Technical notes on using Analog Devices products and development tools
Visit our Web resources http://www.analog.com/ee-notes and http://www.analog.com/processors or e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

Using ADSP-2156x High Performance FIR/IIR Accelerators

Contributed by Sanket Nayak and Mitesh Moonat *Rev 2 – August 14, 2019*

Introduction

ADSP-2156x processors incorporate high-performance hardware accelerators dedicated to performing two widely-used signal processing operations: FIR (Finite Impulse Response) and IIR (Infinite Impulse response), hereby referred to as FIRA and IIRA, respectively. As illustrated in Figure 1, these accelerators can be run in tandem with the core to perform dedicated fixed-function computations (FIR and IIR), thereby increasing the overall processing capability of the processor.

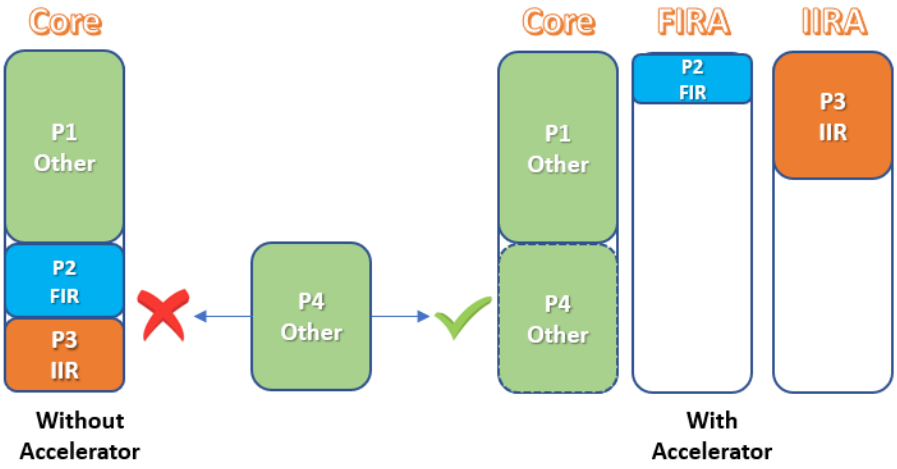


Figure 1: Processing Capability with and without FIR/IIR Accelerators

This application note provides a brief overview of the functional and performance improvements achieved with the ADSP-2156x FIR/IIR accelerators (hereby referred to as accelerators) when compared with their predecessors. The note discusses the following topics related to:

- Accelerator performance for different configurations and contributing factors
- Accelerator CrossCore® Embedded Studio (CCES) drivers, their usage, examples and driver benchmark details
- Different Accelerator usage models for optimum performance in various application scenarios

For more details about the accelerator architecture and programming model, refer to the *ADSP-2156x SHARC+ Processor Hardware Reference* [2].

ADSP-2156x Accelerator Enhancements

Accelerator enhancements include performance and functional improvements.

Performance Improvements

- The computation speed is increased to the core clock (CCLK) frequency, as compared with the SCLK frequency, which was a maximum of CCLK/4 on the ADSP-SC57x processors. This performance was made possible by replacing the MAC unit of the accelerator with the one compatible with the SHARC+ core.
- Data and MMR latency is reduced with a closer integration of the SHARC+ core and accelerator. The accelerator master ports can directly access SHARC L1 memory with reduced latency without going through the system fabric. The core can also access the accelerator MMR registers with reduced latency.

Functional Improvements

The ADSP-2156x processor has some additional features included with the accelerator. These features are available in a new mode of operation known as the Auto Configuration Mode (ACM). The legacy mode of operation is intact, and the accelerator can be switched to ACM using the `CTL1.ACM` bit. The following additional features are available in ACM:

Channel Specific Parameter Selection

ACM allows some Global Control (`CTL1`) register fields to be made channel-specific (e.g, to be loaded as part of the TCB in the Secondary Global Control register (`SGCTL`)). These fields include Output Rounding Mode (`RND`), Fixed Point Select (`FXD`) and Two's Complement Format (`TC`) in the FIRA and Output Rounding Mode (`RND`), Forty-bit (`FORTYBIT`), Save State Enable (`SS`), and Save State Completion Interrupt Deselect (`SSESEL`) bits in the IIRA. In legacy mode, these bits globally configure all channels.

