*LIGO Laboratory / LIGO Scientific Collaboration*

LIGO-T2400089-v1 Advanced LIGO March 2024

**IIR Filter Bank for FPGAs**

Daniel Sigg

Distribution of this document:

LIGO Scientific Collaboration

This is an internal working note

of the LIGO Laboratory.

|  |  |
| --- | --- |
| **California Institute of Technology****LIGO Project – MS 18-34****1200 E. California Blvd.****Pasadena, CA 91125**Phone (626) 395-2129Fax (626) 304-9834E-mail: info@ligo.caltech.edu | **Massachusetts Institute of Technology****LIGO Project – NW22-295****185 Albany St****Cambridge, MA 02139**Phone (617) 253-4824Fax (617) 253-7014E-mail: info@ligo.mit.edu |
| **LIGO Hanford Observatory****P.O. Box 1970****Richland WA 99352**Phone 509-372-8106Fax 509-372-8137 | **LIGO Livingston Observatory****P.O. Box 940****Livingston, LA 70754**Phone 225-686-3100Fax 225-686-7189 |

http://www.ligo.caltech.edu/

Table of Contents

[1 Overview 3](#_Toc163552972)

[2 The Filter Engine 3](#_Toc163552973)

[3 The Filter Bank 5](#_Toc163552974)

[3.1 Coefficient Memory 6](#_Toc163552975)

[3.2 Microcode 7](#_Toc163552976)

[3.3 Changing Coefficients 7](#_Toc163552977)

[4 Pipelines 8](#_Toc163552978)

[4.1 5x Configuration 8](#_Toc163552979)

[4.2 2x Configuration 8](#_Toc163552980)

[4.3 5x Configuration with SOS Switches Enabled 10](#_Toc163552981)

# Overview

An IIR filter can be implemented as a series of second order sections (SOS). Here we look at an FPGA implementation written in VHDL code in Vivado and targeting the Xilinx 7‑series and Ultrascale devices. The filter implementation uses a fixed-point number representation with 52‑bits that is roughly equivalent to a double precision number.

Filter coefficients can be generated from a foton filter file using the CoeffGen52\_2x program. The coefficient memory is a dual ported RAM that has a 64-bit wide read-only mode on the filter side, and a 32/64-bit wide read-write mode on the interface side. Typically, the interface side is accessible through the PCIe bus.

# The Filter Engine

Following Reference[[1]](#footnote-1) we write a filter as:

|  |  |
| --- | --- |
| $$H\left(z\right)=g\prod\_{k=1}^{N}c\_{0k}\frac{1+b\_{1k} z^{-1}+b\_{2k} z^{-2}}{1-a\_{1k} z^{-1}-a\_{2k} z^{-2}}$$ | (1) |

In the above equation we separated out an overall gain factor, $g$, and individual gain factors, $c\_{0k}$. The idea is to implement the multiplications $c\_{ok}$ as an efficient shift operation which reduces a second order section to 4 MAC (multiply-accumulate) operations that can be implemented efficiently using the DSP slices available in the FPGA. This shift operation for each SOS is important to keep the scaling near unity at a characteristic frequency and to take full advantage of all the digits in the fixed-point representation.

Restricting ourselves to a single second order section and denoting input values by $x\_{i}$ and output values with $y\_{i}$ the filter section can be written as:

|  |  |
| --- | --- |
| $$y\_{i}=c\_{0}\left(x\_{i}+b\_{1}x\_{i-1}+b\_{2}x\_{i-2}\right)+a\_{1}y\_{i-1}+a\_{2}y\_{i-2}$$ | (2) |

Since we have chosen the direct form I structure[[2]](#footnote-2) rather than the more traditional direct form II, both input and output values serve as history values. And, since we set up the filter to be near unity gain, a suitable fixed-point representation is straight forward. Since the output value of the first second order section is the input value of the next section, the entire filter can be implemented using a single multiply-accumulate pipeline, where the very first input value must be scaled by the gain $g$ (see Figure 1). History output values are shared with the history input values of the next section; therefore, we are only requiring two additional history values compared to the direct form II structure.

