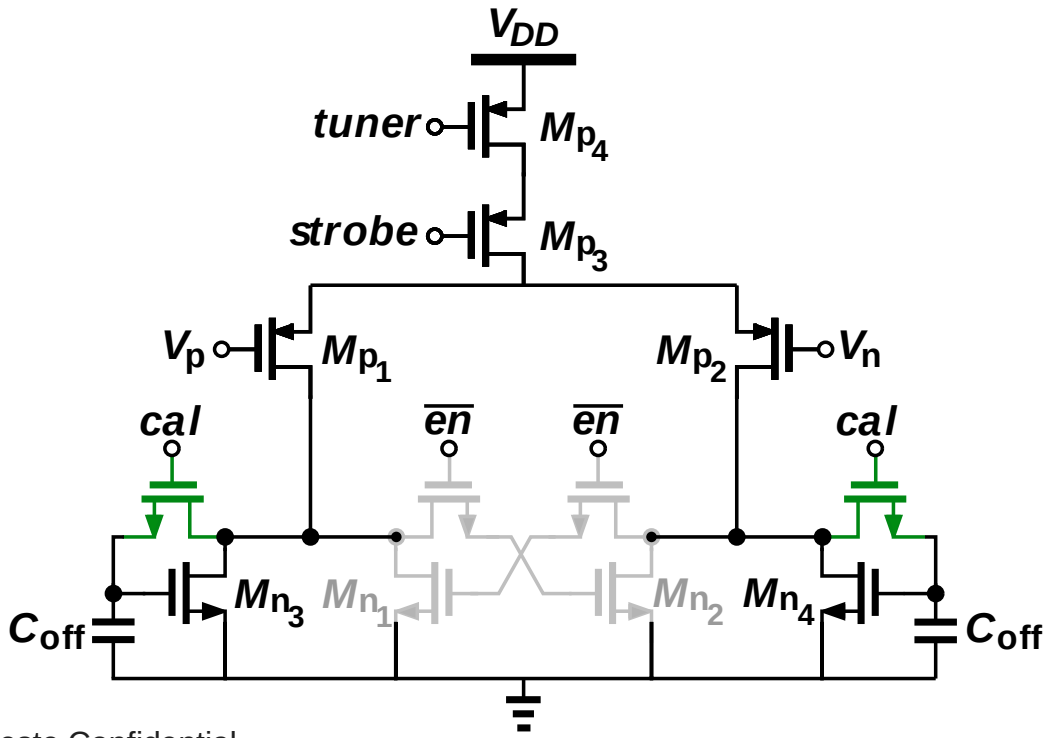
Experiment: single stage “All-in-One” Preamplifier

- The idea was to do the preamplification and offset cancellation all in one stage.
- A programmable solution was proposed as shown in the figures to maintain the old functionality as well.
- However, the addition of the calibration switch introduces a complex pole pair in the left-half plane.
- That leads to a ripple on the output in the nominal mode which can be sustained through the cross-coupled pair(positive feedback).
- **Therefore, this solution was not pursued due to a risk of oscillation.**

Nominal Mode



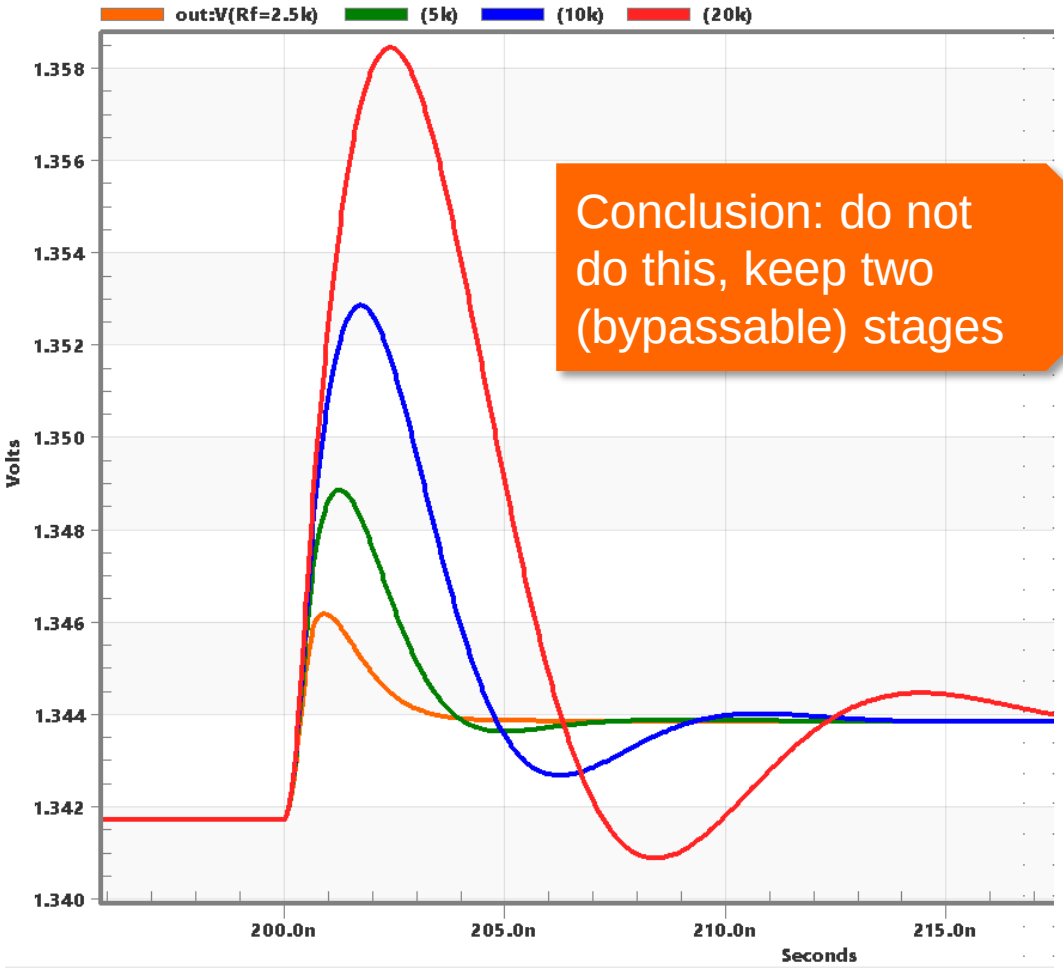
Offset Cancellation Mode



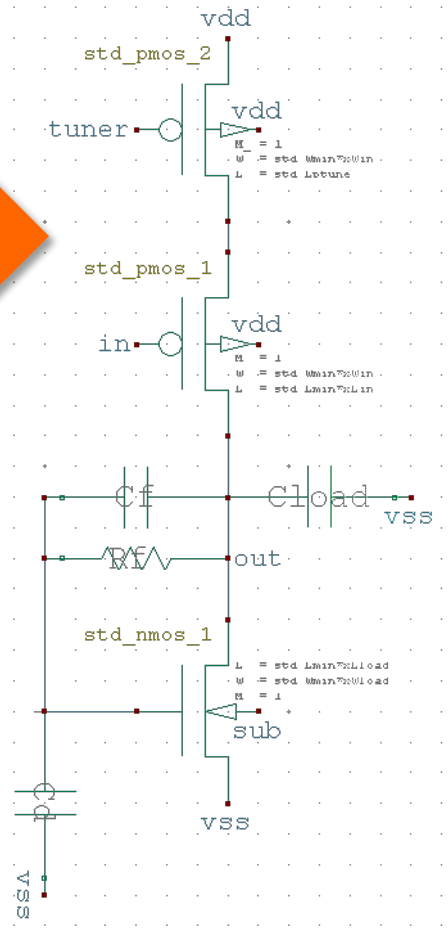
■ ON
■ Toggling
■ OFF

Experiment: single stage “All-in-One” Preamplifier (2)

- The damping factor is affected by the switch size, the load g_m and the capacitance values(parasitic, load, calibration).
- The simulations and models suggest a minimum switch size of 8 to avoid peaking at this given operating point.



Conclusion: do not do this, keep two (bypassable) stages



R	Switch S	Damping Factor (ζ)
$20k\Omega$	1	0.28
$10k\Omega$	2	0.39
$5k\Omega$	4	0.55
$2.5k\Omega$	8	0.77
0	No switch	flat

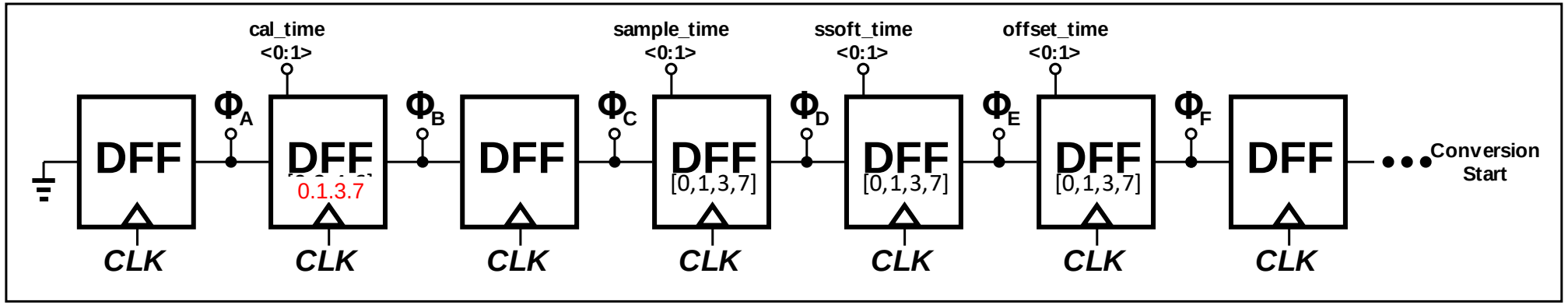
caeleste

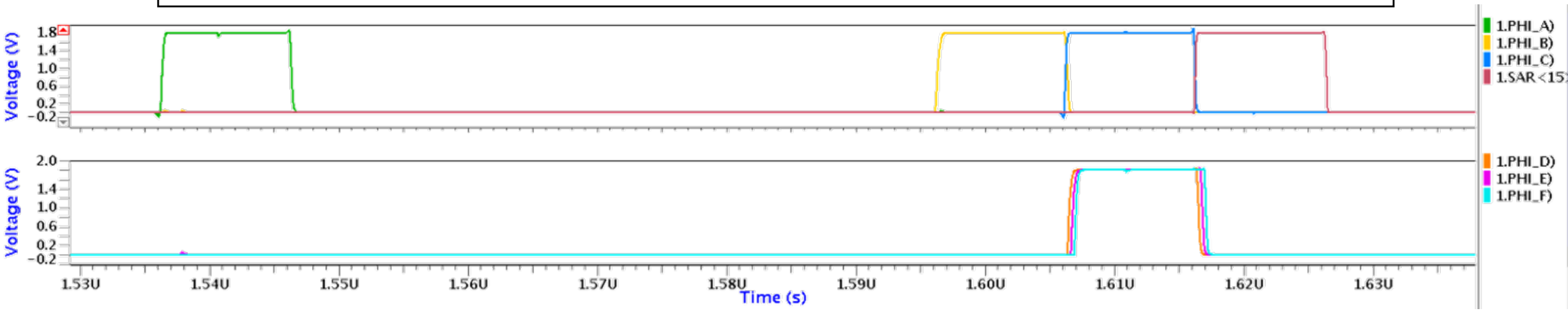
Sequencer

Sequencer concept: chain of DFFs

caeleste

- The sequencer consists of a dff9_sync0 chain that controls the timing of the ADC.
- Prior to AD conversion, several operations need to take place:
  1. Calibration 2. Sampling 3. Level shift (S versus R)
- DFFs with programmable #clocks delay are used to accommodate different conversion rates.
- The **nominal** mode uses **24** clock cycles: 6(Calibration)+1(Sampling extension)+1(level)+16(SAR onversion).





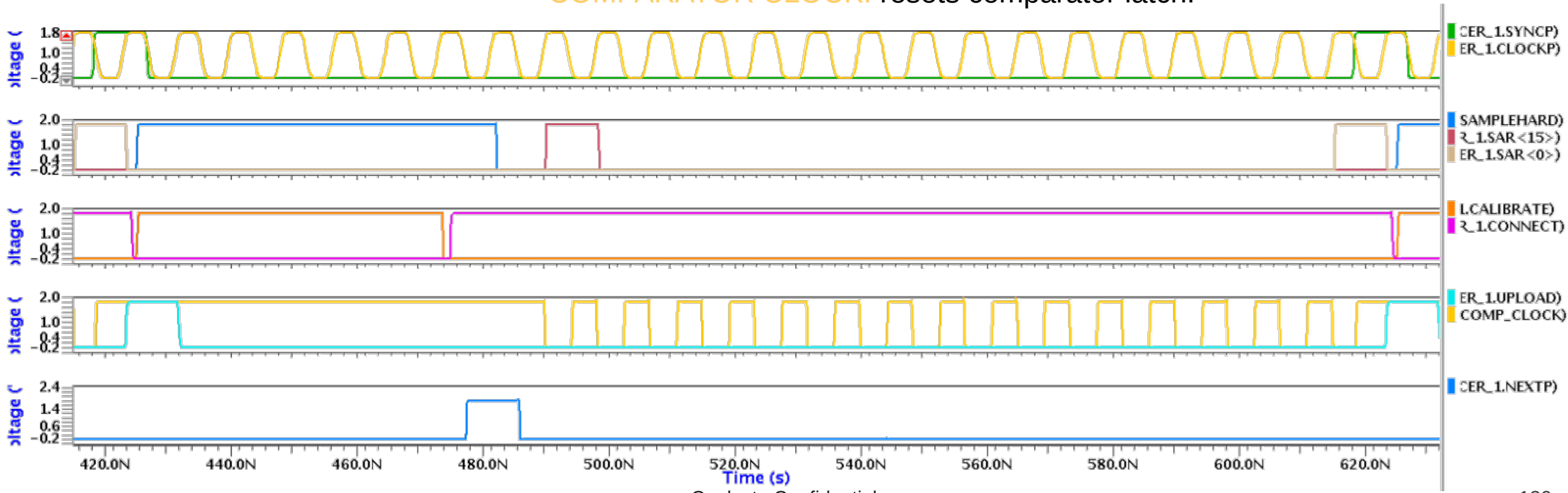
Sequencer:

Inputs:

- CLOCK: sequencer clock
- SYNC: kickstarts the sequencer defining the ADC cycle.

Outputs:

- UPLOAD: latches the output bits after conversion.
- SAR<0:15>: propagates 16 comparator decisions to the CDAC.
- SAMPLE[Hard]: samples the R&S inputs on the CDAC.
- CALIBRATE: the inputs of the comparator are shorted to the R signal and the comparator offset is recorded if offset cancellation is enabled.
- CONNECT: the CDAC is connected to the comparator after calibration.
- NEXT: SYNC towards next ADC in an interleaved configuration.
- COMPARATOR CLOCK: resets comparator latch.



Sample&hold

- Classic sample&hold
- Active sample&hold

Classic: cell SAMPLE

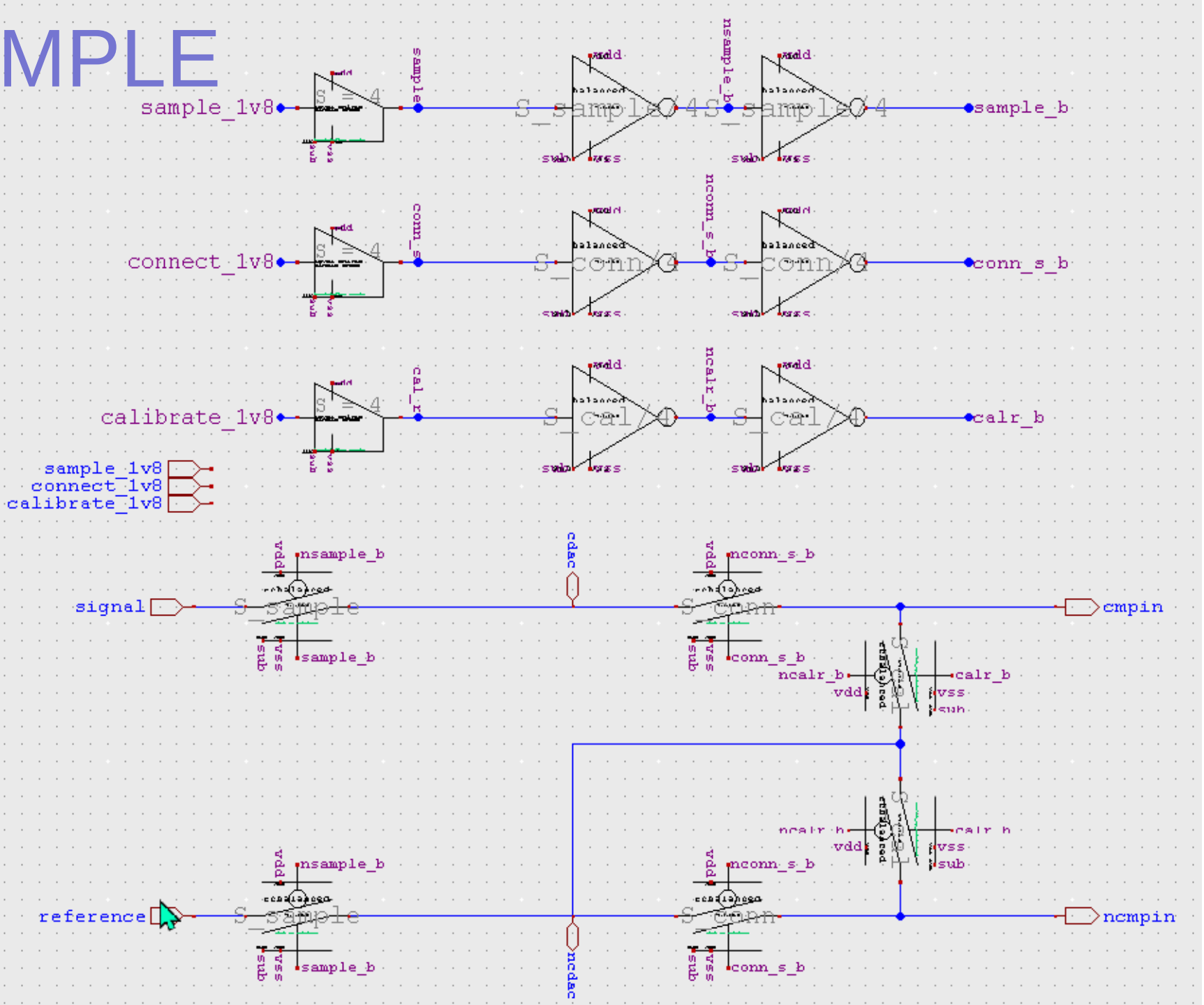
Baseline in the design is this “classic” S&H.