No Channel Number Limitation

In legacy mode, the number of channels to be processed by the accelerator is programmed in the `CTL1.NCH` bit field. This limits the number of FIR/IIR channels that can be processed in one iteration. In ACM, the Accelerator processes the channels/TCBs until it sees a channel/TCB with its Chain Pointer (CHNPTR) value as zero. Hence, there is no theoretical limit on the number of channels that the accelerator can process in ACM.

Halt Feature

In ACM, when the accelerator is processing a set of TCBs, the core can halt the accelerator TCB processing at any stage using the `CTL1.HALT` bit and dynamically insert new TCBs for processing by the accelerator. The advantage of this mechanism is that if a request for processing of a set of TCBs arrives when the accelerator is already processing another set of TCBs, the core can dynamically halt the accelerator, insert the new TCB list, and resume the processing. In the absence of this feature, the core must maintain the newly requested TCB list in software, wait for completion of the processing of the already existing TCB list, and then restart the accelerator for processing the new TCB list.

Selective Interrupt Generation

In legacy mode, an interrupt can be generated either after processing the entire TCB list or after the processing of each TCB using the `CTL1.CCINTR` bit. In ACM, interrupt generation is customizable per channel using the `CTL2.IMASK` bit which is loaded as part of TCB.

Trigger Generation and Trigger Wait

In ACM, each channel can be configured to generate a trigger after its completion using the `CTL2.TMASK` bit. This trigger can be used as a master trigger and can be routed to a trigger slave using the Trigger Routing Unit (TRU). Each channel can be configured to wait for a master trigger after loading the TCB using the `CTL2.TWAIT` bit. In legacy mode, trigger support is not available.

Accelerator Performance

The accelerator's processing cycles consist of two components: DMA cycles and Compute (MAC) cycles.

DMA Cycles

The contribution of DMA cycles in the total processing cycles depends on the following factors.

Filter Parameters

The DMA cycles consist of two operations:

- Initial preload of the coefficients and delay line (FIRA) or state variables (IIRA) - for IIRA, the coefficient load and state variable load can optionally be skipped in legacy mode to save these cycles. The amount of data to be transferred for this operation depends on the **Tap Length** (FIRA) or the **Number of Biquads** (IIRA).
- Fetch/Store of the Input/Output buffers - the amount of data to be transferred for this operation depends on the **Window Size**. However, these operations take place in parallel to compute operation and therefore do not add any overhead for larger tap length or number of biquads. For smaller tap length and biquad values, these cycles may dominate the compute cycles. The crossover point (N) in terms of tap length (FIRA) and biquads (IIRA) depends on the memory access latency. The value of N increases as buffers are moved from L1 to L2 memory and then to L3 memory.

Buffer Placement

The DMA cycles also depend on whether the input, output, and coefficient buffers are placed in L1/L2/L3 memory, as each memory type has different access latencies.

Compute (MAC) Cycles

Unlike in previous SHARC processors, the MAC units used on the ADSP-2156x accelerators are compatible with the SHARC+ core MAC unit. Therefore, the multiplication operation takes place in two stages. This design implies that at least N+1 number of cycles are required to perform N MAC operations. For the FIRA, the tap length size is generally significantly higher; therefore, the theoretical number of cycles per sample per tap is close to 0.25 (1/4). For the IIRA, processing each biquad requires 6 cycles (for 5 MAC operations).

