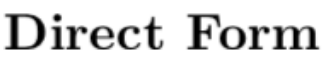
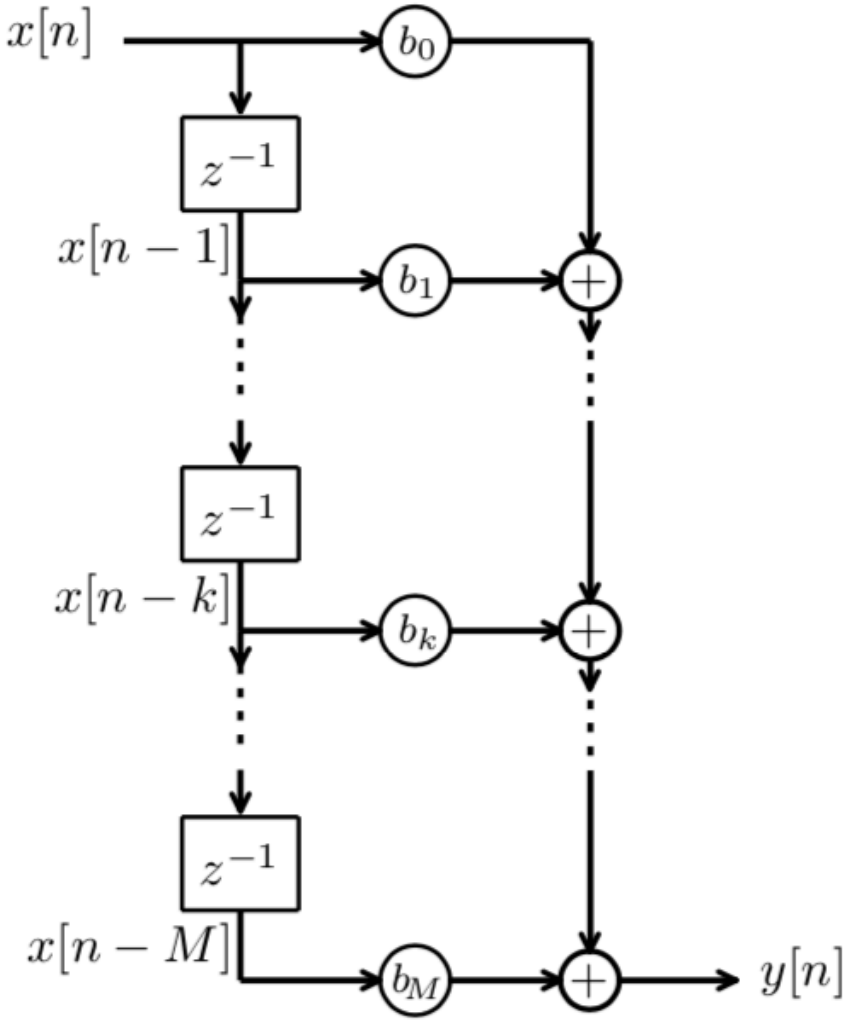
Why FIR uses often direct-form instead of transposed-form?