The cells has but one parameter S, then

S_sample = S

S_conn = S/4

S_cal = S/8



optional:
cell SAMPLE_active

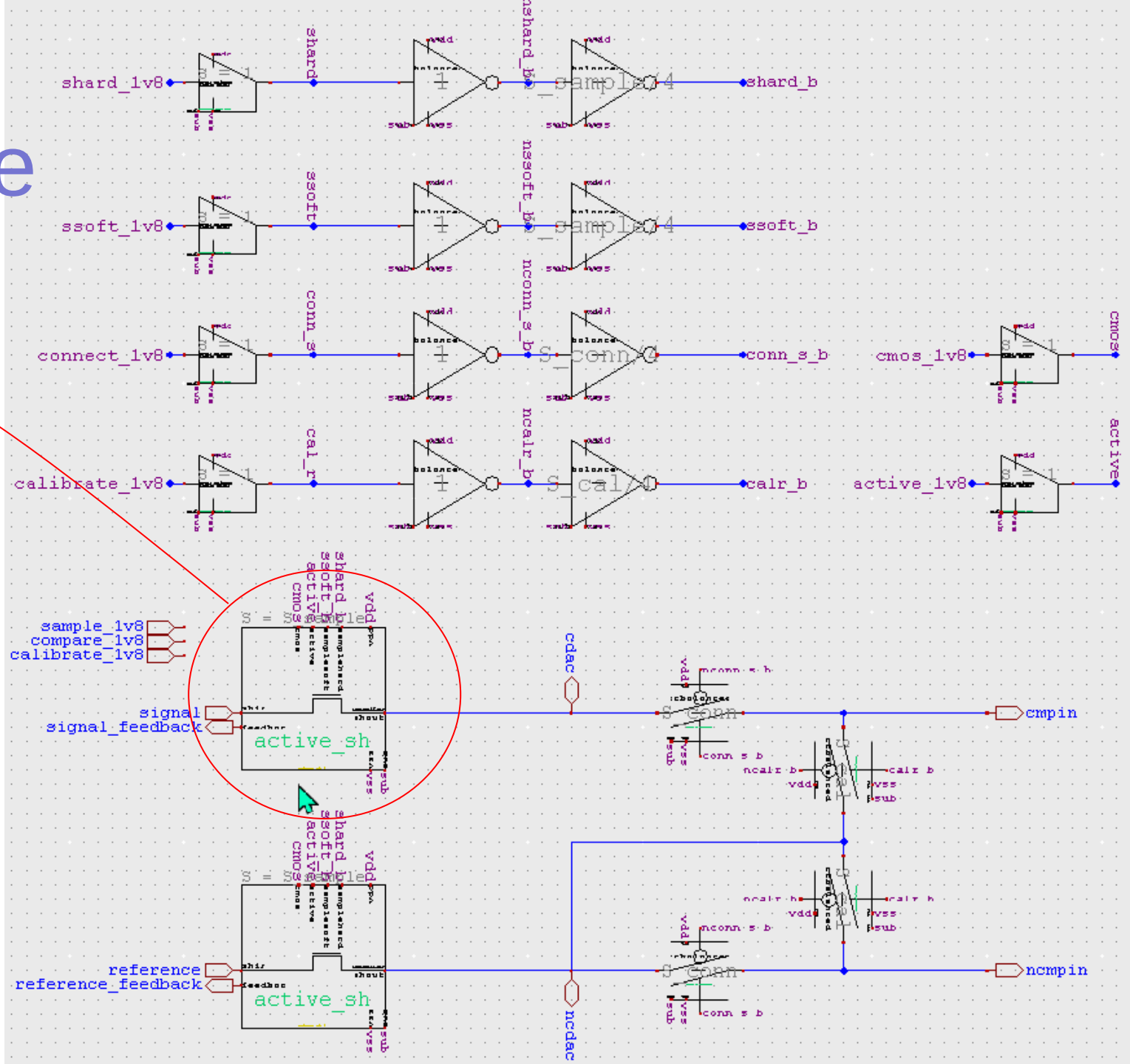
Uses active sample & hold, with cell [lib stdcells] active_sh

The concept of active S&H is explained in Notebook [0653 20230120amba noise simulations on active sample and hold.pdf]

Expected improvement in SARADC is speculative

← No 100% conclusive theorem or simulation

← In this implementation we put active S&H only on the series sampling switch. What about the other switches?



Design/Simulations/Details

Ahmed kept logs at:

- "S:\projects\scib\IPblocks\SARADC\SARADC10\Caeleste documents\20221228am SARADC10.pptx“
- "S:\projects\scib\IPblocks\SARADC\SARADC10\Caeleste documents\20221230am Everything ADC.xlsm"

caeleste

Serialization

- For an interleaved configuration
- For a parallel configuration

Serializing interleaved core ADCs

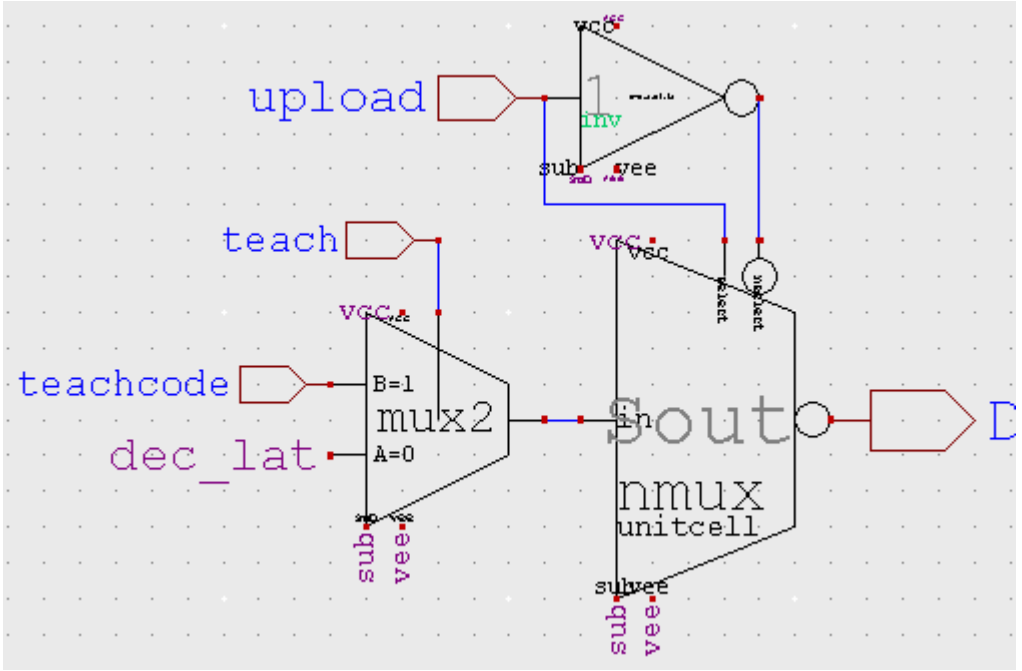
caeleste

The ADC core has a 16 bit output “D<0:15>”.

Its is brought out via a nmux_unitcell when “upload” is high.

These can be directly shunted in parallel to a bus as input to the serializer.

As this bus likely has a significant capacitance, one uses it as dynamic memory.



Serializing parallel core ADCs

caeleste

The solution of previous slide cannot be used as is for a parallel configuration.

One need an extra set of nmux_unitcells,

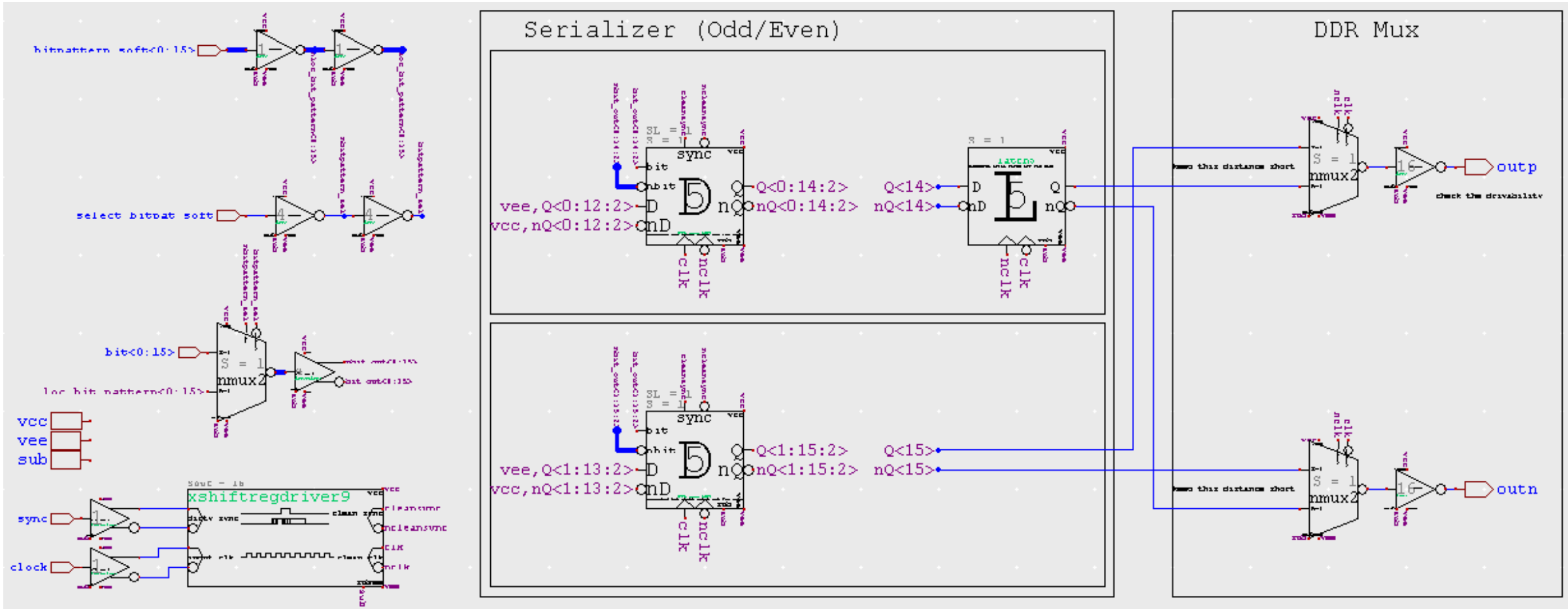
accessed by a shiftregister where

- Sync is ADC_sync
- Clock is LVDS_sync

Serializer

caeleste

Fixed topology and schematic to handle 16 bits,
identical approach as in the binary SARADC.
One can output less bits by apply less LVDS_clock/LVDS_sync



caeleste

NB2B

Non-binary to binary conversion

approaches

caeleste

A. The runtime conversion

Software, off-line

the non-binary word, bit per bit is multiplied by its weight

Hardware, on-chip

The same algorithm

the hardware realization is an array of 15 16-bit adders, conditionally adding 16 “weights”.

Firmware, in FPGA

similar as the Hardware implementation

when mature, may serve as a baseline to retarget to on-chip hardware implementation

B. The calibration

=One time per design/waferlot/wafer/device pinpointing the 16 weights

caeleste

DATASHEET

SARADC10

Contents

Electrical Specifications

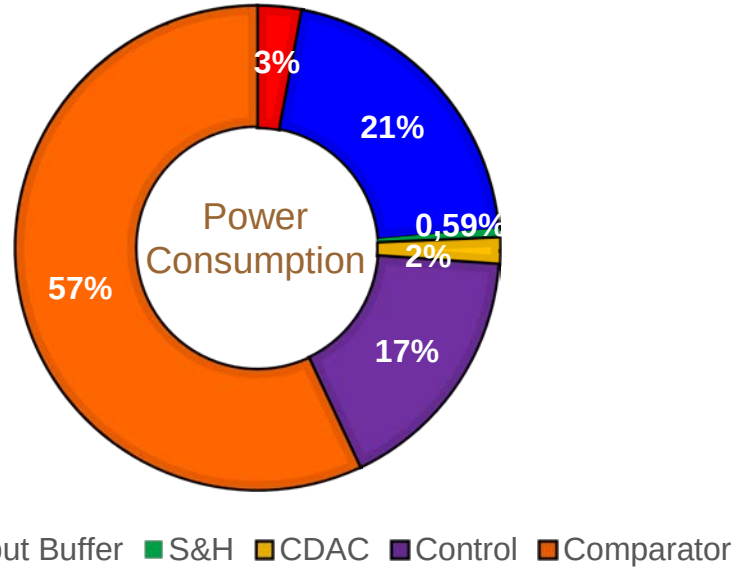
caeleste

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
$V_{CC} - V_{EE}$	Supply Voltage		1.7	1.8	1.9	V
	Supply Noise			1000		μV_{rms}
	Supply Voltage		-	0	-	V
V_{HIGH}-V_{LOW}	Voltage		1.7	1.8	1.9	V
	Noise	Signal proportional		50		μV_{rms}
	Noise	uncorrelated		100	100	μV_{rms}
	Noise	Correlated to clocks		1000		μV_{rms}
$I_{V_{CC}} \& I_{V_{EE}}$	Supply current	Average				
		Peak				
		FWHM				
V_{DD}	Supply Voltage			3.3		V
	Supply Noise			1000		μV_{rms}
V_{SS}	Supply Voltage		-	0	-	V
$I_{V_{DD}} \& I_{V_{SS}}$	Supply current	Average				

Power Consumption similation@120MHz

Block	Sub-block	Minimum	Typical	Maximum	Unit
Sequencer	**Total**		0.1		mW
Core	**Total**		3.28		mW
	Input Buffer		0.7		mW
	Sample and Hold		0.02		mW
	CDAC		0.06		mW
	Control Logic		0.57		mW
	Comparator		1.92		mW

- Input buffer tuner=17(linear range: $9 \rightarrow 17$).
- Comparator tuner=19(linear range: $13 \rightarrow 22$).
- Average over full conversion period?
- A single core and a single sequencer?



Timing Specifications

caeleste

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit

Example Timing Setups

caeleste

- The “next” signal is the “sync” of the subsequent ADC core in an interleaved architecture.
- The case of only 1 clock cycle delay for the “next” ADC core is not implemented as it is only relevant for fitting 16 interleaved ADCs into 16 clock cycles. In such case one better fits 8 pairs of ADCs.
- Sampling starts with calibration. The shortest possible sample time is one clock cycle on top of the calibration time.
- we assume that the #bits readout must be a divider of the ADC conversion duration.

No. of Interleaved ADCs	Next Delay #clks (code)	ADC conversion duration #clocks	Calibration #clks (cal_time code)	Sample and Hold (Extra) #clks (sample_time code)	Active Sample and Hold #clks (ssoft_time code)	Level shift #clks (level_time code)	#of SAR steps	#Bits read out over serializer
8	2 (00)	16	1 (01)	1 (00)	0 (00)	1 (00)	13	
8	2 (00)	16	0 (00)	1 (00)	0 (00)	1 (00)	14	
8	**3 (01)**	**24**	**7 (11)**	**1 (00)**	**0** (00)	**1 (00)**	**15**	**12**
8	4 (10)	32	7 (11)	4 (10)	0 (00)	4 (00)	16 (+1 void)	16
12	3 (01)	36	7 (11)	8 (10)	0 (00)	4 (00)	16 (+1)	12
8	5 (11)	40	7 (11)	4 (10)	7 (11)	1 (00)	16 (+3)	16
4	2 (00)	8	0 (00)	1 (00)	0 (00)	1 (00)	7	8

1.8V ASPI Registers

caeleste

ASPI register: Sequencer timing

Bit	Name	Default Value	Function
9:8	next_delay<1:0>	0	Control the delay of the “next” sync signal. 00: 2 clock cycles 01: 3 clock cycles 10: 4 clock cycles 11: 5 clock cycles
7:6	level_time<1:0>	0	Control the duration of the level shift operation. 00: 0 cycles 01: 1 cycles 10: 3 cycles 11: 7 cycles 20230417baab name change
5:4	ssoft_time<1:0>	0	Control the no. of active sampling clock cycles. 00: 0 cycles 01: 1 cycles 10: 3 cycles 11: 7 cycles
3:2	sample_time<1:0>	0	Control the no. of *extra* sampling clock cycles. 00: 1 cycles 01: 2 cycles 10: 4 cycles 11: 8 cycles
1:0	cal_time<1:0>	3	Control the no. of calibration clock cycles. 00: 0 clock cycles (no calibration) 01: 1 clock cycles 10: 3 clock cycles 11: 7 clock cycles

1.8V ASPI Registers (Cont'd)

caeleste

ASPI register: **Serializer and bit patterns**

Bit	Name	Default Value	Function
			~~Controls bits 8-10 in the ADC teach code.~~ Removed, these are hardcoded
17	teach	0	Apply the hardwired teach code to the output latch of the ADC.
16:1	bitpattern<15:0>	0…0	Controls the test bit pattern for the serializer.
0	select_bitpattern	0	Apply the test bit pattern to the input of the serializer.