In an ideal scenario, to make maximum utilization of the computing power of the accelerator, the percent contribution of the compute/MAC cycles, called the **Compute Efficiency (CE)**, must be 100%. CE can be measured by the following expression:

***CE** = (M/N)*100*

Where,

M = Theoretical Cycles per Tap (FIRA = 0.25) or per biquad (IIRA=6)

N = Measured Cycles per Tap (FIRA) or per biquad (IIRA)

Measured FIR/IIR Performance

This section provides FIR/IIR performance measurements for the ADSP-2156x accelerators for different filter parameters. For comparison, the measured performance for the following cases with varying core clock (CCLK) and accelerator clock (ACLK) frequencies is also provided:

1. ADSP-2156x FIRA/IIRA @ maximum ACLK (=CCLK) = 1 GHz
2. ADSP-SC57x FIRA/IIRA @ maximum ACLK(=SCLK0) = 125 MHz and @ maximum CCLK = 500 MHz
3. ADSP-214xx FIRA/IIRA @ maximum ACLK(=PCLK=CCLK/2) = 225 MHz and @ maximum CCLK = 450 MHz
4. ADSP-2156x (SHARC+) core @ maximum CCLK = 1 GHz

Although the ADSP-SC58x accelerator performance numbers are not provided here, the performance is similar to ADSP-SC57x except for a slight performance loss due to the absence of DMA burst mode.

The numbers are provided both in core cycles and in absolute time (µs) units. All the buffers are placed in L1 memory for these measurements. Core FIR/IIR cycles are measured with the help of the optimized library functions available with CCES.

Additional measurement and analysis are provided to discuss how the various factors affect the MAC Utilization Efficiency (MUE) of the accelerator.

FIR Performance

This section contains the following information:

- Measured performance of the ADSP-2156x FIRA for different filter parameters and comparison with other cases
- Measured FIRA Compute Efficiency (CE) and contributing factors
- Effect of buffer placement (L1/L2/L3) on FIRA performance

Comparison with ADSP-SC57x Accelerator, ADSP-2156x Core, and ADSP-214xx Accelerator

Figure 2 and Figure 3 show the measured FIR performance in core cycles and time units (µs), respectively, for the five cases. Figure 4 shows the performance factor of the ADSP-2156x FIR accelerator as compared to the other cases for different window size values and tap length of 512.

The performance numbers shown in Figure 2 correspond to the selected number of combinations of window size and tap length. For any other combinations, the following example code provided with this application note[4] can be used to measure the FIR performance on the evaluation boards for the four cases:

1. ADSP_2156x_FIRA_Performance
2. ADSP_SC57x_FIRA_Performance
3. ADSP_214xx_FIRA_Performance
4. ADSP_2156x_FIR_Core_Performance