The implementation of the filter engine includes optional output registers, a separate output for the old input value, and optional switches that allow to turn off individual sections. Since the gain of a second order section isn’t unity, a disabled SOS has to be replaced by a simple gain multiplication using a bypass path.

Additional logic is used to process overflow conditions. An overflow condition occurs if the result of the acculamator cannot fit into the bits of the output value. An overflow is also signaled if the input indicates an overflow from a previous processing stage.

The parameters that are used to configure the filter engine:

Figure : Filter Engine.

Bypass

Input Value

Output Value

Mux

History Buffer

Accumulator

Shifter

Register

Register

Gain, Coefficients, Shift

Output Overflow

Microcode

Odd/Even Cycle

Input Overflow

Clock Enable

Clock

Reset

Delay

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Default** | **Description** |
| InputWidth | Natural | 32 | Bit width of filter input and output values |
| OutputReg | Boolean | True | Determines if an output register is implemented |
| Cycles | Natural | 6 | Base 2 logarithm of number clock cycles within a filter cycleCan be 3 (8 cycles), 4 (16 cycles), 5, 6, 7, 8 or 9 (512 cycles) |
| ShiftBits | Natural | 6 | Base 2 logarithm of number bits used in the barrel shifter Shift values range from 2−(ShiftBits−1) to 2ShiftBits−1−1 |
| Multiplier | String | 5x | Multiplier configuration: “2x” uses 2 multipliers (DSP slices) in parallel, “5x” uses 5 slices |
| DSP | String | DSP48E1 | DSP slice configuration: “DSP48E1” for 7-series devices and “DSP48E2” for Ultrascale |

If the “2x” configuration is selected, The OutputReg parameter should be set to True, since the availability of the filter output value depends on the Cycles count. With a register the filter value will always be available at the last cycle of the filter cycle or later. For a “5x” multiplier the output register can be omitted if the filter value is clocked out during the last cycle of the filter cycle. If a register is used, the filter value will be available on the first cycle of the next filter cycle. Also, for the “2x” configuration the minimum Cycles count is 5 (32).

# The Filter Bank

Figure : Filter Bank

Time

Clock Divider

Clock/Reset

Microcode

Memory

Cycle Number

Address

Filter Engine

Dual Ported RAM

Coefficients

Filter Input

Filter Output

Control

Odd/Even Cycle

Filter Selection

Switches

Clock Enable

The IIR\_filter\_bank module implements a bank of IIR filters, where each filter is using identical filter coefficients. A filter bank consists of a clock divider, a microcode generator, a memory block for the filter coefficients, and an IIR filter engine that can work on multiple input values in parallel (see Figure 1).

The clock divider generates a cycle count that enumerates through all filter steps and that is aligned to the 1 PPS of the GPS time. It generates a clock enable when the filter is intended to run slower than the clock rate. An odd/even cycle bit is used to switch the filter history values at the end of a cycle.

The microcode block generates both the control signals and the coefficient memory address based on the cycle count.

The memory block implements either a ROM or a dual ported RAM that contains the gain, filter coefficients and shift values. Typically, the memory block is larger than required for a filter. In this case the parameters of multiple filters can be loaded into the memory and selected using a filter selection input.

The parameters that are used to configure the filter bank:

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Default** | **Description** |
| RESOLUTION | Natural | 26 | Timing clock resolution or period is 2−RESOLUTION seconds |
| Cycles | Natural | 6 | Base 2 logarithm of number clock cycles within a filter cycleCan be 3 (8 cycles), 4 (16 cycles), 5, 6, 7, 8 or 9 (512 cycles) |
| LBD | Natural | 0 | Base 2 logarithm of clock divider ratio to run filter slower than clock rate; Must be lower or equal RESOLUTION |
| Filters | Positive | 1 | Number of filters in the filter bank |
| FilterWidth | Natural | 32 | Bit width of filter input and output values |
| FilterReg | Boolean | True | Determines if an output register is implemented |
| ShiftBits | Natural | 6 | Base 2 logarithm of number bits used in the barrel shifter Shift values range from 2-(ShiftBits-1) to 2ShiftBits-1-1 |
| Multiplier | String | 5x | Multiplier configuration: “2x” uses 2 multipliers (DSP slices) in parallel, “5x” uses 5 slices |
| DSP | String | DSP48E1 | DSP slice configuration: “DSP48E1” for 7-series devices and “DSP48E2” for Ultrascale |
| GainSwitch | Boolean | False | Enable setting the filter gain to 0 |
| SosSwitch | Boolean | False | Enable individual switching of SOS stages |
| ResetType | String | Full | Reset type: “Full” - apply the reset for a full cycle and release at the end of a cycle, “Instant” - apply only as long as the reset line is active, “Goertzel” - apply every second during the last cycle |
| MemoryType | String | TPRAM | Memory type used: “SPROM” (ROM) or “TDPRAM” (RAM) |
| MemoryDepth | Natural | 10 | Memory depth in number of address bits: The minimum is 10 for TDPRAM, and 9 for SPROM |
| MemoryBank | Natural | 7 | Memory bank size in address bits: The minimum is 6, and up to the MemoryDepth Must be long enough to contain the required SOS sections |
| MemoryDelay | Natural | 0 | Additional delay in the output of the coefficient memory: 0 - data is available on the next clock cycle, 1 - uses an output register |
| MemoryFile | String | “none” | Memory initialization file |
| MemoryReg | Natural | 1 | Memory pipeline depth on the interface side of the dual port RAM: 1 - use of memory latch only; 2 - use of output register |
| DataBWidth | Natural | 32 | Data width on B side |

## Coefficient Memory

The coefficient memory is organized by filter sets. Each filter set covers an entire IIR filter. It is then possible to switch between filter sets using the filter selection input to load alternative filter coefficients. Typically this filter selection input is mapped into the PCIe parameter space and can be accessed through the host computer.

If the SosSwitch parameters is enabled, some space usually used for filter sets will be used for the alternate gain coefficients instead. This means that in this case the coefficient memory needs to be big enough to cover at least 2 filter sets. If big enough for just 2 filter sets, the space of the second filter set will be used for the alternate gain coefficients. If big enough for 4 or more filter sets, a quarter of the available space will be used for the alternate gain coefficients. Currently, SOS switches are only supported when the multiplier is set to “5x”.

The space of each filter set is sub divided into groups of 4 filter coefficients. The first group is used for the zero value, 2 64-bit values reserving up to 128 bits for the gain and SOS switches, and the overall gain coefficient, respectively. Each subsequent group represents a single SOS section with its $b\_{1}$, $b\_{2}$, $a\_{1}$, and $a\_{2}$ coefficients, respectively. The $c\_{0}$ value is stored as part of the $a\_{2}$ coefficient.

Gain and filter coefficients are represented by 53-bit fixed point values that are shifted left into a 64-bit quad word. The 53-bit coefficient values are two’s complements values with the MSB being the sign bit and with the decimal point being located between the 2nd and 3rd bit. This allows for coefficients to represent any value between -2.0 (inclusive) and +2.0 (exclusive). The shift value $c\_{0}$ is encoded in the unused lower 8-bits of the $a\_{2}$ coefficient. This shift value is an 8-bit two’s complements value ranging from -32 to +31. This allows for multiplications in the range of 2-32 to 231 in multiples of 2.

The SOS switches are bit encoded in the 2 64-bit values in the first group of coefficients. The first value represents the gain switch at the LSB, then the SOS switches for the first SOS up to the 31st SOS. The second value represents the SOS switches for the 32nd SOS up to the 63rd SOS. A zero for the gain bit indicates that the normal gain is used, whereas a one indicates that the normal gain value is replaced by zero. For the SOS switches, a zero indicates a normal calculation of the section, whereas a one indicates that the section is off and the alternate gain value is used instead.

If filters are designed using the LIGO foton program, the CoeffGen52\_2x program[[3]](#footnote-3) can be used to generate the mem files needed by Vivado to preload the coefficient memory.

## Microcode

The microcode block generates both the control signals and the coefficient memory address based on the cycle count. Examples for the “5x” multiplier configuration can be found in section 4.1, whereas section 4.2 shows the “2x” configuration. Section 4.3 shows an example where some of the SOS switches are set to off.