1.8V ASPI Registers (Cont'd)

caeleste

ASPI register: **Split capacitors**

Bit	Name	Default Value	Function
11:8	split_ncdac<3:0>	0000	Select the voltage on the bottom plate of the split capacitors connected to the negative CDAC node. 1111: VHIGH, VHIGH, VHIGH, VHIGH 1000: $VHIGH_{MSB}$, VLOW, VLOW, VLOW … 0000: VLOW, VLOW, VLOW, VLOW
7:4	split_cdac<3:0>	0000	Select the voltage on the bottom plate of the split capacitors connected to the positive CDAC node. 1: VHIGH 0: VLOW
3:0	split_conn<3:0>	1111	Connect the top plates of the split capacitor array to the CDAC nodes.

1.8V ASPI Registers (Cont'd)

caeleste

ASPI register: **level shift capacitors**

Bit	Name	Default Value	Function
9	level_node	0	Select the CDAC node to apply the offset to: 0: NCDAC 1: CDAC
8	level_direction	0	Select the offset polarity. 0: negative offset 1: positive offset
7:4	level_bit<3:0>	1000	Control the value of the offset from VLOW to VHIGH with a 4-bit resolution.
3:0	level_conn<3:0>	1110	Connect the top plates of the offset capacitor array to the CDAC nodes.

Note 20230417baab change name from "offset" to "level"

3.3V ASPI Registers

caeleste

ASPI register: **Input buffer and comparator**

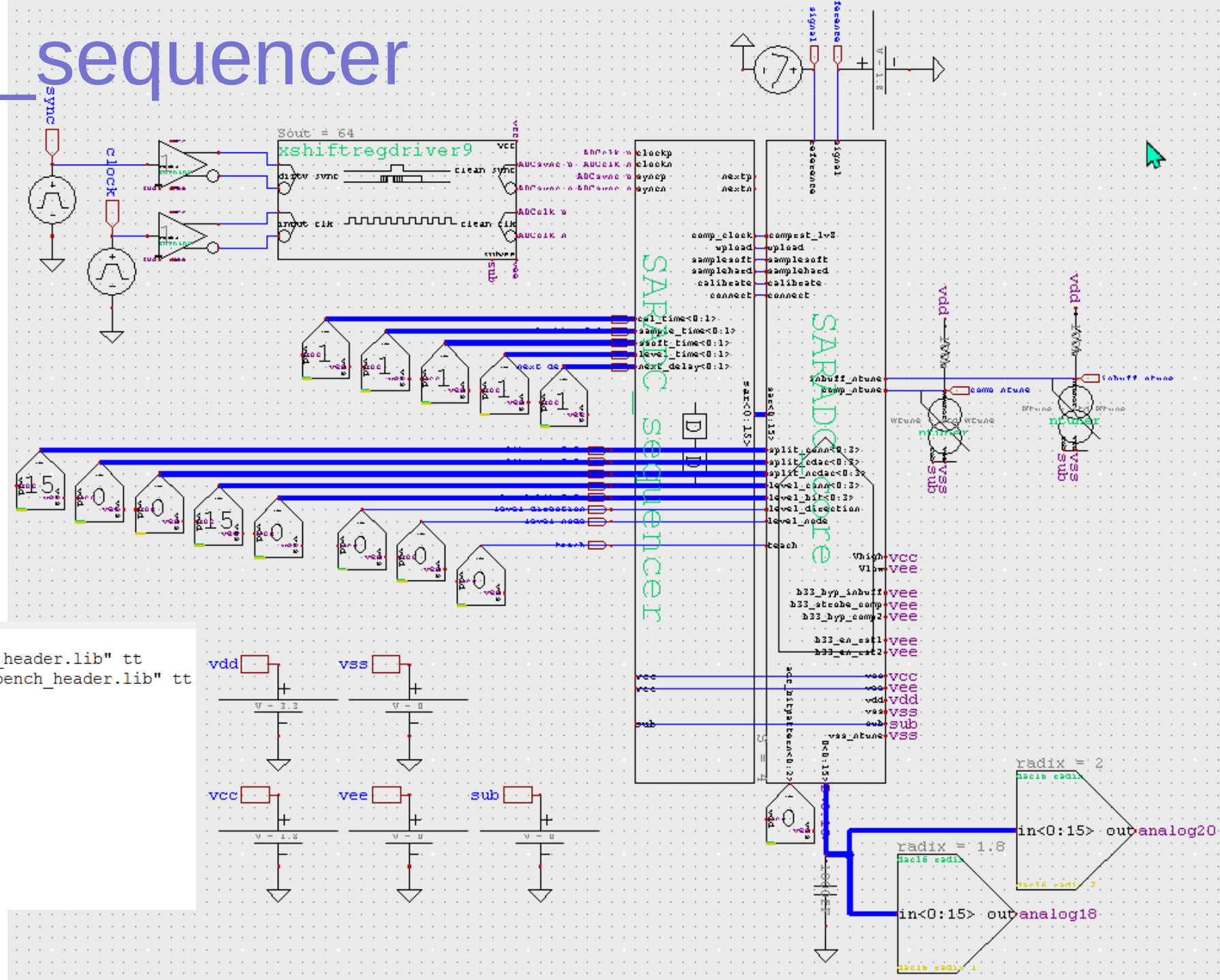
Bit	Name	Default Value	Function
4	b33_byp_comp2	0	Bypass the comparator stage 2. (1=yes, 0=no)
3	b33_en_rst2	1	Enable the reset signal of the comparator stage 2.
2	b33_en_rst1	1	Enable the reset signal of the comparator stage 1.
1	b33_strobe_comp	0	1=Strobe the ADC comparator: All comparator stages are powered off
0	b33_byp_inbuff	0	Bypass the input buffer of the ADC.

caeleste

simulations

SB_core_plus_sequencer

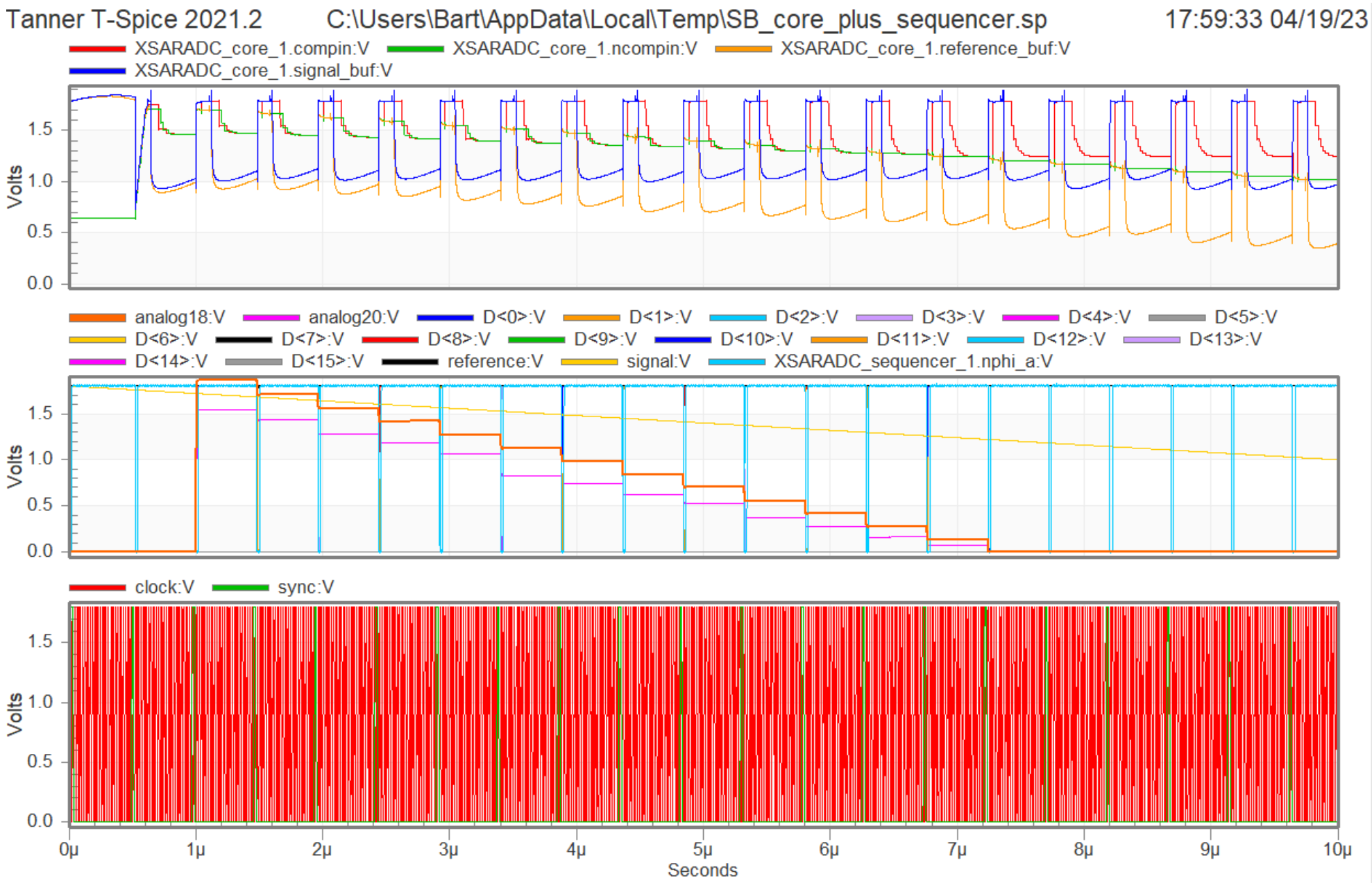
SB_core_plus_sequencer




```
*
.lib "S:/technologies/tsl018/v2.0/default_testbench_header.lib" tt
*.lib "S:/technologies/xfab_xc018/v2.0/default_testbench_header.lib" tt

.options method=BDF
.options numnt=100
*.options abstol=1e-15
*.options reltol=0.005
.options lvltim=3

.param period=20ns

.print tran
.probe tran
```

Transient

caeleste



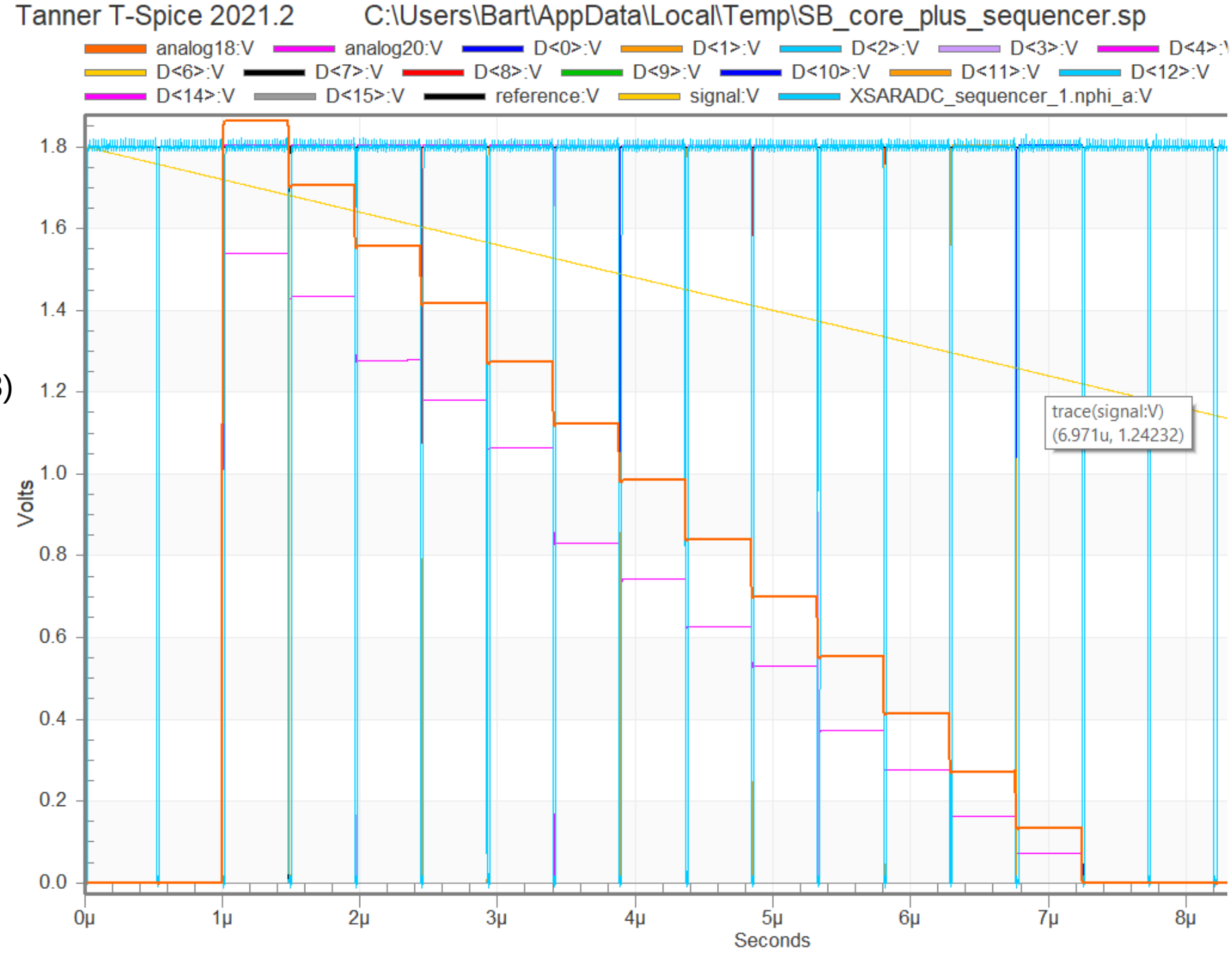
Zoom in

Reference: 1.8V

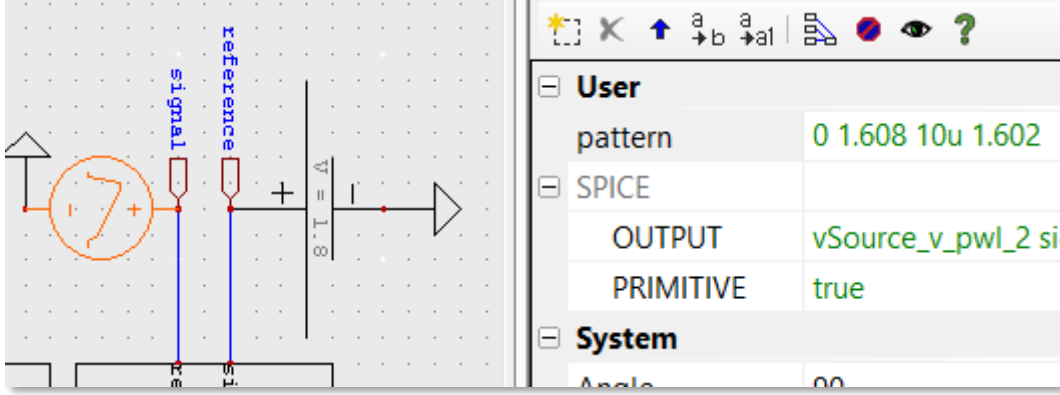
Signal: 1.8->1.2V

Radix 1.8 DAC (analog1.8) looks ~nice linear

“normal binary radix 2” (analog20) shows uneven steps.