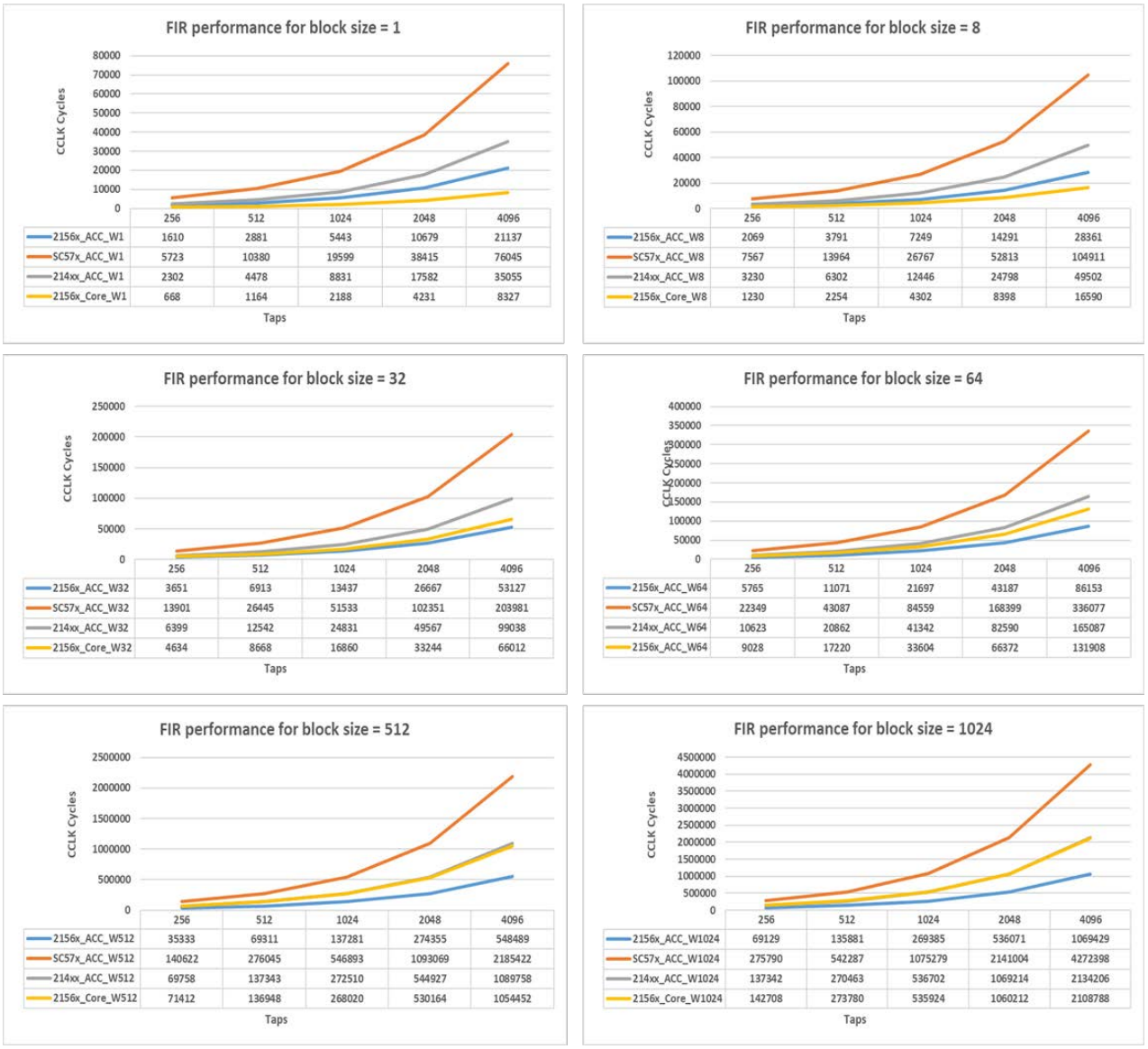


Figure 2: FIR Performance in Core Cycles Across Window Size and Tap Length