For the “5x” configuration 4 cycles are needed to calculate a single SOS, whereas for the “2x” configuration 12 cycles are needed. For the “5x” configuration 4 cycles are needed to apply the overall gain and flush the pipeline, whereas the “2x” configuration requires 7 cycles of overhead. The table below shows the available SOSs for filters with different cycle count:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Config. | 8 cycles | 16 cycles | 32 cycles | 64 cycles | 128 cycles | 256 cycles | >256 cyc. |
| 2x | — | — | 2 | 4 | 10 | 20 | 31 |
| 5x | 1 | 3 | 7 | 15 | 31 | 63 | 127 |

## Changing Coefficients

Changing coefficients in a filter memory section that is currently used for computing an IIR filter is not recommended. There is no guarantee that the coefficients will be updated at the start of the filter calculation and not half way through. If new filter coefficients are to be loaded for an online filter, one should load the new filter coefficients into an unused filter memory section and then use the filter selection bits to switch to this newly prepared filter. Changing the active filter is guaranteed to only take place at the beginning of a filter computation and is therefore save.

# Pipelines

## 5x Configuration

Y−mn = X−m(n+1) with m = 0 or 1 and n = 0, 1, … max sos.

Address sssssssh: sssssss is SOS numbering from 0 to 127; h=1 is −2 history, this bit is inverted every other cycle

Control: v is write result value, h is history write, r is accumulator reset, l is accumulator load for shift,

 m is 1-save sum2 to history/0-save accumulator to history, and i is 0-input/1-history.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | History | InputMux | Coefficients | Multiplier | Sum1 | Sum2 | Accumulator | Control |
| Addr | Mem | Mem | Reg | vhrl\_0m0i |
| -3 | 0b0000000\_0 | — | — | S1 | — | — | a12\*Y−12 | a22\*Y−22 | c02 ΣX | 0b0000\_0001 |
| -2 | 0b0000000\_1 | X−20 | — | b20 | — | — | — | a12\*Y−12 | c02 ΣX+a22\*Y−22 | 0b0000\_0001 |
| -1 | 0b0000000\_0 | X−10 | X−20 | b10 | b20 | — | — | — | Y02 = c04 ΣX+ ΣY+… → next Y−22 | 0b1100\_0001 |
| 0 | 0b0000001\_1 | — | X−10 | g | b10 | b20\*X−20 | — | — | — | 0b0000\_0000 |
| 1 | 0b0000001\_0 | Y−20 | X00 | a20/c00 | g | b10\*X−10 | b20\*X−20 | — | — | 0b0010\_0001 |
| 2 | 0b0000001\_1 | Y−10 | Y−20 | a10 | a20/c00 | g\*X00 | b10\*X−10 | b20\*X−20 | 0 | 0b0000\_0001 |
| 3 | 0b0000001\_0 | X−21 | Y−10 | b21 | a10 | a20\*Y−20 | g\*X00  | b10\*X−10 | b20\*X−20 | 0b0000\_0001 |
| 4 | 0b0000010\_1 | X−11 | X−21 | b11 | b21 | a10\*Y−10 | a20\*Y−20 | g\*X00 | b20\*X−20+b10\*X−10 | 0b0100\_0101 |
| 5 | 0b0000010\_0 | Y−21 | X−11 | a21/c01 | b11 | b21\*X−21 | a10\*Y−10 | a20\*Y−20 | b20\*X−20+b10\*X−10+ g\*X00 | 0b0001\_0001 |
| 6 | 0b0000010\_1 | Y−11 | Y−21 | a11 | a21/c01 | b11\*X−11 | b21\*X−21 | a10\*Y−10 | c00 ΣX+a20\*Y−20 | 0b0000\_0001 |
| 7 | 0b0000010\_0 | X−22 | Y−11 | b22 | a11 | a21\*Y−21 | b11\*X−11 | b21\*X−21 | Y00 = X01 = c00 ΣX+ ΣY → next X−21 | 0b0100\_0001 |
| 8 | 0b0000011\_1 | X−12 | X−22 | b12 | b22 | a11\*Y−11 | a21\*Y−21 | b11\*X−11 | Y00+ b21\*X−21 | 0b0000\_0001 |
| 9 | 0b0000011\_0 | Y−22 | X−12 | a22/c02 | b12 | b22\*X−22 | a11\*Y−11 | a21\*Y−21 | Y00+ b21\*X−21+b10\*X−10 | 0b0001\_0001 |
| 10 | 0b0000011\_1 | Y−12 | Y−22 | a12 | a22/c02 | b12\*X−12 | b22\*X−22 | a11\*Y−11 | c01 ΣX+a21\*Y−21 | 0b0000\_0001 |
| 11 | 0b0000011\_0 |  | Y−12 | S0 | a12 | a22\*Y−22 | b12\*X−12 | b22\*X−22 | Y01 = c01 ΣX+ ΣY+… → next Y−21 | 0b0100\_0001 |
|  |  |  |  |  |  |  |  |  |  |  |
| 12 | — | — | — | S1 | — | a12\*Y−12 | a22\*Y−22 | b12\*X−12 | Y01+b22\*X−22 | 0b0000\_0001 |
| 13 | 0b0000000\_0 | — | — | 0 | — | — | a12\*Y−12 | a22\*Y−22 | Y01+b22\*X−22+b12\*X−12 | 0b0001\_0001 |
| 14 | 0b0000000\_1 | X−20 | — | b20 | — | — | — | a12\*Y−12 | c02 ΣX+a22\*Y−22 | 0b0000\_0001 |
| 15 | — | X−10 | X−20 | b10 | b20 | — | — | — | Y02 = c04 ΣX+ ΣY+… → next Y−22 | 0b1100\_0001 |