Asked 3 years, 9 months ago Modified 3 years, 9 months ago Viewed 1k times

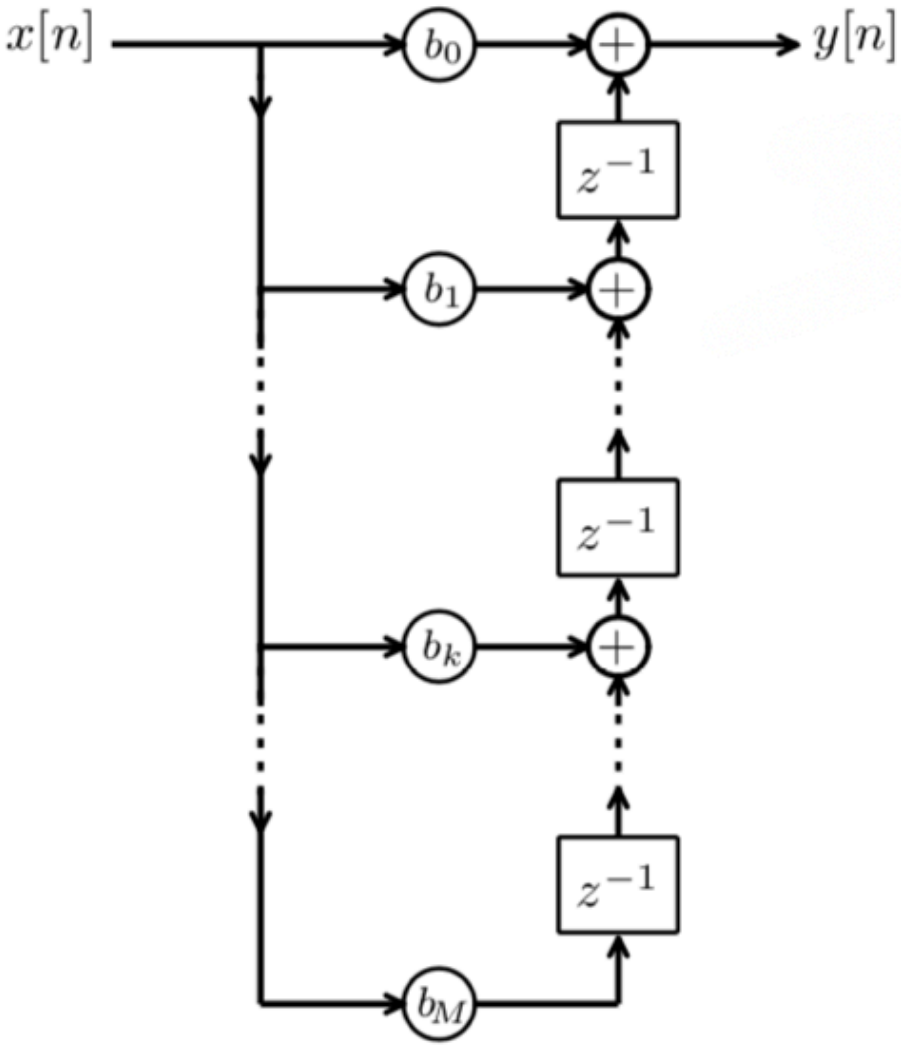
Even though tranposed-form has less computational burden and memory requirements, why FIR uses mostly direct-form?

2





Transposed Direct Form



filters discrete-signals signal-analysis finite-impulse-response digital-filters

Share Improve this question Follow

edited Jun 9, 2020 at 12:06

asked Jun 9, 2020 at 6:37

Ece Su Ildiz
161 7

2 I dont think that df1/df2/tf matters much for a FIR filter. Arithmetic complexity is going to be N mults and N-1 adds. For IIR filters choice of topology is often dictated by numerical considerations. – Knut Inge Jun 9, 2020 at 7:07

1 Can you show that the transposed form uses fewer computations and less memory than the transversal filter structure? – Matt L. Jun 9, 2020 at 10:44

This is because the direct form needs memory elements to realize the shift register for x[n] and memory elements to realize the adder trees in the coefficient multipliers, whereas the transposed direct-form can do it with just one kind of memory elements, which realize both at the same time, the shift register as well as the the coefficient-multiplier memories. – Ece Su Ildiz Jun 9, 2020 at 10:49

1 I don't see how memory or computational burden can possibly be reduced, perhaps you could

included a diagram of each showing that? – Dan Boschen Jun 9, 2020 at 11:47

Are you confusing FIR with IIR perhaps ? Transposed vs direct makes more of a meaningful difference for IIR filters – Hilmar Jun 9, 2020 at 12:02