Sweep with 1/100 range

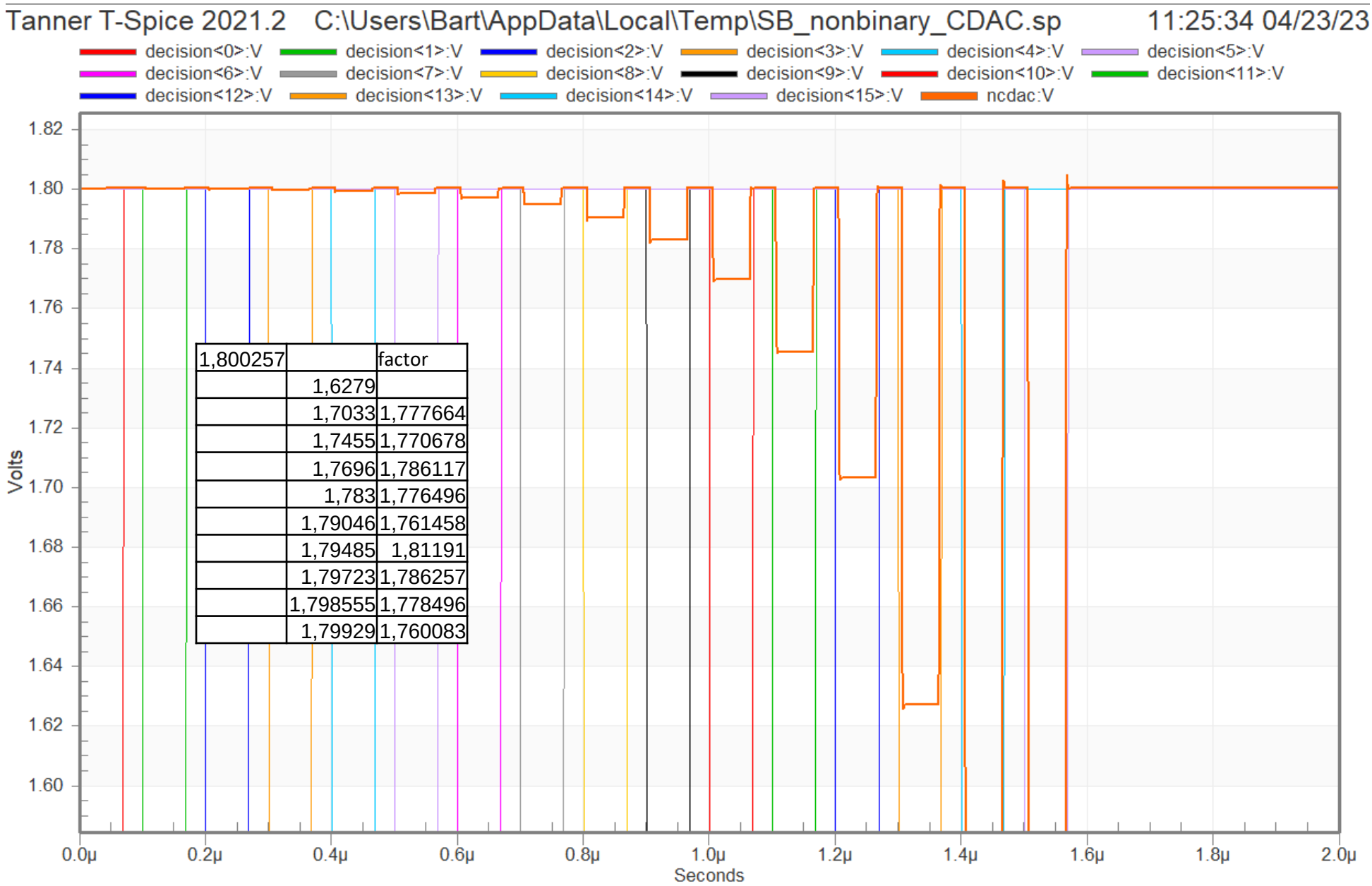


SB_nonbinary_CDAC

caeleste

Simulation bench showing each successive delta V.

Allows to quickly check the radix factors, targeted to be ~1.8

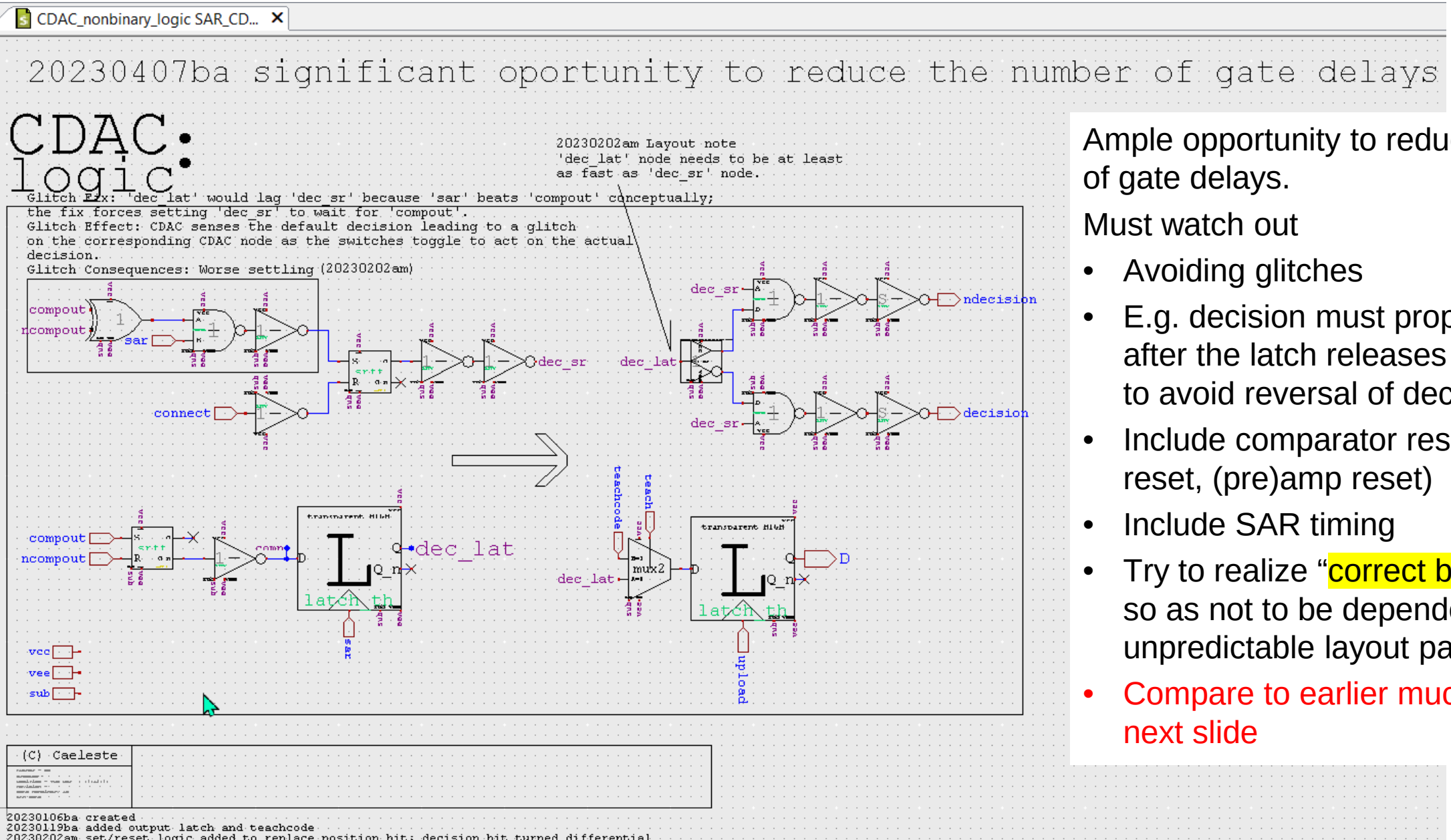


1,800257		factor
	1,6279	
	1,7033	1,777664
	1,7455	1,770678
	1,7696	1,786117
	1,783	1,776496
	1,79046	1,761458
	1,79485	1,81191
	1,79723	1,786257
	1,798555	1,778496
	1,79929	1,760083

caeleste

Non-binary: future improvements

Improve cell CDAC_control_unit_nonbinary in library SAR_CDAC



Ample opportunity to reduce the number of gate delays.

Must watch out

- Avoiding glitches
- E.g. decision must propagate to CDAC after the latch releases its reset, so as to avoid reversal of decision
- Include comparator reset timing (latch reset, (pre)amp reset)
- Include SAR timing
- Try to realize "correct by construction", so as not to be dependent on unpredictable layout parasitics.
- Compare to earlier much simpler cell next slide

Earlier version: _obsolete20230202_CDAC_control_unit

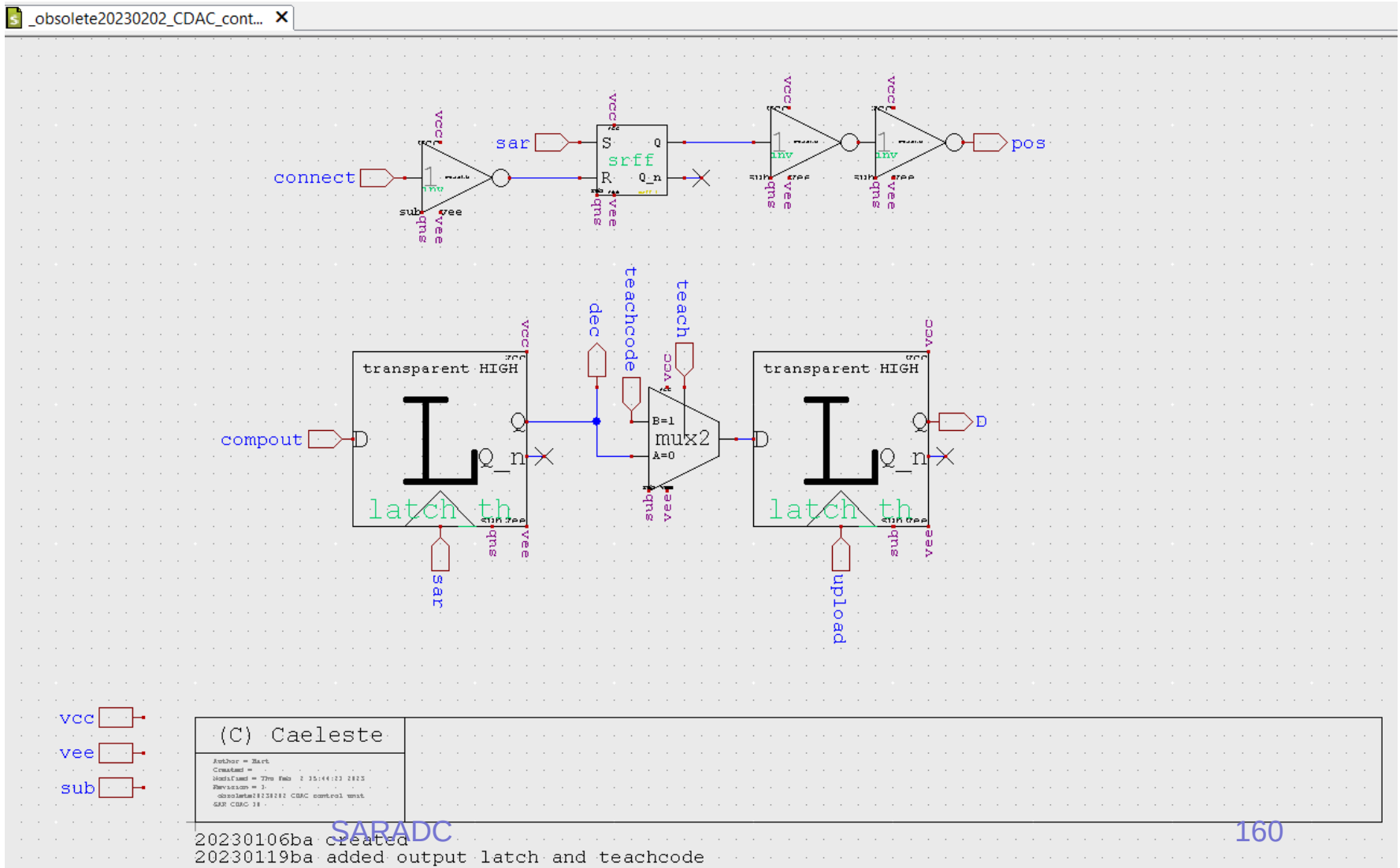
This is an earlier generation of +/- the same.

It worked (?)

Has much less gate delays

but has glitches (?).

Can one devise a compact elegant control unit not having the glitches?



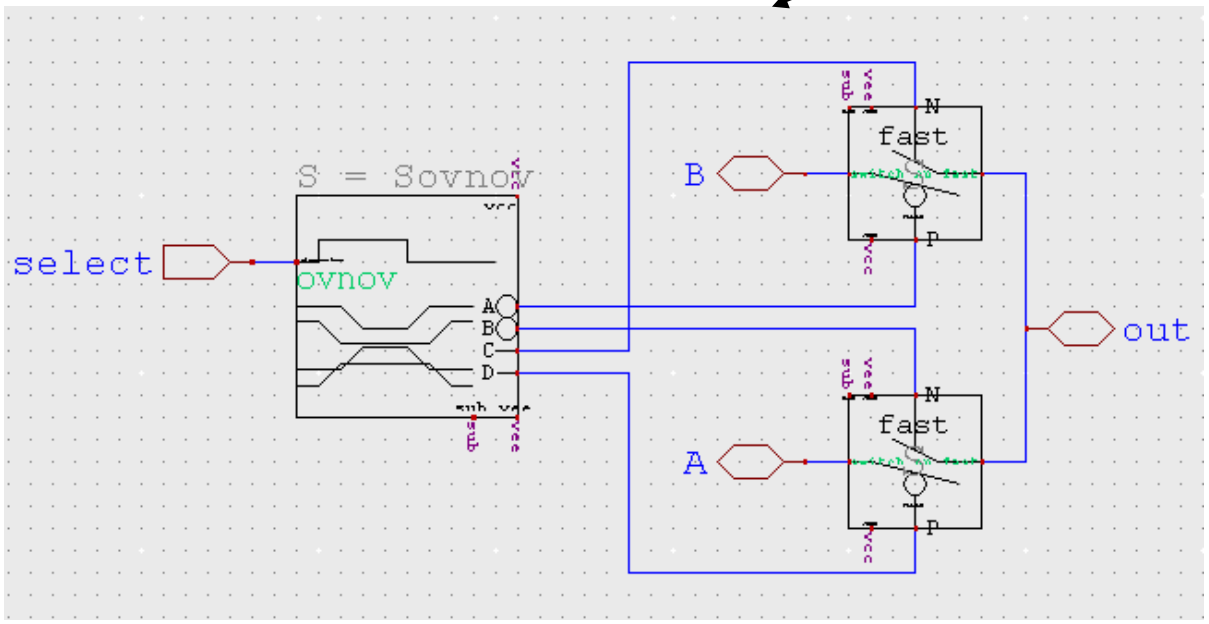
Simplify BBM switches

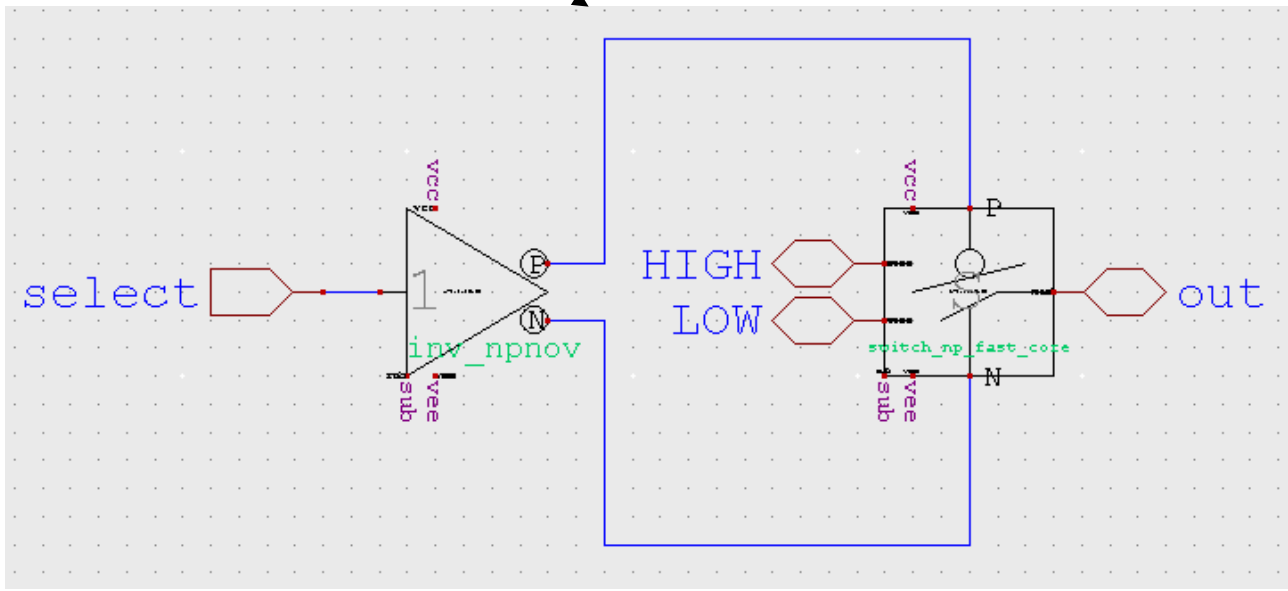
caeleste

The CDAC uses BBM switches2. They are fully complementary and symmetric on both inputs.

However, when switching between VHIGH=1.8V and VLOW=0V, a single PMOSFET and NMOSFET suffice.