## 2x Configuration

Y−mn = X−m(n+1) with m = 0 or 1 and n = 0, 1, … 3.

Address sssssbbh: h=1 is −2 history, sssss is SOS numbering, bb are lsb(00), isb(01), msb(10).

Control: v is write result value, h is history write, r is accumulator reset, l is accumulator load,

 mm is 00-no shift/01-17b shift/1X-34b shift, and ii is 00-input(lsb)/01-input(isb)/10-input(msb)/11-history.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # | History | InputMux | Coefficients | Multiplier | Shifter | Accumulator | Control |
| Addr | Mem | Mem | Reg | vhrl\_mmii |
| 0 | — | — | — | g | — | — | — | — | 0b0000\_0000 |
| 1 | — | — | X00,l | g | g | — | — | — | 0b0000\_0001 |
| 2 | 0b00000\_001 | — | X00,i | g | g | g\*X00,l | — | — | 0b0010\_0010 |
| 3 | 0b00000\_011 | X−20,l | X00,m | b20 | g | g\*X00,i | g\*X00,l | 0 | 0b0000\_0111 |
| 4 | 0b00000\_101 | X−20,i | X−20,l | b20 | b20 | g\*X00,m | g\*X00,i | g\*X00,l | 0b0000\_1011 |
| 5 | 0b00000\_000 | X−20,m | X−20,i | b20 | b20 | b20\*X−20,l | g\*X00,m | g\*X00,l+ g\*X00,i | 0b0000\_0011 |
| 6 | 0b00000\_010 | X−10,l | X−20,m | b10 | b20 | b20\*X−20,i | b20\*X−20,l | g\*X00 → next X−20 | 0b0100\_0111 |
| 7 | 0b00000\_100 | X−10,i | X−10,l | b10 | b10 | b20\*X−20,m | b20\*X−20,i | … | 0b0000\_1011 |
| 8 | 0b00001\_001 | X−10,m | X−10,i | b10 | b10 | b10\*X−10,l | b20\*X−20,m |  | 0b0000\_0011 |
| 9 | 0b00001\_011 | Y−20,l | X−10,m | a20/c00 | b10 | b10\*X−10,i | b10\*X−10,l | g\*X00+b20\*X−20 | 0b0000\_0111 |
| 10 | 0b00001\_101 | Y−20,i | Y−20,l | a20/c00 | a20/c00 | b10\*X−10,m | b10\*X−10,i | … | 0b0000\_1011 |
| 11 | 0b00001\_000 | Y−20,m | Y−20,i | a20/c00 | a20/c00 | a20\*Y−20,l | b10\*X−10,m |  | 0b0000\_0011 |
| 12 | 0b00001\_010 | Y−10,l | Y−20,m | a10 | a20/c00 | a20\*Y−20,i | a20\*Y−20,l | g\*X00+b20\*X−20+b10\*X−10 | 0b0001\_0111 |