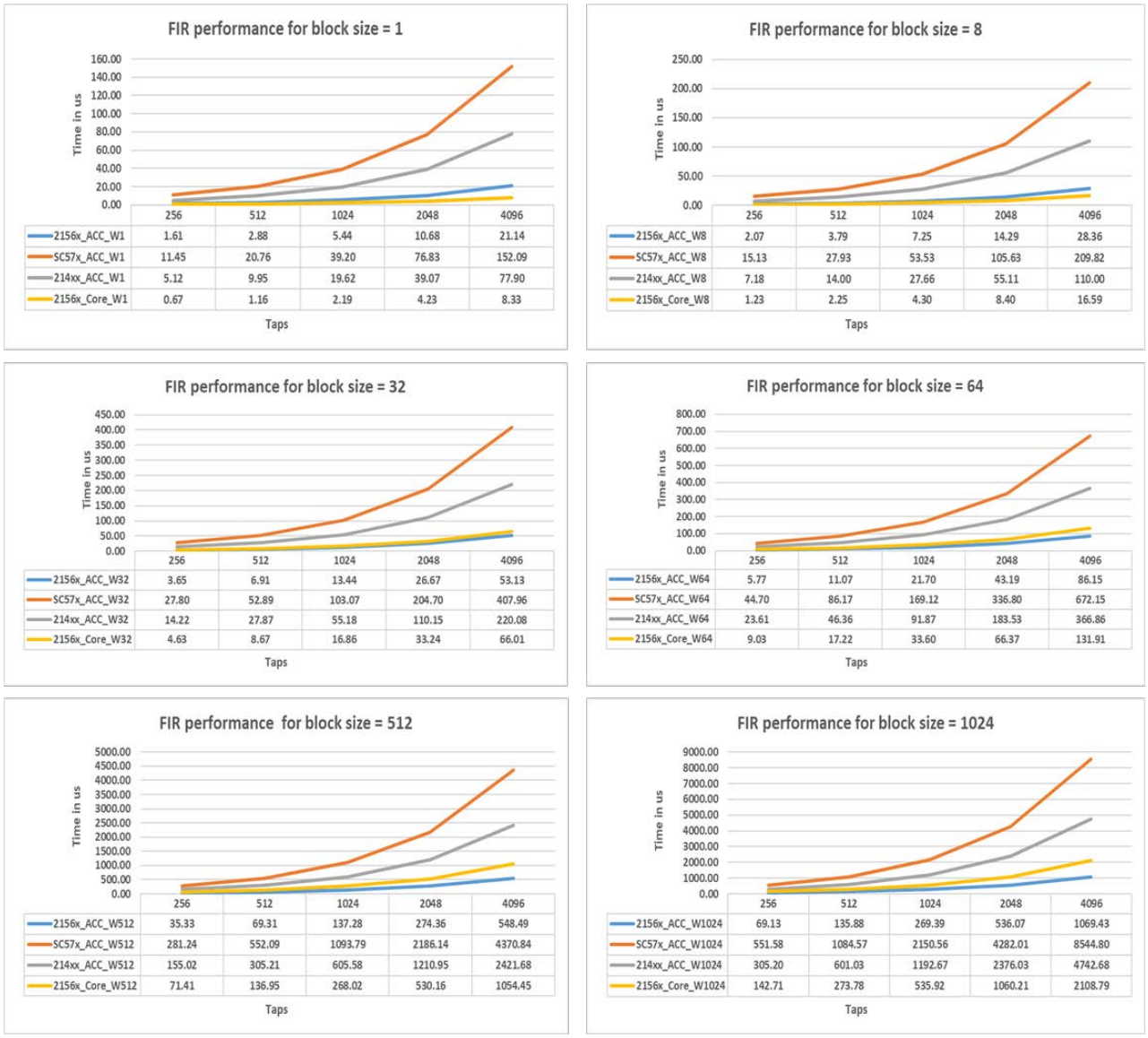


Figure 3: FIR Performance in Time (μs) Across Window Size and Tap Length

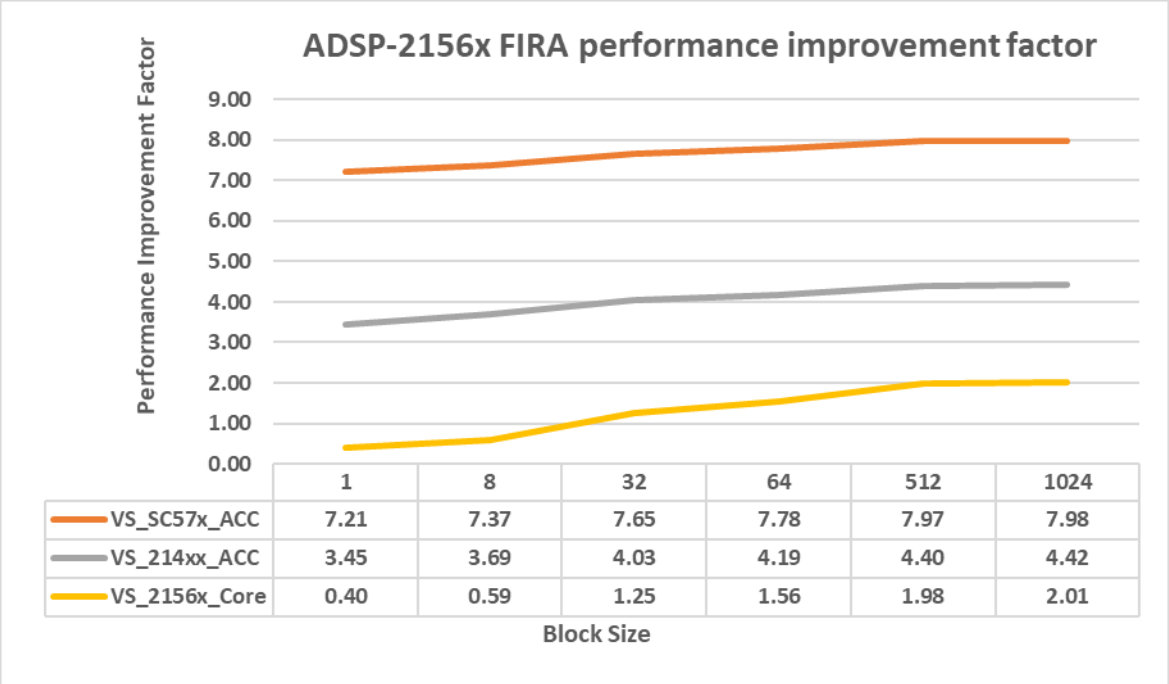