One could eventually replace all bottom switch2bbm_fast by switch2bbm_highlow_fast





Split/level capacitor switches

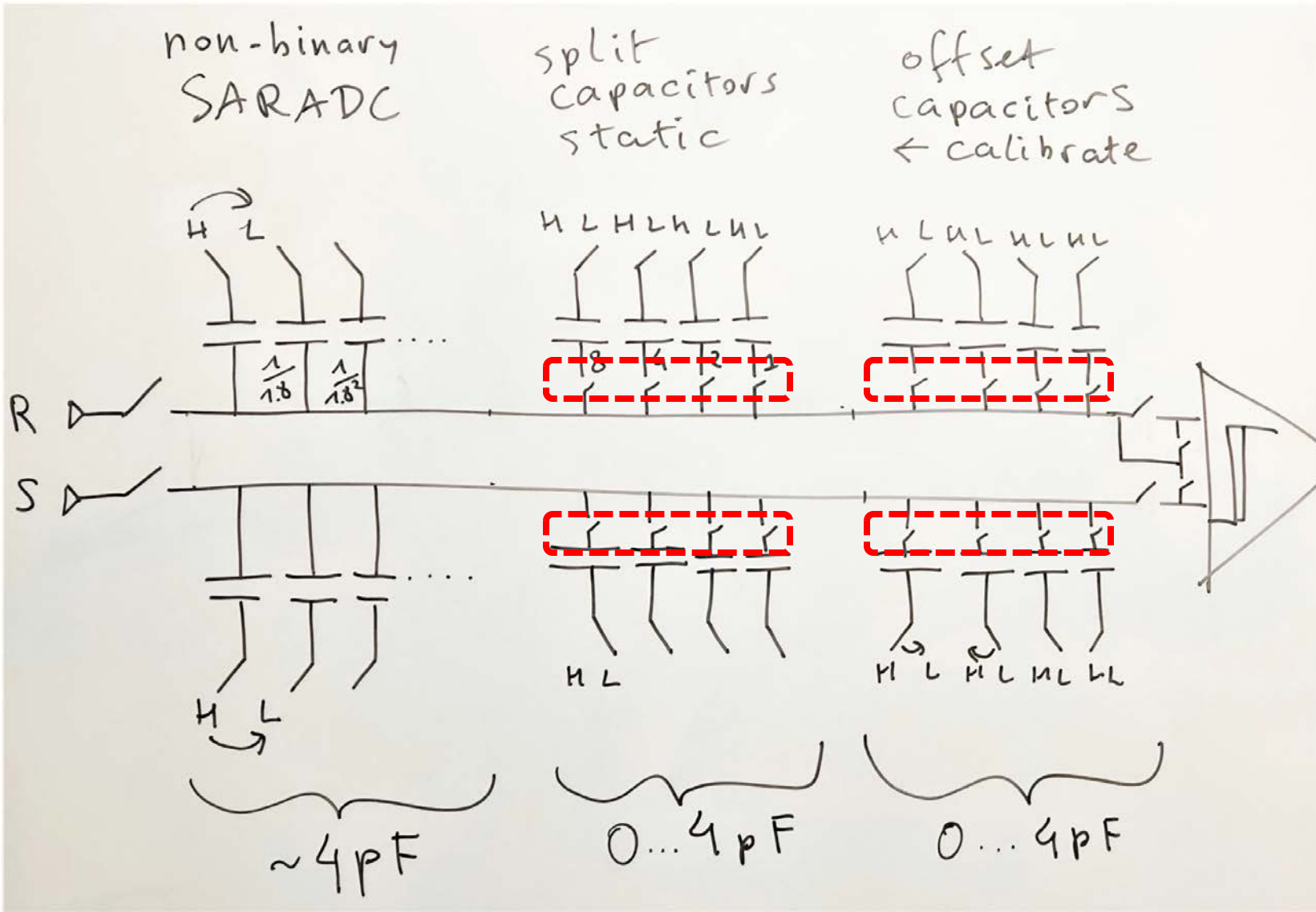
caeleste

The switches on this position are presently 1.8V fast switches.

However, the R and S may be higher than 1.8V.

Solutions?

- Make these 3.3V switches (baseline). Whatch out to maintain short RC timeconstant.
- Put these switches at the bottom. Possible by logically combining them in the bottom LH switches. (not baseline)
- …



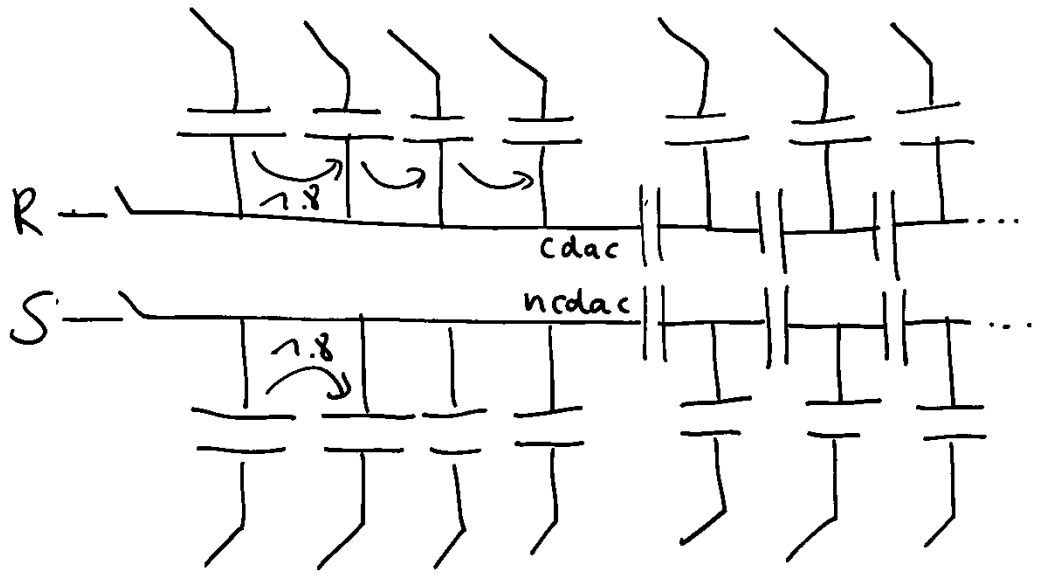
CDAC coupling cap: do this for every bit.

Advantages: somewhat simpler design

Identical unit cell for 8 LSB

Same radix for all 8 LSB

Avoids non-linearity of the subranges.



Experiment fails: this C2C arrangement cannot result in a radix 1.8. Only radix > 2.

Alternative: per group of 2?

Non-binary CDAC approximating radix with #unit caps

As in Notebook 0677

What if the non-binary CDAC is composed of matched unit capacitors?

This facilitates or freezes the adder coefficients.

Examples of how the eight MSBs might be assembled:

The only remaining variable is the size of the coupling capacitor, which must remain finely programmable on its own (as in TIMA paper).

Greatly simplifies parameter upload to 16xADDER. Adder itself becomes smaller. Yet makes the CDAC(proper) much bigger

what if the non-binary CDAC is composed of matched u				
this facilitates or freezes the adder coefficients				
	#units	ratio	#units	ratio
MSB	94		170	
|	52	1,81	94	1,81
|	29	1,79	52	1,81
|	16	1,81	29	1,79
|	9	1,78	16	1,81
|	5	1,80	9	1,78
|	3	1,67	5	1,80
MSB8	2	1,50	3	1,67
total#units	210		378	
coupling	?		?	
MBS9	9		16	
|	5		9	
|	3		5	
MSB12	2		3	
coupling	?		?	
MSB13	9		16	
|	5		9	
|	3		5	
LSB	2		3	

NB2B with integer weights

From Notebook 0645

94
52
29
16
9
5
3
2

The weights can be encoded in a 8-bit word, adding to a maximum value of 210. One needs 7 [8-bit or less] adders.

The weights of the lower bits add up to less than the above “2”, again can be realized in 8 (7/8/9) bit adders. These coefficient are variable, yet have a large systematicity.

For comparison Notebook 0645 brute force approach→

Possible implementation of on-chip NB2B processor

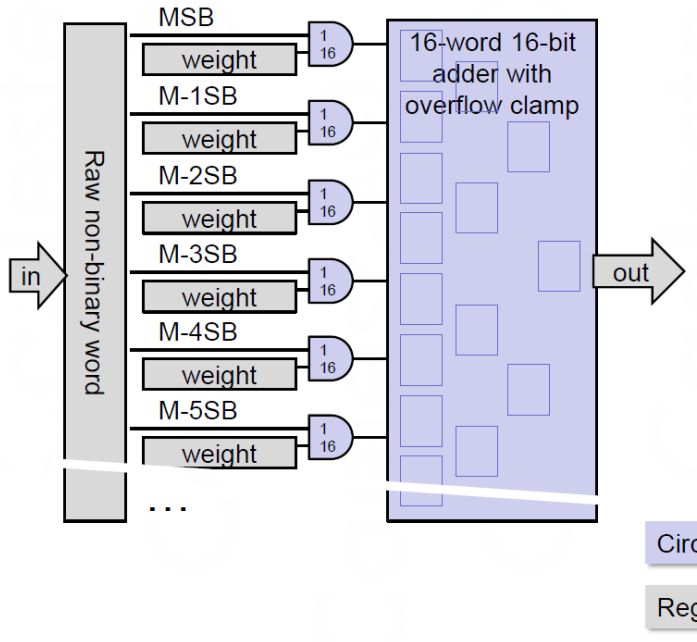
Foreach each bit of the non-binary word, when 1 add a “weight” (from ASPI, static).

One should foresee sub-LSB bits to maintain resolution and noise.

Each adder is similar to the adders presently used in the SARADC logic, which has an execution time equal to the ADC clock.

The required execution time is the LVDS sync frequency which is (presently) 3x slower than the ADC clock.

A 16-layer adder can be made as a 8+4+2+1 tree, with minimal extra delay.



20 December 2022 confidential 15

Memory effect

caeleste

It was suggested that settling of the inputs stage causes memory effect 20230713 layout review of SARADC10

A way to cancel his might be to briefly shunt / preset / precharge the capacitor nodes.

Needs confirmation.

Embed upload<> generation in each ADC core.

Presently (SARAD10) the circuit xscanner_parallel8 created the control to enable the output of data of each ADC core in sequence.

We propose to remove xscanner_parallel8 and embed short pieces of DFF9 inside each ADC core. This ressembles the earlier implementation of “sync/next” circuitry in the sequencer to be used in interleaving mode.

Advantage:

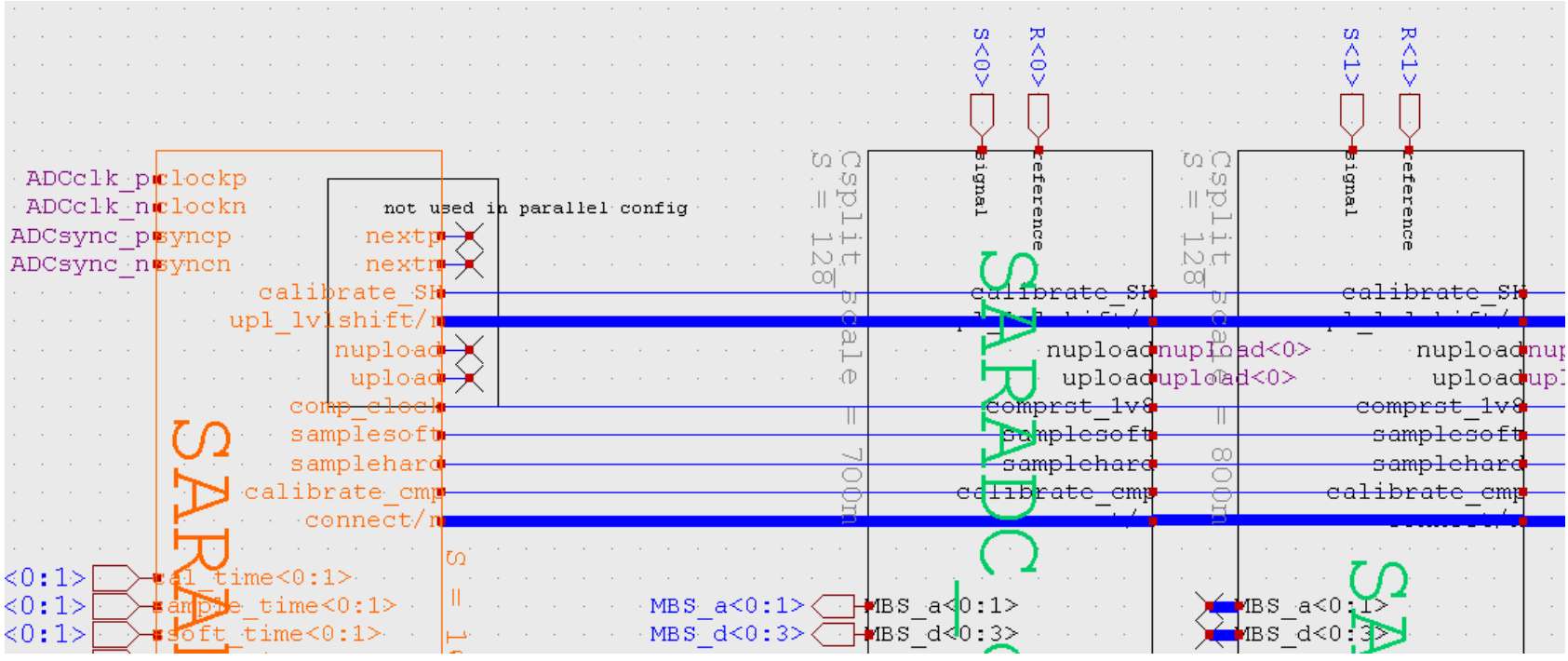
- Scalable, not needing to know number of parallel cores in advance
- Simpler circuit, no cleansync,
- Using DFF9 instead of DFF5 which still has risk of racing.

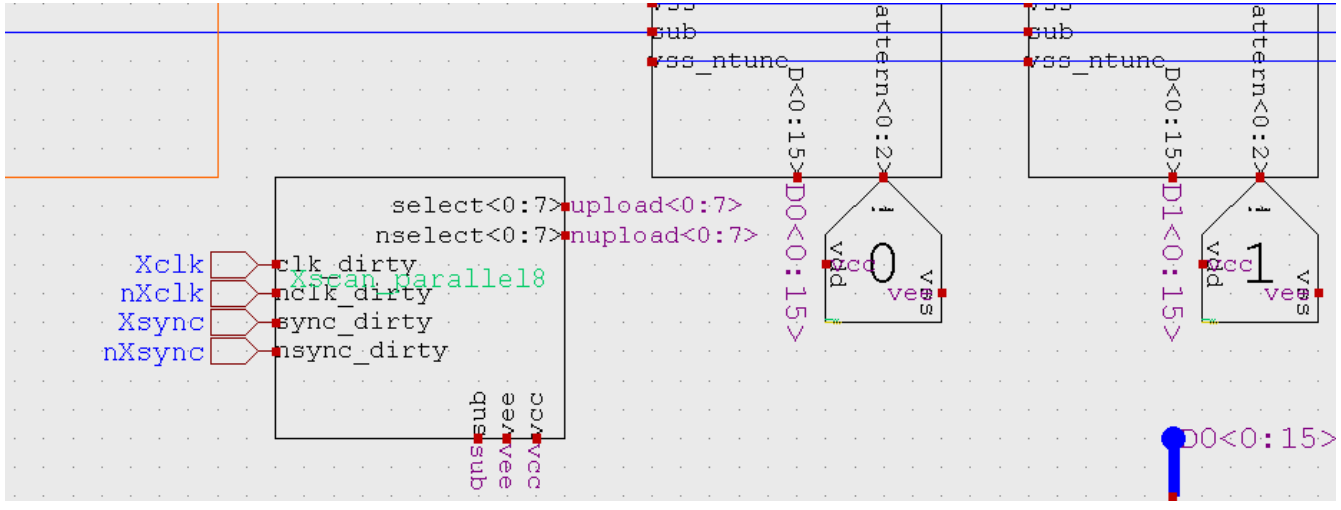
Sequencing upload<> in parallel configuration

The core ADCs are driven in parallel from the sequencer.

There is a separate little shiftregister enabling the upload from each core in sequence. The “upload<>” controls are made by a separate circuit xscan_parallel8.

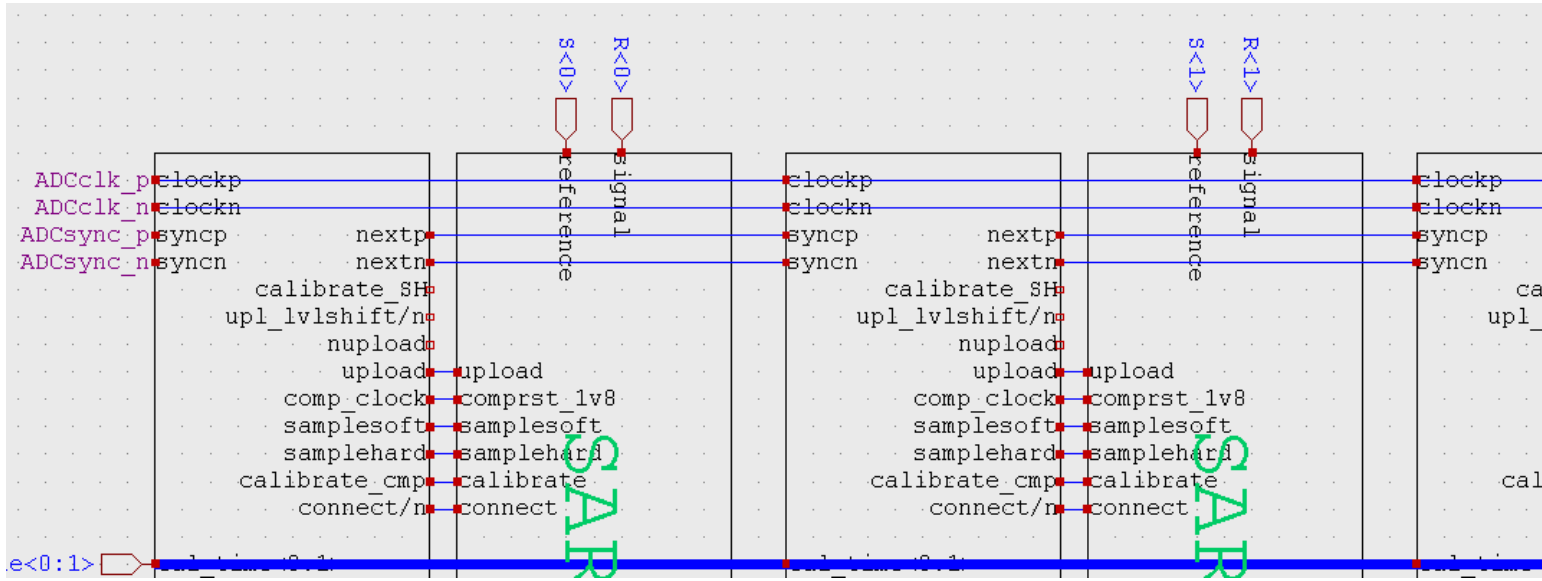
We propose to remove this and replace it by a solution as in next slide.