| 13 | 0b00001\_100 | Y−10,i | Y−10,l | a10 | a10 | a20\*Y−20,m | a20\*Y−20,i | c00 ΣX+ a20\*Y−20,l | 0b0000\_1011 |
| 14 | 0b00001\_001 | Y−10,m | Y−10,i | a10 | a10 | a10\*Y−10,l | a20\*Y−20,m | … | 0b0000\_0011 |
| 15 | 0b00001\_011 | X−21,l | Y−10,m | b21 | a10 | a10\*Y−10,i | a10\*Y−10,l |  | 0b0000\_0111 |
| 16 | 0b00001\_101 | X−21,i | X−21,l | b21 | b21 | a10\*Y−10,m | a10\*Y−10,i |  | 0b0000\_1011 |
| 17 | 0b00001\_000 | X−21,m | X−21,i | b21 | b21 | b21\*X−21,l | a10\*Y−10,m |  | 0b0000\_0011 |
| 18 | 0b00001\_010 | X−11,l | X−21,m | b11 | b21 | b21\*X−21,i | b21\*X−21,l | Y00 = X01 = c00 ΣX+ ΣY → next X−21 | 0b0100\_0111 |
| 19 | 0b00001\_100 | X−11,i | X−11,l | b11 | b11 | b21\*X−21,m | b21\*X−21,i | … | 0b0000\_1011 |
| 20 | 0b00010\_001 | X−11,m | X−11,i | b11 | b11 | b11\*X−11,l | b21\*X−21,m |  | 0b0000\_0011 |
| 21 | 0b00010\_011 | Y−21,l | X−11,m | a21/c01 | b11 | b11\*X−11,i | b11\*X−11,l |  | 0b0000\_0111 |
| 22 | 0b00010\_101 | Y−21,i | Y−21,l | a21/c01 | a21/c01 | b11\*X−11,m | b11\*X−11,i |  | 0b0000\_1011 |
| … |  |  |  |  |  |  |  |  |  |
| 49 | 0b00100\_100 | Y−13,i | Y−13,l | a13 | a13 | a23\*Y−23,m | a23\*Y−23,i |  | 0b0000\_1011 |
| 50 | — | Y−13,m | Y−13,i | a13 | a13 | a13\*Y−13,l | a23\*Y−23,m |  | 0b0000\_0011 |
| 51 | — | — | Y−13,m | — | a13 | a13\*Y−13,i | a13\*Y−13,l |  | 0b0000\_0111 |
| 52 | — | — | — | — | — | a13\*Y−13,m | a13\*Y−13,i |  | 0b0000\_1011 |
| 53 | — | — | — | — | — | — | a13\*Y−13,m |  | 0b0000\_0011 |
| 54 | 0b00100\_010 | — | — | — | — | — | — | Y03 = c03 ΣX+ ΣY+… → next Y−23 | 0b1100\_0000 |
| 59 | — | — | — | — | — | — | — |  | 0b0000\_0000 |
| … |  |  |  |  |  |  |  |  |  |
| 62 | — | — | — | — | — | — | — |  | 0b0000\_0000 |
| 63 | — | — | — | — | — | — | — |  | 0b0000\_0000 |

## 5x Configuration with SOS Switches Enabled

The switches are off for the first SOS and third SOS:

* The overall gain will be replaced by 0, if the gain switch is set to off.
* For the first SOS an alternate gain, g1, has to be supplied instead of a20. This value must be the gain in the first SOS minus 1, times the overall gain.
* For the third SOS an alternate gain, g2, has to be supplied instead of a12. This value must or the gain in the third SOS minus 1. This is the same for any other SOS.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | History | InputMux | Coefficients | Multiplier | Sum1 | Sum2 | Accumulator | Control |
| Addr | Mem | Mem | Reg | vhrl\_bm0i |
| -3 | 0b0000000\_0 | — | — | S1 | — | — | a12\*Y−12 | a22\*Y−22 | c02 ΣX | 0b0000\_0001 |
| -2 | 0b0000000\_1 | X−20 | — | b20 = 0 | — | — | — | a12\*Y−12 | c02 ΣX+a22\*Y−22 | 0b0000\_0001 |
| -1 | 0b0000000\_0 | X−10 | X−20 | b10 = 0 | 0 | — | — | — | Y02 = c04 ΣX+ΣY+… → next Y−22 | 0b1100\_0001 |
| 0 | 0b0000001\_1 | — | X−10 | g | 0 | 0\*X−20 | — | — | — | 0b0000\_0000 |
| 1 | 0b0000001\_0 | Y−20 | X00 | g1 | g | 0\*X−10 | 0 | — | — | 0b0010\_0000 |
| 2 | 0b0000001\_1 | Y−10 | X00 | a10= 0 | g1 | g\*X00 | 0 | 0 | 0 | 0b0000\_0001 |
| 3 | 0b0000001\_0 | X−21 | Y−10 | b21 | 0 | g1\*X00 | g\*X00  | 0 | 0 | 0b0000\_0001 |
| 4 | 0b0000010\_1 | X−11 | X−21 | b11 | b21 | 0\*Y−10 | g1\*X00 | g\*X00 | 0 / g\*X00 → next X−20 | 0b0100\_0101 |
| 5 | 0b0000010\_0 | Y−21 | X−11 | a21/c01 | b11 | b21\*X−21 | 0 | g1\*X00 | g\*X00 | 0b0001\_0001 |
| 6 | 0b0000010\_1 | Y−11 | Y−21 | a11 | a21/c01 | b11\*X−11 | b21\*X−21 | 0 | g\*X00+g1\*X00 | 0b0000\_0001 |
| 7 | 0b0000010\_0 | X−22 | Y−11 | b22 = 0 | a11 | a21\*Y−21 | b11\*X−11 | b21\*X−21 | Y00 = X01 = (g+g1)\*X00 → next X−21 | 0b0100\_0001 |
| 8 | 0b0000011\_1 | X−12 | X−22 | b12 = 0 | 0 | a11\*Y−11 | a21\*Y−21 | b11\*X−11 | Y00+b21\*X−21 | 0b0000\_0001 |
| 9 | 0b0000011\_0 | Y−22 | X−12 | a22/c02 | 0 | 0\*X−22 | a11\*Y−11 | a21\*Y−21 | Y00+b21\*X−21+b10\*X−10 | 0b0001\_0001 |
| 10 | 0b0000011\_1 | Y−12 | Y−22 | g2 | 0 | 0\*X−12 | 0 | a11\*Y−11 | c01 ΣX+a21\*Y−21 | 0b0000\_0001 |
| 11 | 0b0000011\_0 |  | Y−12 | S0 | g2 | 0\*Y−22 | 0 | 0 | Y01 = c01 ΣX+ΣY+… → next Y−21 | 0b0100\_1001 |
|  |  |  |  |  |  |  |  |  |  |  |
| 12 | — | — | — | S1 | — | g2\*Y01 | 0 | 0 | Y01 | 0b0000\_0001 |
| 13 | 0b0000000\_0 | — | — | 0 | — | — | g2\*Y01 | 0 | Y01 | 0b0001\_0001 |
| 14 | 0b0000000\_1 | X−20 | — | b20 | — | — | — | g2\*Y01 | Y01 | 0b0000\_0001 |
| 15 | — | X−10 | X−20 | b10 | b20 | — | — | — | Y02 = (1+g2)Y01 → next Y−22 | 0b1100\_0001 |

1. D. Sigg, “Implementing an IIR Filter in Hardware,” [LIGO-T050060](https://dcc.ligo.org/LIGO-T050060). [↑](#footnote-ref-1)
2. A.V. Oppenheim & R.W. Schafer, “Discrete-Time Signal Processing”. [↑](#footnote-ref-2)
3. <https://svn.ligo.caltech.edu/svn/Altium-D2D/Software/FPGA-VHDL/PCIeTiming/IIRFilter/CoeffGen/CoeffGen52_2x> [↑](#footnote-ref-3)