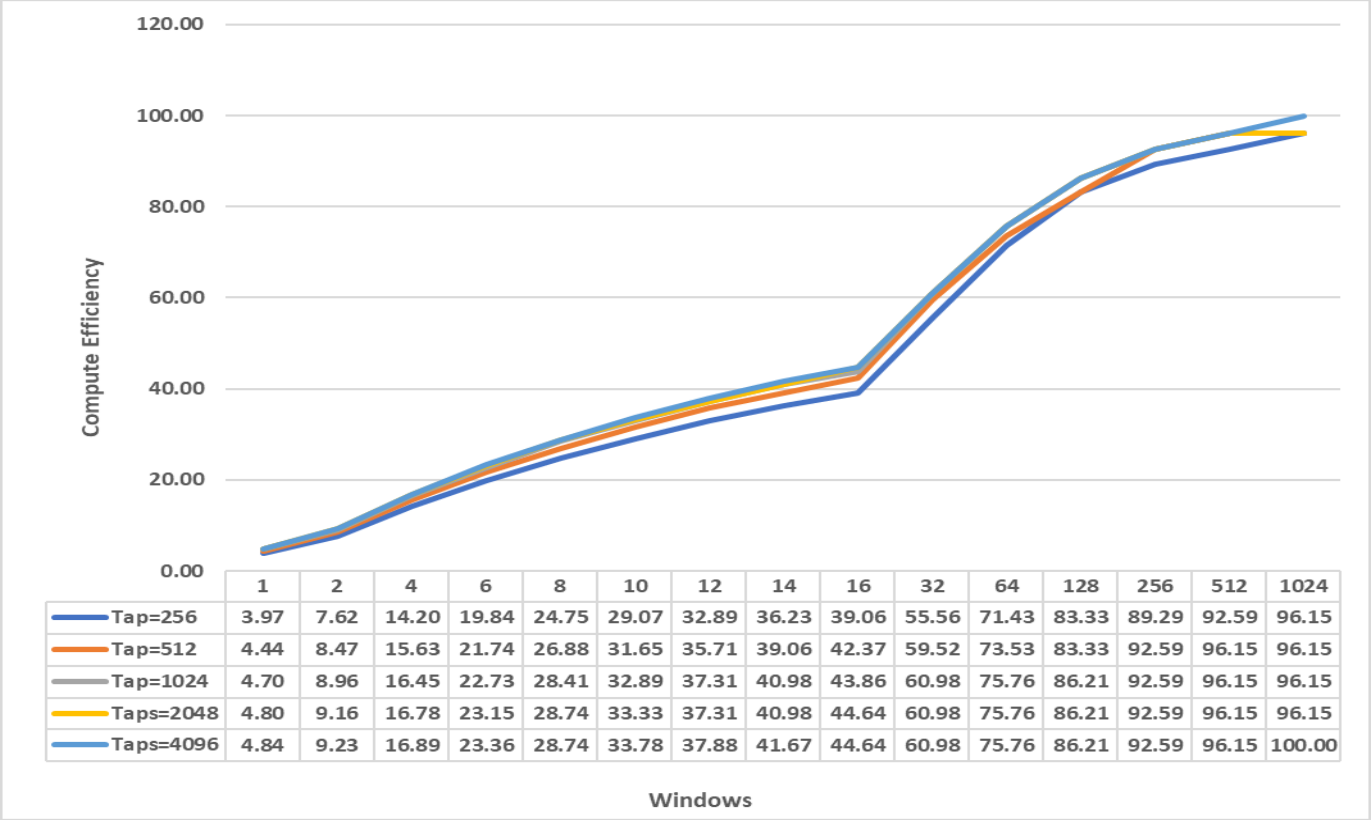
Figure 4: ADSP-2156x FIR Performance Improvement Factor for 512 Taps Considering Time