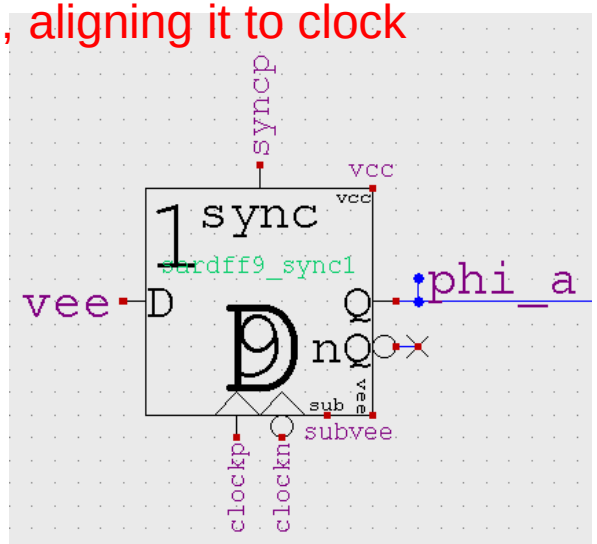


sync/next shiftregister when interleaving

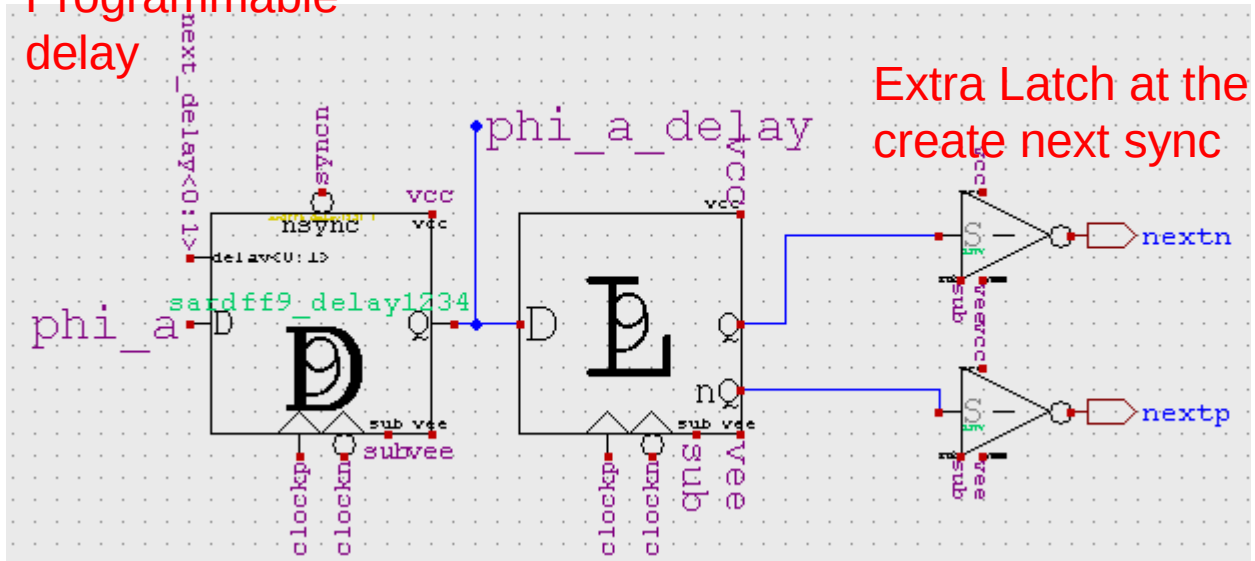
For use in the interleaving topology, each sequencer contains a short shiftregister, propagating its own “sync” with a delay to “next” sync of the next ADC sequencer&core. This realizes the delayed start of conversion for the interleaved ADC cores.



phi_a is generated by sync, aligning it to clock



Programmable delay



Extra Latch at the end to create next sync

“Cheap” true binary operation mode

caeleste

The non-binary ADC has 16 nominal bits, stemming from 15 consecutive capacitor pairs being toggled. Capacitor values decrease with factors ~1.8.

The difference with a simple classic SARADC is minimal. The essential difference are the factors being as exactly as possible equal to 2. However such configuration suffers from dynamic errors leading to missing codes around major bit transitions.

What if we compose the Capacitors partly of unit capacitors, so that they can be used in two modes

1. Non-binary, where capacitance ratios are ~1.8, all as previous
2. Binary where capacitance ratios are equal to 2 by unit capacitor number.
   - ← In this mode we take the burden that dynamic errors will happen. Probably not more that 8 useful bits
   - ← Yet, the raw ADC output is the final ADC output. No NB2B conversion needed

“Cheap” true binary operation mode

caeleste

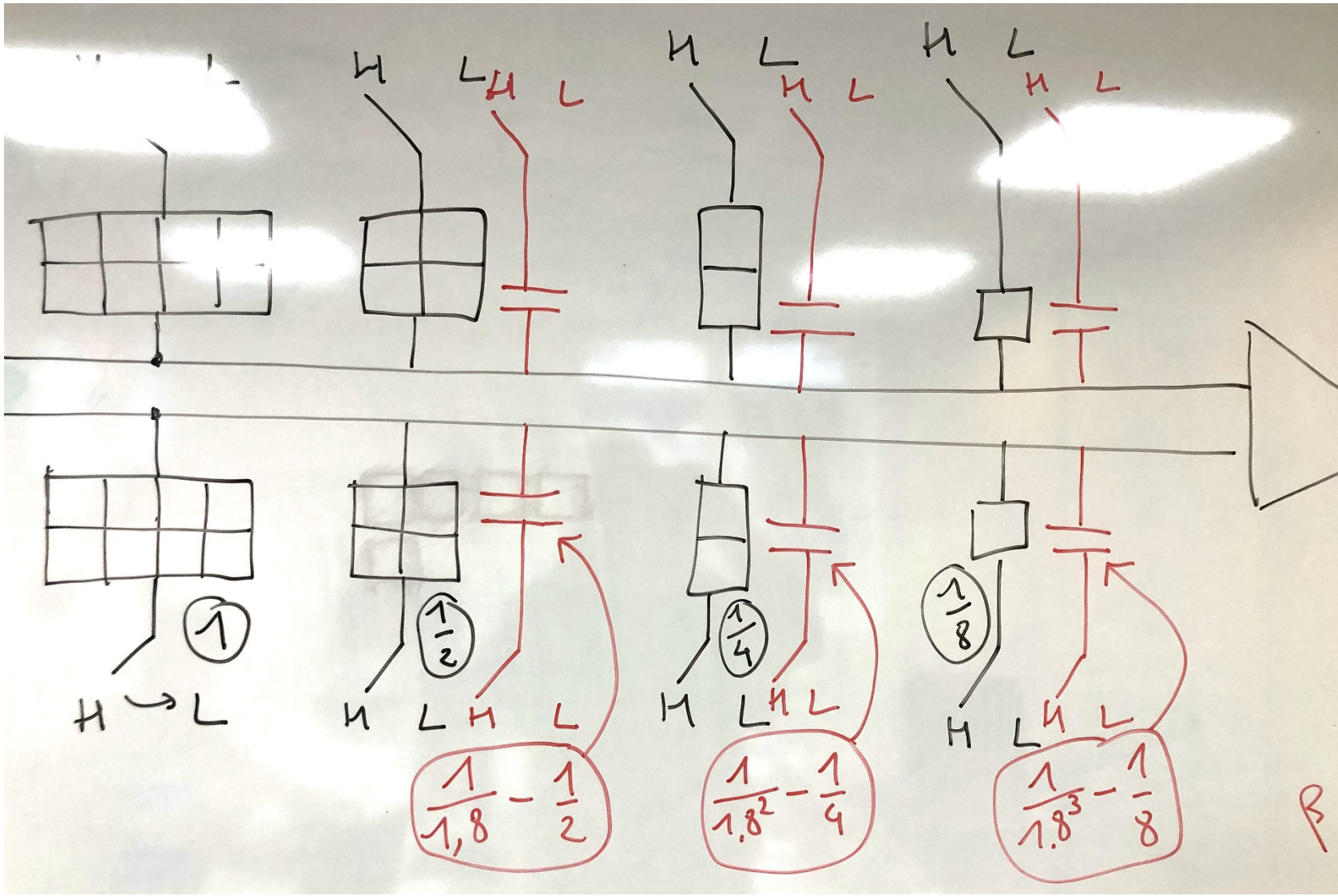
Implementation example as if only 4 bits.

The black 1:1/2:1/4:1/8:… series should be as exactl as possible.

The red extra capacitors could be only approximately.

In binary mode only the black part switches

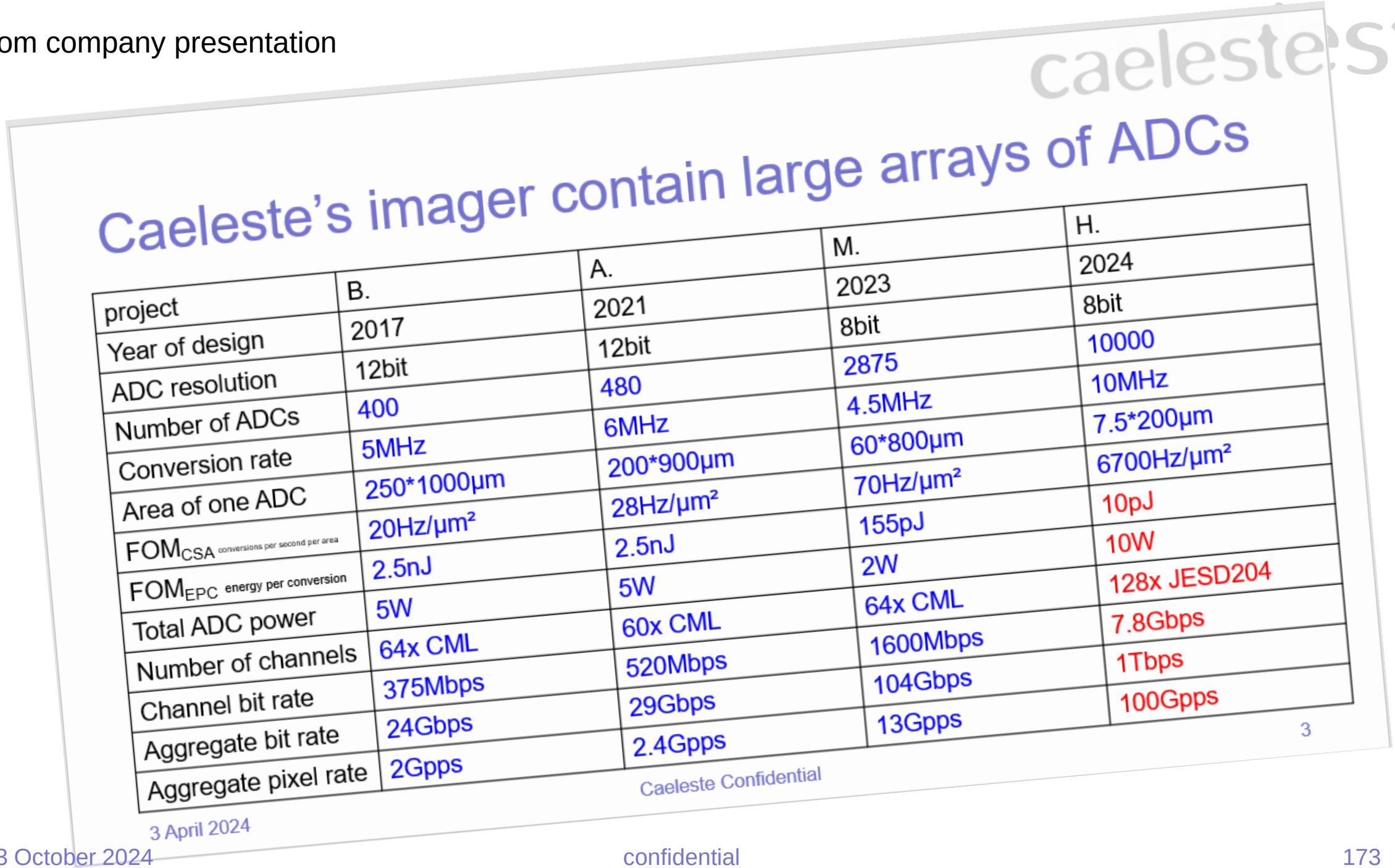
In non-binary mode both black and red parts switches.



section

3. Helena & SASC

From company presentation



Caeleste's imager contain large arrays of ADCs

project	B.	A.	M.	H.
Year of design	2017	2021	2023	2024
ADC resolution	12bit	12bit	8bit	8bit
Number of ADCs	400	480	2875	10000
Conversion rate	5MHz	6MHz	4.5MHz	10MHz
Area of one ADC	250*1000µm	200*900µm	60*800µm	7.5*200µm
FOM_{CSA} conversions per second per area	20Hz/µm²	28Hz/µm²	70Hz/µm²	6700Hz/µm²
FOM_{EPC} energy per conversion	2.5nJ	2.5nJ	155pJ	10pJ
Total ADC power	5W	5W	2W	10W
Number of channels	64x CML	60x CML	64x CML	128x JESD204
Channel bit rate	375Mbps	520Mbps	1600Mbps	7.8Gbps
Aggregate bit rate	24Gbps	29Gbps	104Gbps	1Tbps
Aggregate pixel rate	2Gpps	2.4Gpps	13Gpps	100Gpps

3 April 2024 Caeleste Confidential 3

Array of ADCs: helicopter view

caeleste

The trade-off space is obvious:

1. Maximize FOM_{CSA} **Conversions/second/area**
2. Minimize FOM_{EPC} **energy per conversion** for a given bit resolution
3. Pick the right resolution and perhaps make even this selectable/programmable. E.g. 1/4/8/12bits

Array of ADCs: designers viewpoint

caeleste

1. Suitable to put in an array
2. Aligned to [multiple of] column pitch
    - If ADC in the column: no analog domain memory effect and artifacts
3. Not sensitive to crosstalk by/to supply lines and DC references
4. Elegant power distribution
5. Differential versus SE (Pseudo-diff) signaling

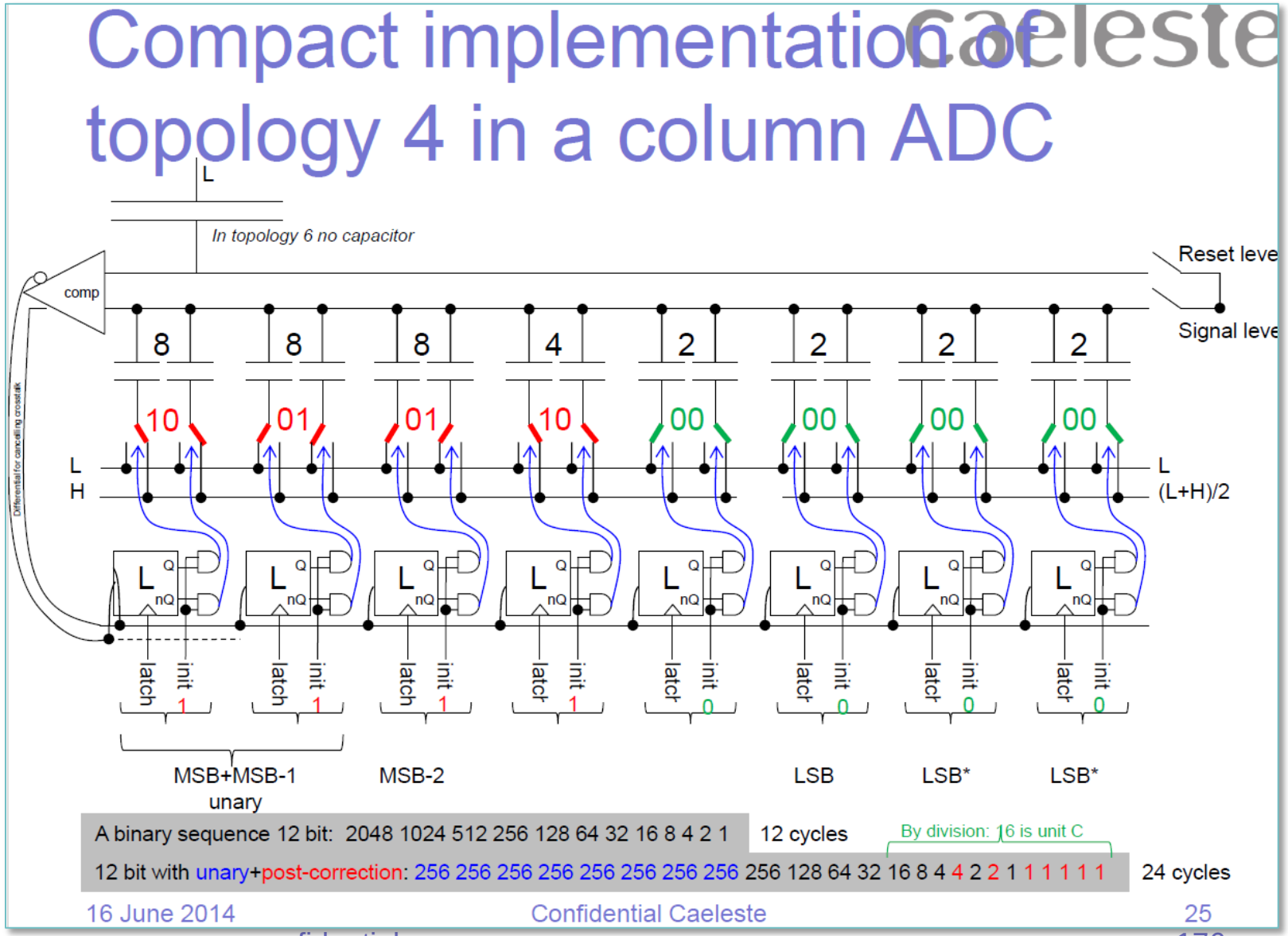
Approach:

Slide from Notebook 0045(!)→

Compact version of our classic SARADC, with double conversion and postcorrection.