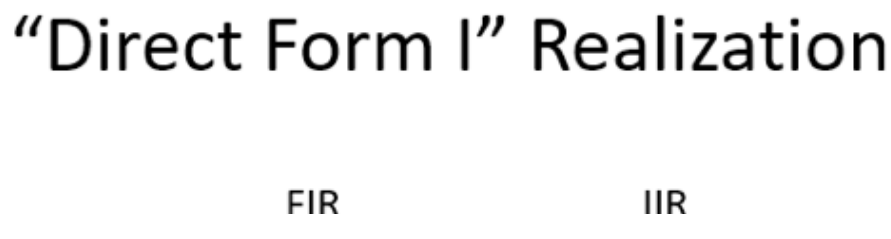
2 Answers

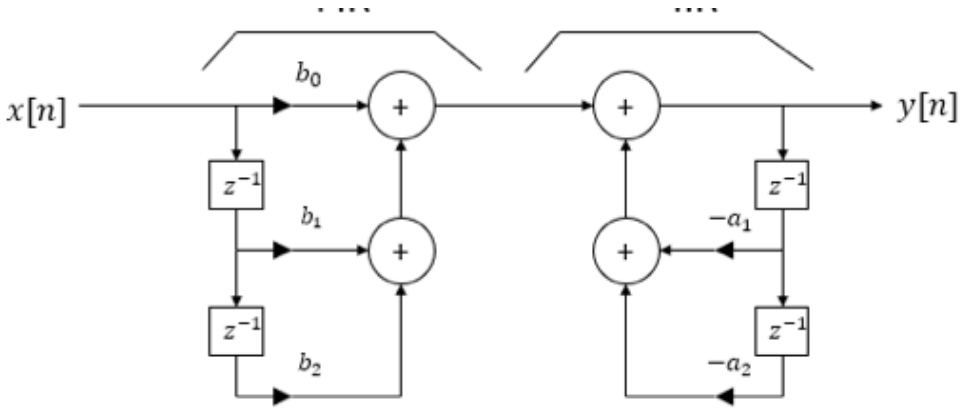
Sorted by: Highest score (default)

2

I don't see how memory or computational burden can possibly be reduced between the two implementations (note this is not implying simplicity, just number of computations required, please read on...) Usually the decision to use transposed-form is for high speed FPGA implementations with a large number of taps as you can eliminate a long adder tree which would otherwise limit the maximum clock frequency to a lower number due to the accumulated effect of multiplier and adder delay, but this can get more complicated for a fixed precision implementation since the bit width must grow as you proceed in the filter. (This consideration is particularly important in IIR implementations where feedback exists but affects FIR implementations with large adder trees, and also a reason to factor longer IIR filters in cascade form). At lower speeds the direct-form is more convenient to implement as you can use a single extended precision accumulator.

I include a summary of the basic forms applicable to FIR/IIR below for benefit of readers that may be less familiar, and see link from Ben in the comments for more advanced structures specific to FPGA implementations:

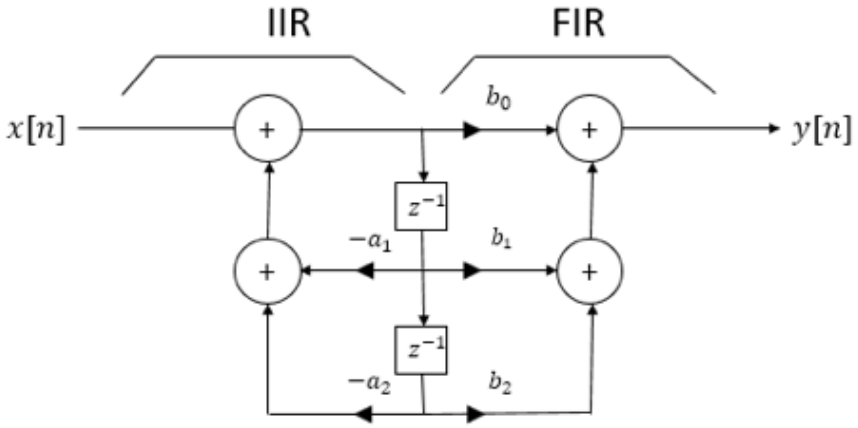




$$H(z) = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

6/9/2020 Copyright © 2019, Dan Boschen 28

"Direct Form II" Realization



$$H(z) = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

6/9/2020 Copyright © 2019, Dan Boschen 30

Transposing Filters

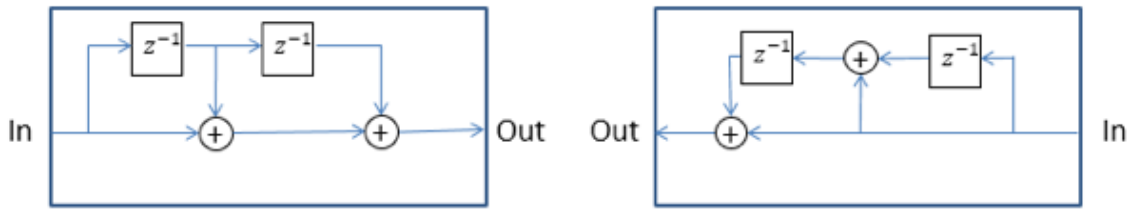
Transposing flow diagrams:

Change direction of all signal flows

Switch output and input

Summing nodes become splitting nodes

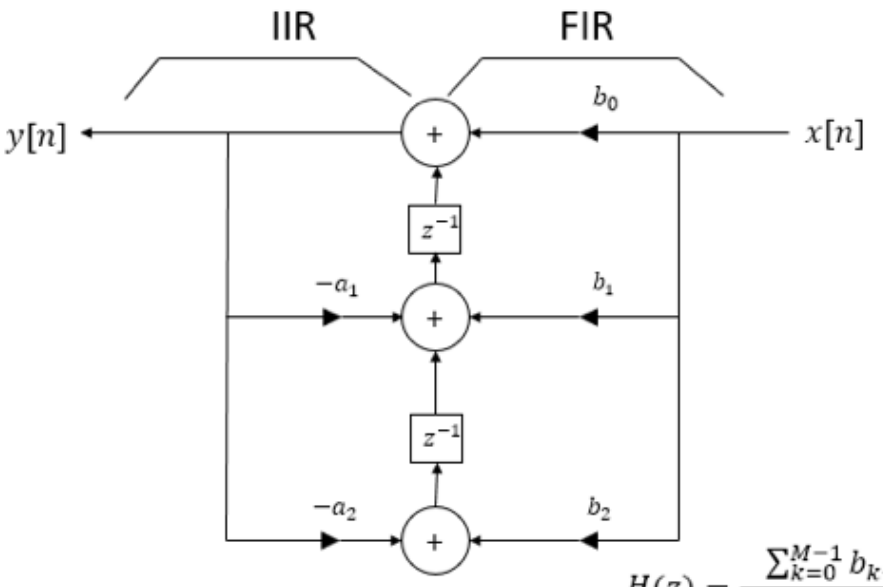
Splitting nodes become summing nodes



These are equivalent

6/9/2020 Copyright © 2019, Dan Boschen 31

Transposed "Direct Form II"



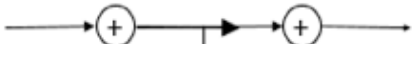
$$H(z) = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

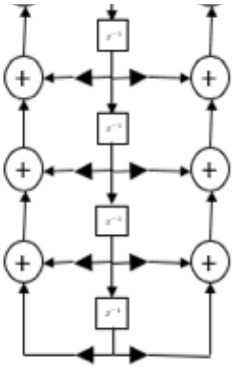
6/9/2020 Copyright © 2019, Dan Boschen 32

Cascade Form Realization

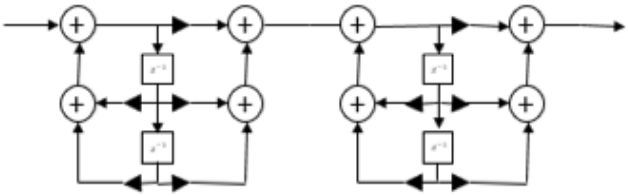
"Direct Form II" Realization

Cascade structure





Could end up being an oscillator as number of sections increase



2nd Order biquadratic sections:
Decouples poles and zeros
(Common structure: "bi-quads")

6/9/2020 Copyright © 2019, Dan Boschen 34

Share Improve this answer Follow

edited Jun 10, 2020 at 1:54

answered Jun 9, 2020 at 11:48

Dan Boschen
50.8k 2 57 135

Thank you Sir, I am sorry, I just used the information from my lecture. Now it makes sense. – Ece Su Ildiz Jun 9, 2020 at 11:55

@EceSulldiz I added some figures in case we are referring to different things – Dan Boschen Jun 9, 2020 at 12:02

Yes I was talking about FIR, I also added some figures. – Ece Su Ildiz Jun 9, 2020 at 12:06

@EceSulldiz Nice, thank you. Notice there are the same number of memory elements and computations? There is no need to keep a buffer of the output of each of the adders in memory, just the final result (so in the direct form 1 the right side represents a single accumulator of each of the scaled and delayed outputs). The time to accumulate everything may be limited, in which case the transposed form is convenient since it can all be done at once in synchronous fashion (good structure in an FPGA to have a synchronous element at the output of every adder versus accumulating delays. – Dan Boschen Jun 9, 2020 at 12:09

1 For IIR filters implemented in FPGAs, adder trees delay don't matter much. Multiplier delays are the biggest performance killers. Hence techniques like "scattered-look-ahead" and "IIR parallelization" people.ece.umn.edu/users/parhi/SLIDES/chap10.pdf – Ben Jun 9, 2020 at 12:21

1

if the implementation preserves bits until quantization **must** occur at the final output, the Direct Form is far simpler than the Transposed Form. using the transposed form requires that your states have double-wide word widths unless you quantize each of those states back to single width. but that is more quantization error than if you just add up a bunch of double-wide words and quantize the result.

Share Improve this answer Follow

answered Jun 9, 2020 at 16:09

robert bristow-johnson

20.6k 4 38 76