The following observations relate to the FIR performance numbers shown in Figure 4:

- **The ADSP-2156x FIRA performance is around 8x the ADSP-SC57x FIRA**. This increase is because the ADSP-2156x FIRA runs at CCLK (1 GHz), whereas the ADSP-SC57x FIRA runs at SCLK, which is a maximum of CCLK/4 (125 MHz).
- **The ADSP-2156x FIRA performance is around 4.44x the ADSP-214xx FIRA performance.** This increase is because the ADSP-2156x FIRA runs at CCLK (1 GHz), whereas the ADSP-214xx FIRA runs at PCLK, which is a maximum of CCLK/2 (225 MHz).
- **The ADSP-2156x FIRA performance is around 2x the ADSP-2156x SHARC+ core's execution of FIR code**. This is because the ADSP-2156x FIRA has four MAC units, whereas the SHARC+ core featured on ADSP-2156x processors has two MAC units. An exception to this conclusion applies to smaller block sizes, where the DMA overhead dominates the compute cycles. The block size crossover point (in number of samples) where the ADSP-2156x FIRA performs better than the SHARC+ core is around 24.

Compute Efficiency (CE)

Figure 5 shows how the CE percentage for the FIRA varies with window size and tap length. As shown in the plot, the CE is very close to 100% for larger window size values. It significantly reduces for very small window size values. For a constant window size, the CE percentage does not change much for different tap length values.