Further:

- To optimize FOM_{CSA}, no unary and just a few double conversions.
- Fully differential
- “clock only high-low” (monotonic clocking)
- Non-binary remains option



Approach:

caeleste

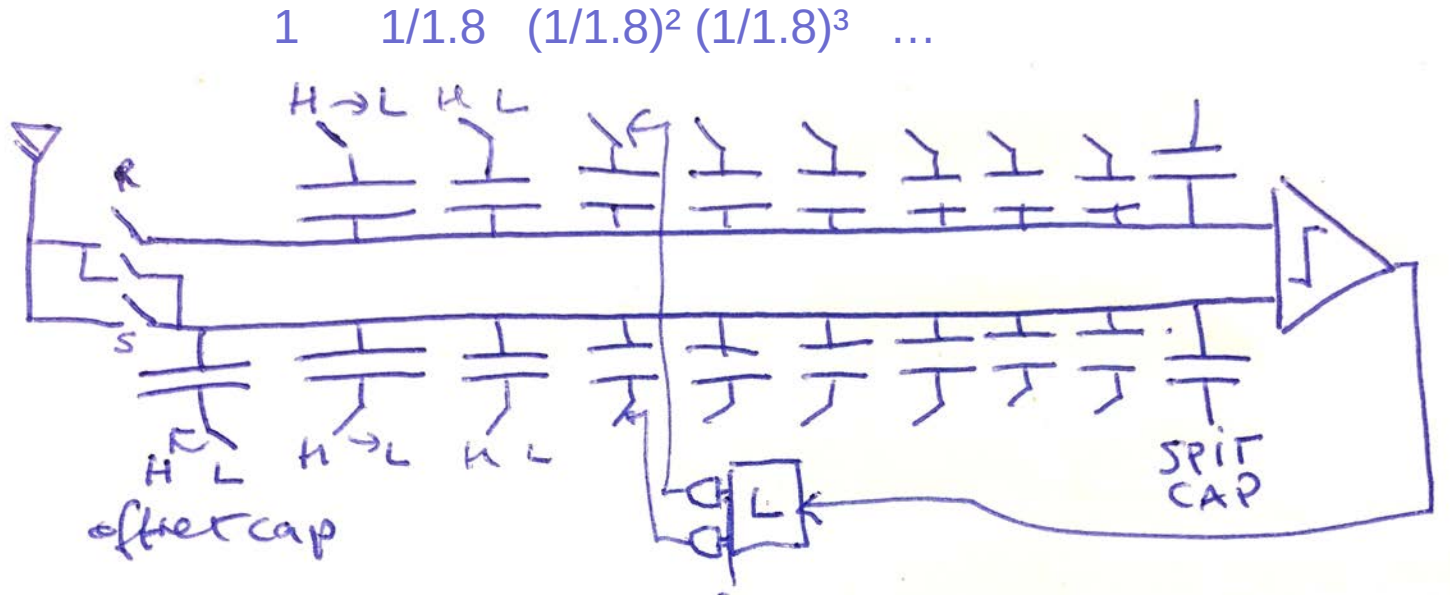
Identical logic as the previous

Fully differential and monotonic

Both binary and non-binary possible

1. Non-binary:
   - ← Most compact capacitors as matching is not really needed.
   - ← Slow settling to 10% is permitted as conversion error is corrected non-binary.
   - ← downside: need non-binary to binary re-coding outside the chip
2. Binary: using unit capacitors

Easy to make 4/8/12 bit operation: just give less clocks.



FOMs: EPC and CSA

caeleste

Figures of merit

- EPC (energy per conversion) must go down to 0.01 nJ range
- CSA (conversions per second per area) area must be something like 1000… µm²

routes are

1. Less bits
2. Smaller capacitors and other circuits.
3. Careful choice of circuit topology
4. No (or minimal) DC dissipators
5. Low supply voltages

Proposed solution in Notebook 0746

Exercise: Notebook 0746

In this Notebook we show that

- it possible to reach EPC=10pJ
- it is possible to layout the core ADC is 1000µm²
- Conversion rate for 8…10bit is at least 10MHz.

Layout size of a bit cell

The latch_nset in XC180 measures about 5.8x7.7=45µm²

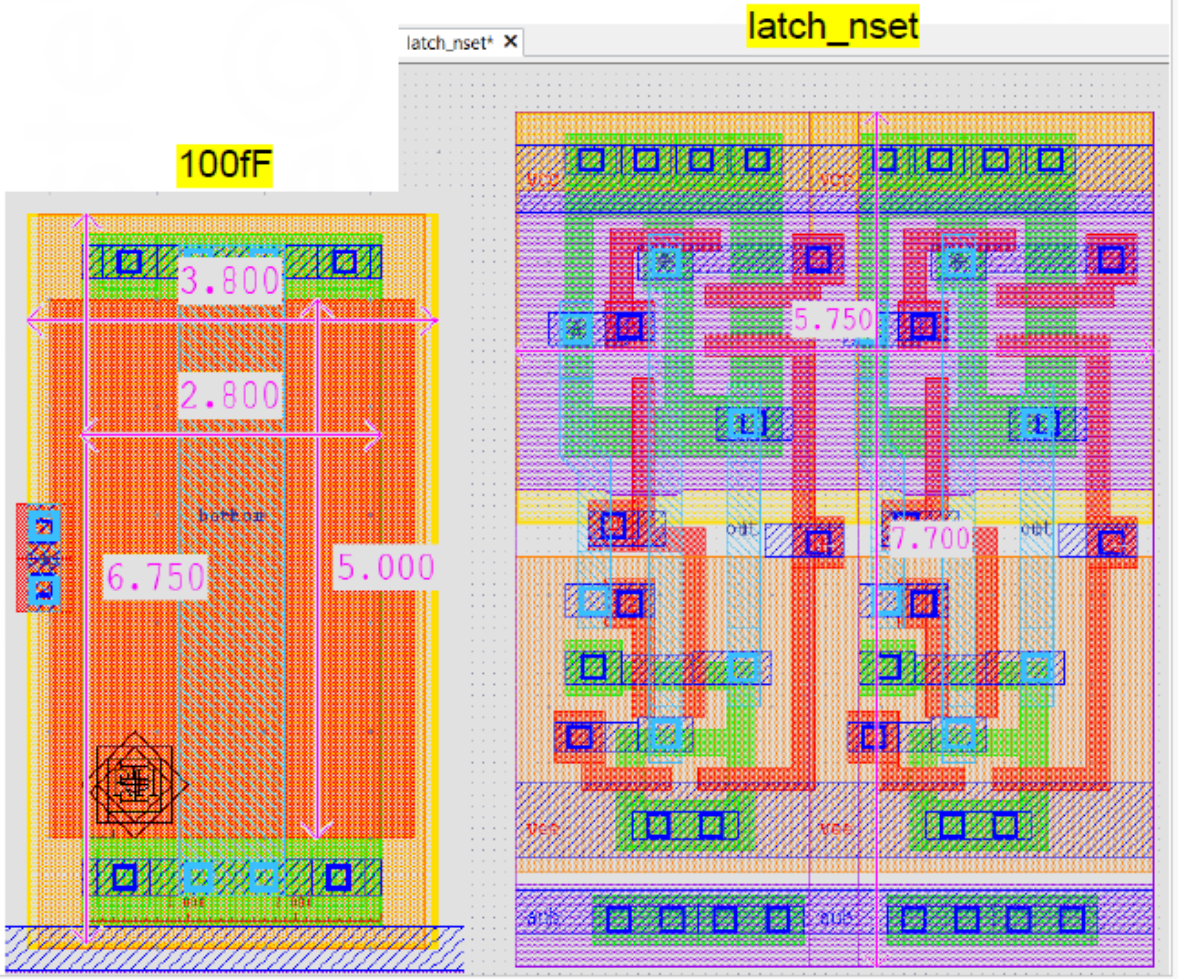
A 100fF POD cap proper measures 14µm². Including overhead about 26µm²

A bit cell depending on the capacitor value will likely measure not more than 50…100µm²

The bit cells of a 8 bit ADC fit in something like 500µm², in 180nm technology.

In 65µm one expects a factor ~4 less.

12 March 2024



ADC: selectable resolution

caeleste

It may be an attractive idea to exchange speed versus resolution in operation, then one can

e.g. nominal speed @4bit. Single bit resolution is probably useless or wo speed advantage
half speed @8bit (local)
third speed @12bit (local)
quarter speed @16bit (local) (only for non-binary or double conversion etc)

Implementation with a SARADC is +/- elegant as one clocks more/less

Output multiplexing of the ADC result to the next faster hierarchy is via a 4bit bus. One can access the bus 1, 2, 3… times

In case of double conversion ADC, one does the 4bit speed without double conversions (DC) and addition

The 8bit speed may be without DC, but with addition.

The 12bit case may have triple readout of 4bit bus, reducing it to effective 10 bit.

The 16bit case may have quad readout of the 4bit bus, reducing it to effective 12bit. Or 14? Or keep all. Anyway the downstream speed is more relaxed.

Use the SARADC *itself* as gainstage

caeleste

The cheapest: gain x2

a) Omit the offset switching to bring the reset level halfway the signal range. Tie the offset caps to vss which is better for PSR and noise.
b) Put the MSB without comparison to 0 and do not read its value out. Consider the MBS-1 as MSB (and barrel-shift the bits one position)

This is essentially gain 2x.

In a similar but more sophisticated way on can do the same with more bits, thus realizing 4x, 8x gain. Hardware shift the bits 1, 2, 3 positions.

e.g. this mode would be useful for low light imaging where number of grey levels is low.

- ✓ Take care that the suppressed (tied to zero) higher bits act do not cause overflow when the signal exceeds the range.
- ✓ When implementing this, do not enter performance limiting trade-offs

The “SASC” library

SARADC sub-circuits

First implementation of modular SARADC (20240601ba)

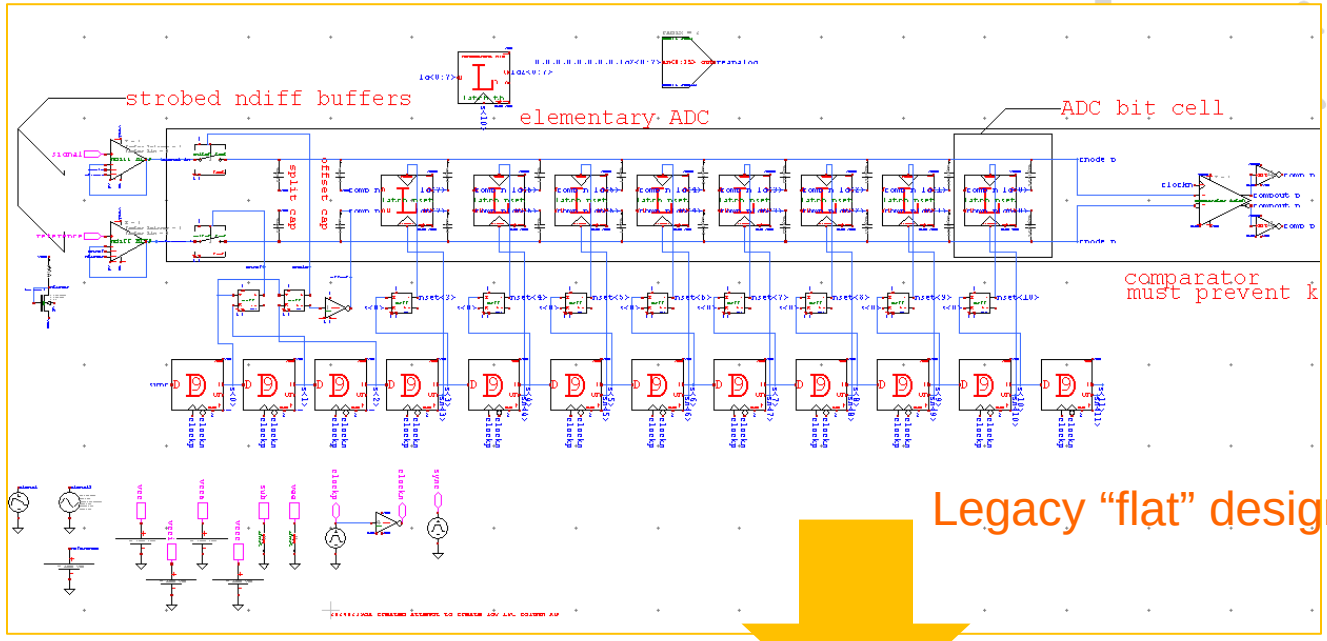
In-SARADC

1. sample*
2. cdac*
3. comparator*

Common

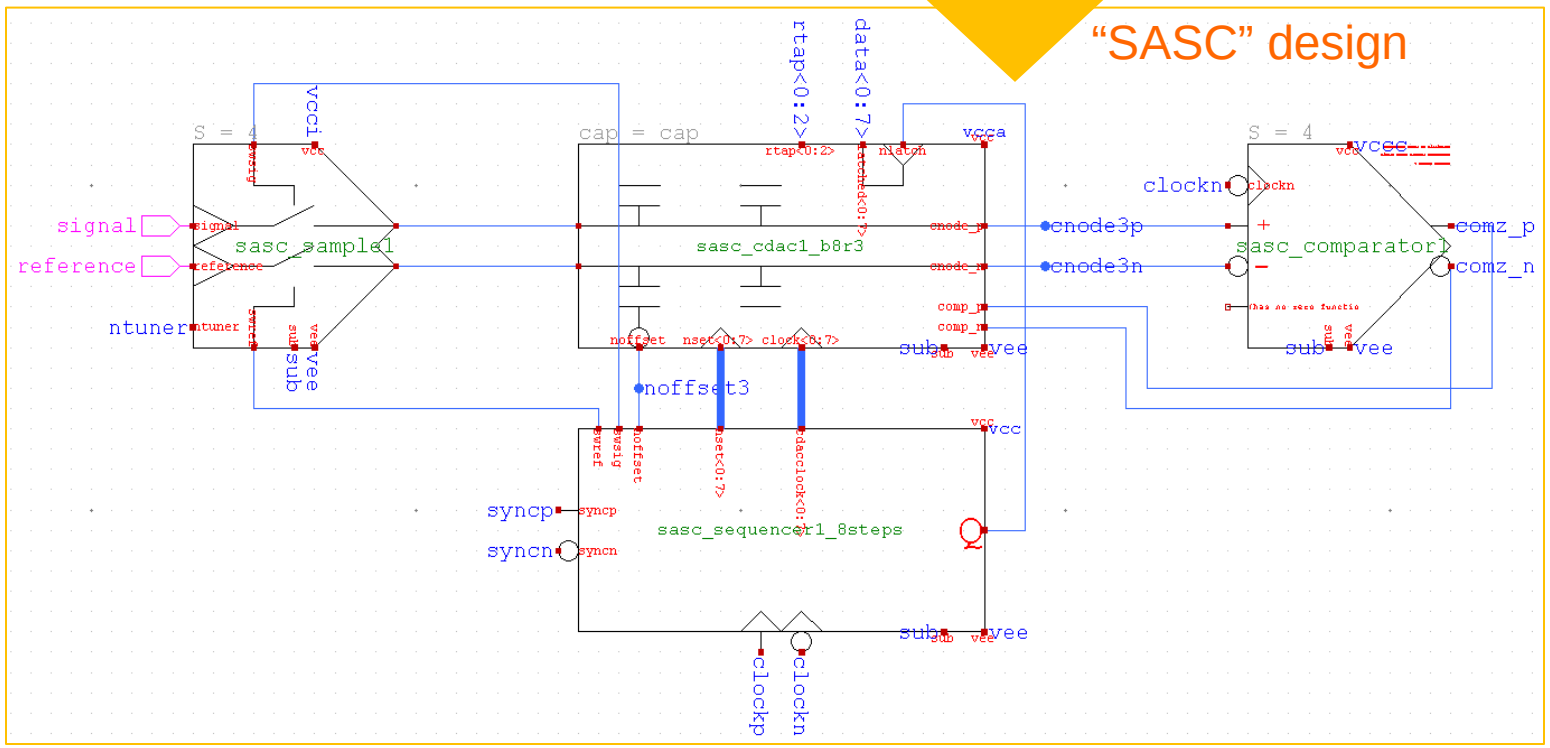
1. sequencer*
2. rdivider*

* means: pincompatible variants, topologies



Legacy “flat” design

“SASC” design



Comparator as in SASC

caeleste

For fast applications:

- Single stage preferred
- no tuner is needed if implemented as a pure clocked comparator
- No offset calibration is preferred, however, the option must exist
- this removes the time needed for the settling of first stage

Con:

- limited gain of one stage is not enough for high ENOB applications

… I need more thought and study on this
… is kick back ‘really’ an issue?

kTC noise of clocked latch/comparator is well understood

kTC and power consumption

caeleste

Case studied:

- 1V swing
- 12b ADC
- 250uV LSB
- Target: 0.25*LSB for each contributor (1pF equivalent)

CDAC: 1pF (total) on R input, 1pF on S input
Assuming a monotonic switching scheme,
(plus buttom plate parasitics)
1pF should be switched (1V)

comparator: For noise reasons, 1pF/16 as its output load (each side?) (assuming an implicit gain of 4)
Comparator will toggle around 16 times during one conversion
2pF should be switched (1.8V or 3.3V).

Input buffer: can be designed with probably same energy as CDAC

Modular SASC library (Notebook 0769)

caeleste

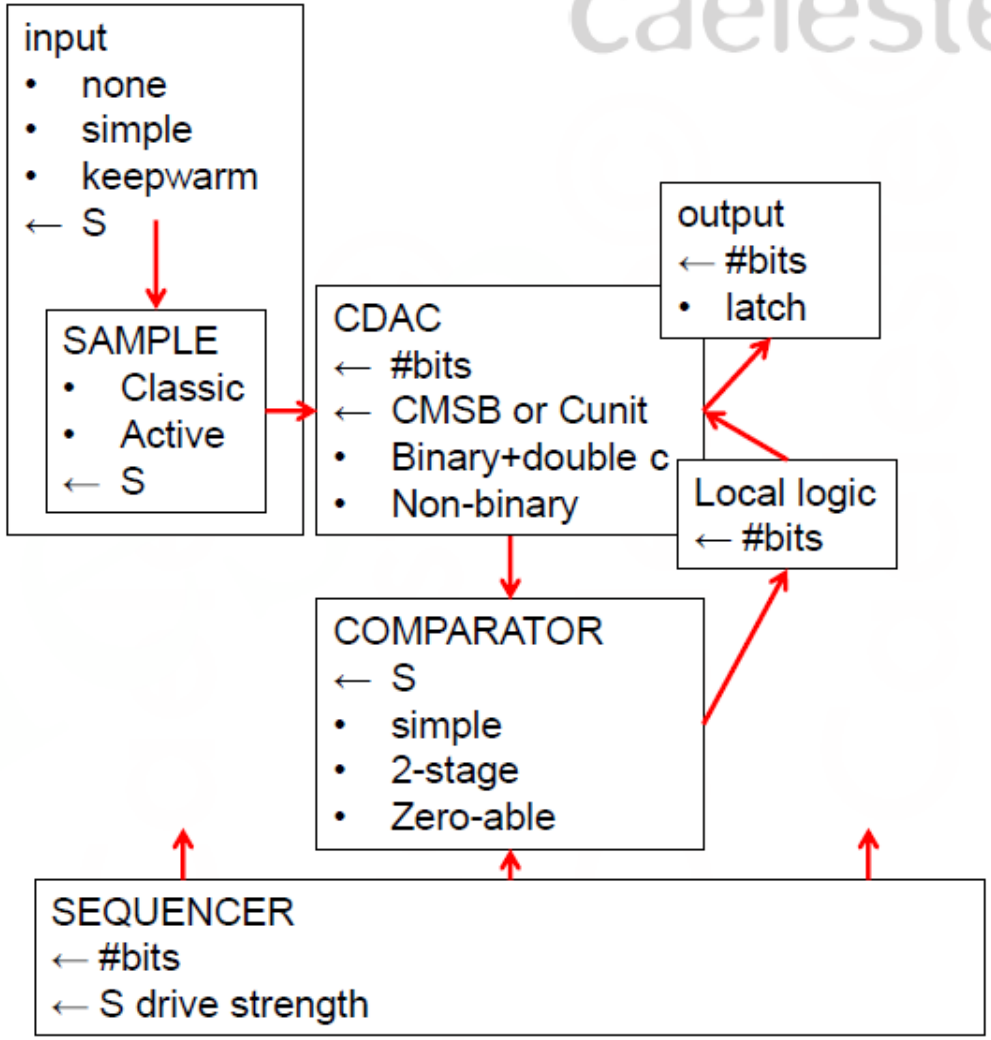
Ideas on ADC modularity

Legend
- topologies
- ← parameters
- ← fixed wire io

The idea of a “single IP block to be reused everywhere” breaks

Approach:

- the ADC is composed of a number of modules
- Modules have parameters (S, #bits…)
  - Some are hardwired, some soft
- Modules have circuit/topology variants
- Modules have a fixed interface symbol to enable easy exchange (correct by construction)
- This, at least in schematic

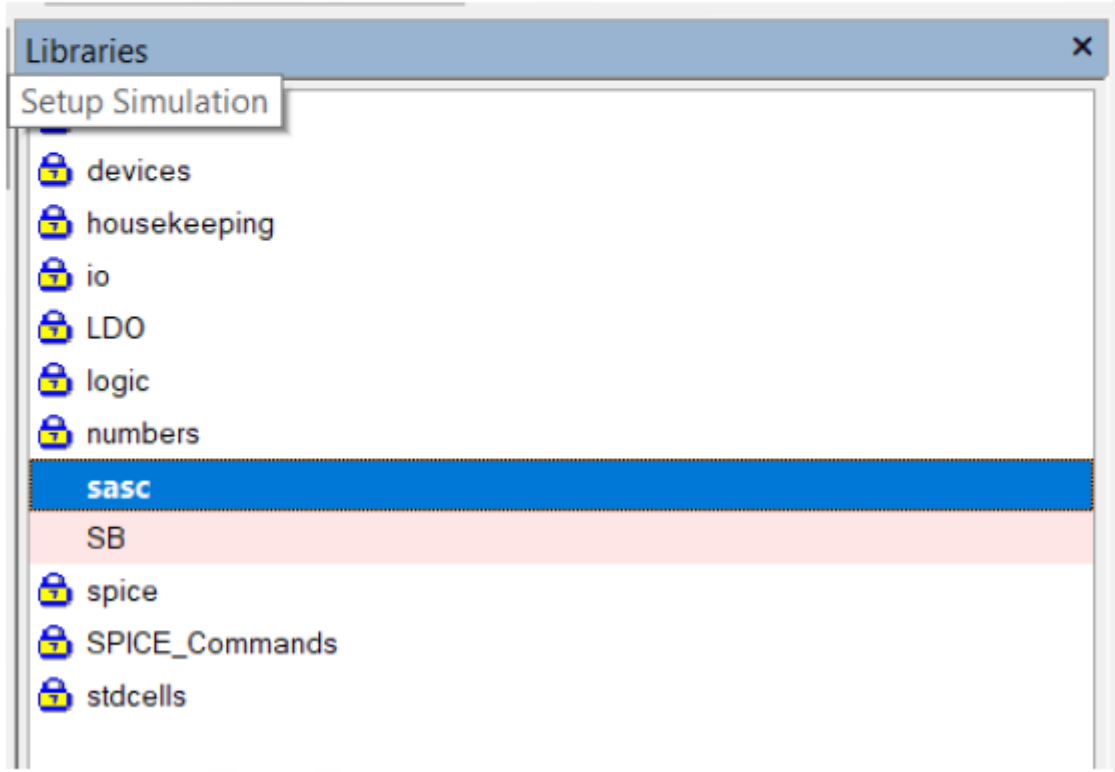


Modular SASC library (Notebook 0769)

caeleste

sasc (saradc subcircuit) library

As decided in SCIB meeting to place at this level



View	Library	Cell
symbol	sasc	_README
symbol	sasc	cdac1__b8r3
symbol	sasc	cdac1__b16r5d4
symbol	sasc	cdac1__nb16r8
symbol	sasc	cdac1_unitcell
symbol	sasc	cdac1_unitcellr
symbol	sasc	cdac1_unitcells
symbol	sasc	comparator1
symbol	sasc	rdivider1__b3
symbol	sasc	rdivider1__b5
symbol	sasc	rdivider1__b8
symbol	sasc	rdivider1__nb8
symbol	sasc	sample1
symbol	sasc	SB_all
symbol	sasc	SB_saradc8
symbol	sasc	SB_saradc8_column_OLD
symbol	sasc	SB_saradc12dc4
symbol	sasc	SB_saradc16nb
symbol	sasc	sequencer1__b8
symbol	sasc	sequencer1__b16
symbol	sasc	sequencer1_unitcell

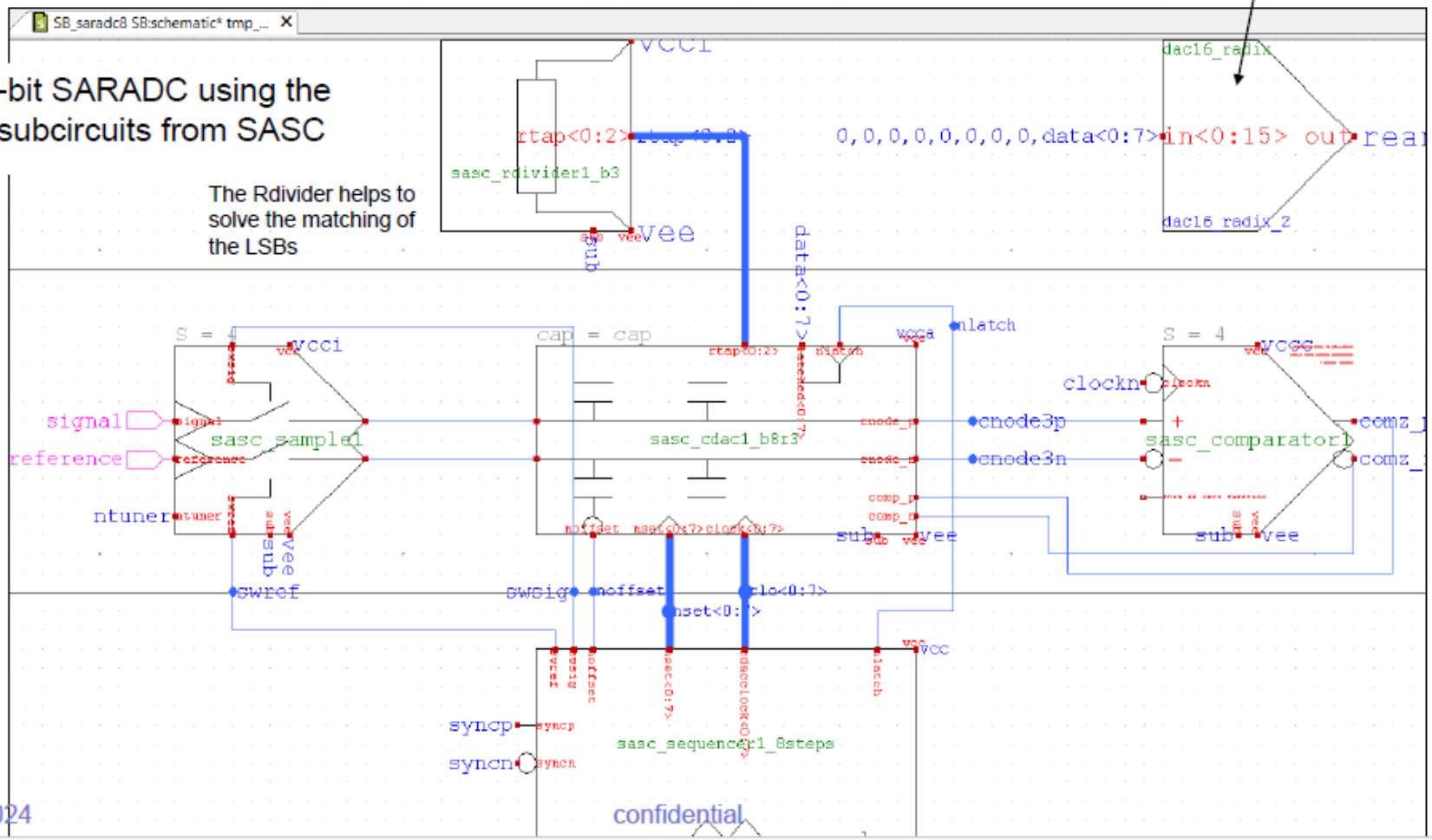
Modular SASC library (Notebook 0769)

Example simple 8bit SARADC

Simple 8-bit SARADC using the modular subcircuits from SASC

The Rdivider helps to solve the matching of the LSBs

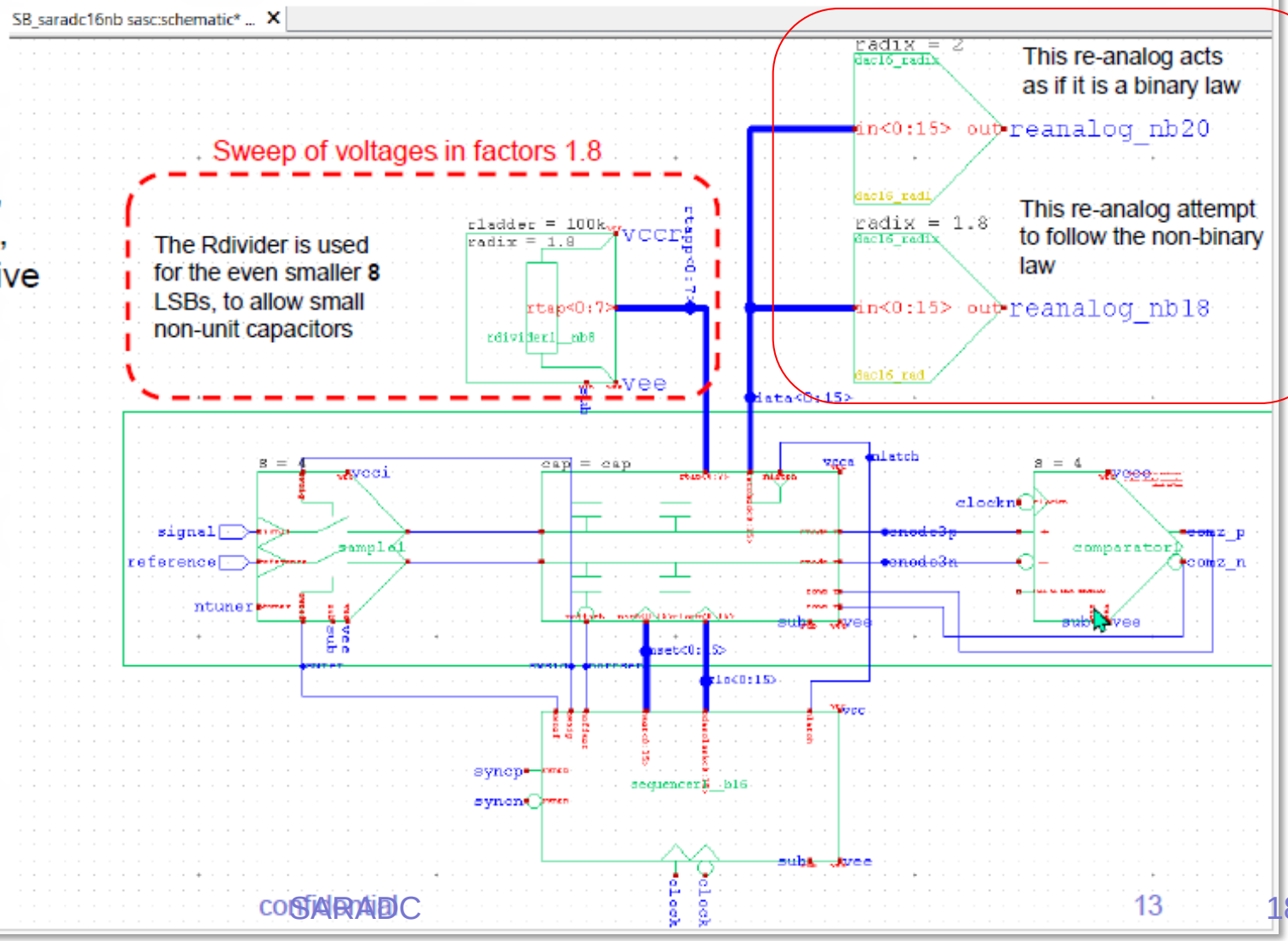
This dac16_radix calculator creates an analog value from the ADC output.



Modular SASC library (Notebook 0769)

Example 16bit non-binary =~14bit effective

Similar to the binary 12b4dc SARADC, except for the CDAC part. The CDAC part is simpler, with 8 bits using a C ratio of 1.8, followed by 8 bits using a resistive divider with 1.8 factors.



Features of SASC

In a binary SARADC with UB, DC and PC, one needs to add spurious bits input a single output word

Non-binary to binary conversion

- Contains 5 modules and variants
  - In-core: **sample**, **cdac**, **comparator**
  - Common: **sequencer**, **rdivider** (optional)
  - Not included (yet): multiplexing multiple ADCs, serializer, binary addition, NB2B
- Variant modules are schematic symbol compatible
- Variant modules have a sequence number appended: comparator1, comparator2…

SASC CDAC features

caeleste

SASC CDACs must comply to

- Monotonic switching (only high→low)
- Uses the cdac1_unitcell latch
  - cdac1_unitcells: has parameter “S” strength, for use with larger capacitors
  - cdac1_unitcellr: accepts the interpolated voltages of the rdivider, for the lower bits.
- Nature of the capacitors in not in the schematic cell.
- Binary variants are built from unit capacitors
- Non-binary variants have plain capacitors
- Binary variants can have unary bits (UB), double conversion (DC) and even postcorrection (PC).
  - Needing afterwads a simple addition to combine all spurious bits
- Less significant bits are based on a common resistive division on the “high reference”, created by a rdivider cell

SASC comparator features

caeleste

SASC “sample” (input stage) features

caeleste