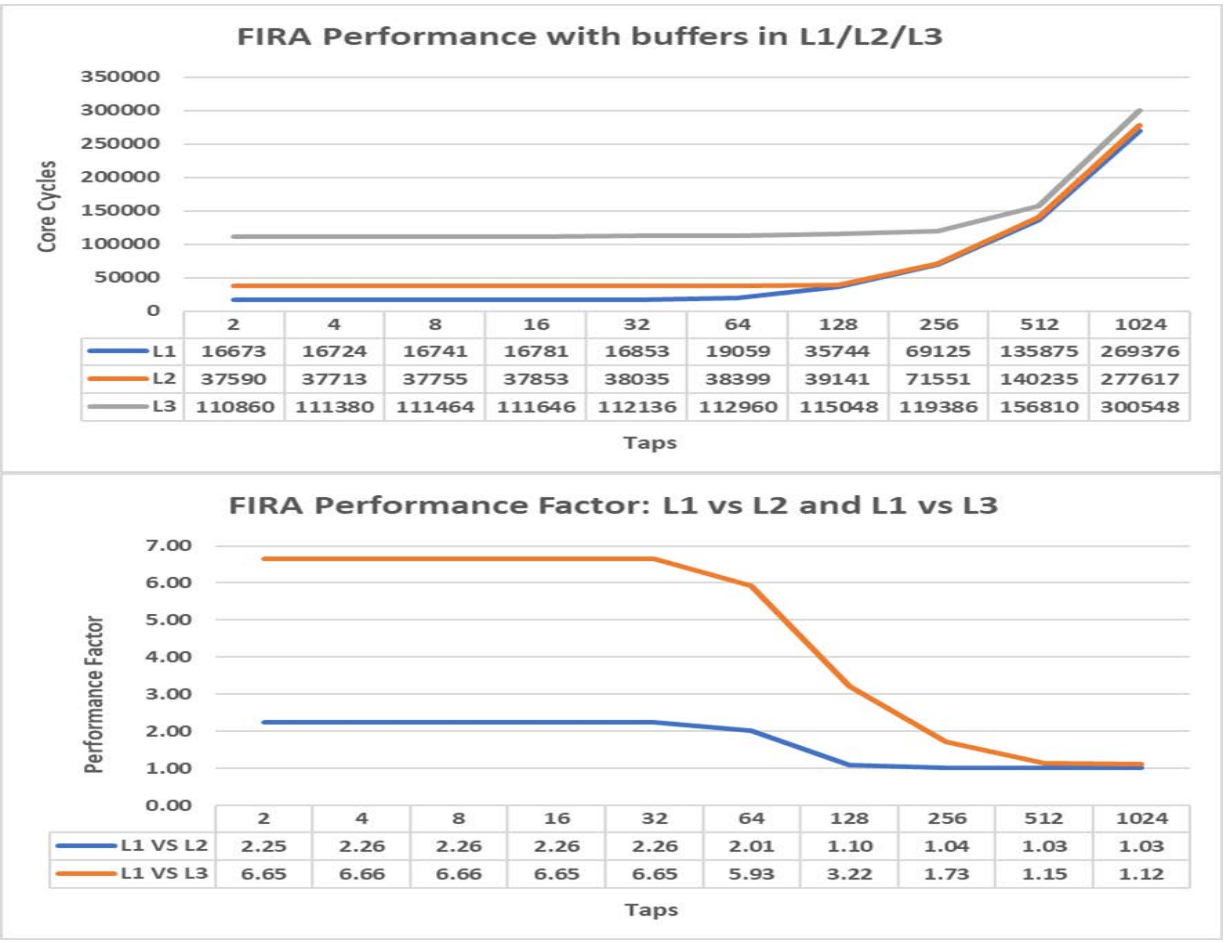


	1	2	4	6	8	10	12	14	16	32	64	128	256	512	1024
Tap=256	3.97	7.62	14.20	19.84	24.75	29.07	32.89	36.23	39.06	55.56	71.43	83.33	89.29	92.59	96.15
Tap=512	4.44	8.47	15.63	21.74	26.88	31.65	35.71	39.06	42.37	59.52	73.53	83.33	92.59	96.15	96.15
Tap=1024	4.70	8.96	16.45	22.73	28.41	32.89	37.31	40.98	43.86	60.98	75.76	86.21	92.59	96.15	96.15
Taps=2048	4.80	9.16	16.78	23.15	28.74	33.33	37.31	40.98	44.64	60.98	75.76	86.21	92.59	96.15	96.15
Taps=4096	4.84	9.23	16.89	23.36	28.74	33.78	37.88	41.67	44.64	60.98	75.76	86.21	92.59	96.15	100.00

Figure 5: FIRA Compute Efficiency (CE) for Different Filter Parameters

Buffer Placement

Figure 6 shows how the core cycles change with increased tap length when the input/output/coefficient buffers are placed in L1/L2/L3 memory for a window size of 1024. It also shows a plot of the performance factor when the buffers are in L1 memory versus in L2/L3 memory.



	2	4	8	16	32	64	128	256	512	1024
L1	16673	16724	16741	16781	16853	19059	35744	69125	135875	269376
L2	37590	37713	37755	37853	38035	38399	39141	71551	140235	277617
L3	110860	111380	111464	111646	112136	112960	115048	119386	156810	300548

	2	4	8	16	32	64	128	256	512	1024
L1 VS L2	2.25	2.26	2.26	2.26	2.26	2.01	1.10	1.04	1.03	1.03
L1 VS L3	6.65	6.66	6.66	6.65	6.65	5.93	3.22	1.73	1.15	1.12

Figure 6: ADSP-2156x FIRA Performance Comparison for Buffers in L1/L2/L3 Memory

The following important observations can be noted from the plots:

- As the tap length is increased for a constant window size, the core cycles remain constant until a certain point (N) and then increase proportionally. For lower tap length values, the input/output DMA cycles dominate the compute cycles.
- As the memory access latency increases from L1 to L2 memory and then further to L3 memory, the value of N increases, and the performance factor trends towards 1.0. Placing buffers in L2 and L3 memory instead of L1 memory might be expensive for tap lengths less than N and comparable for tap lengths greater than or equal to N.

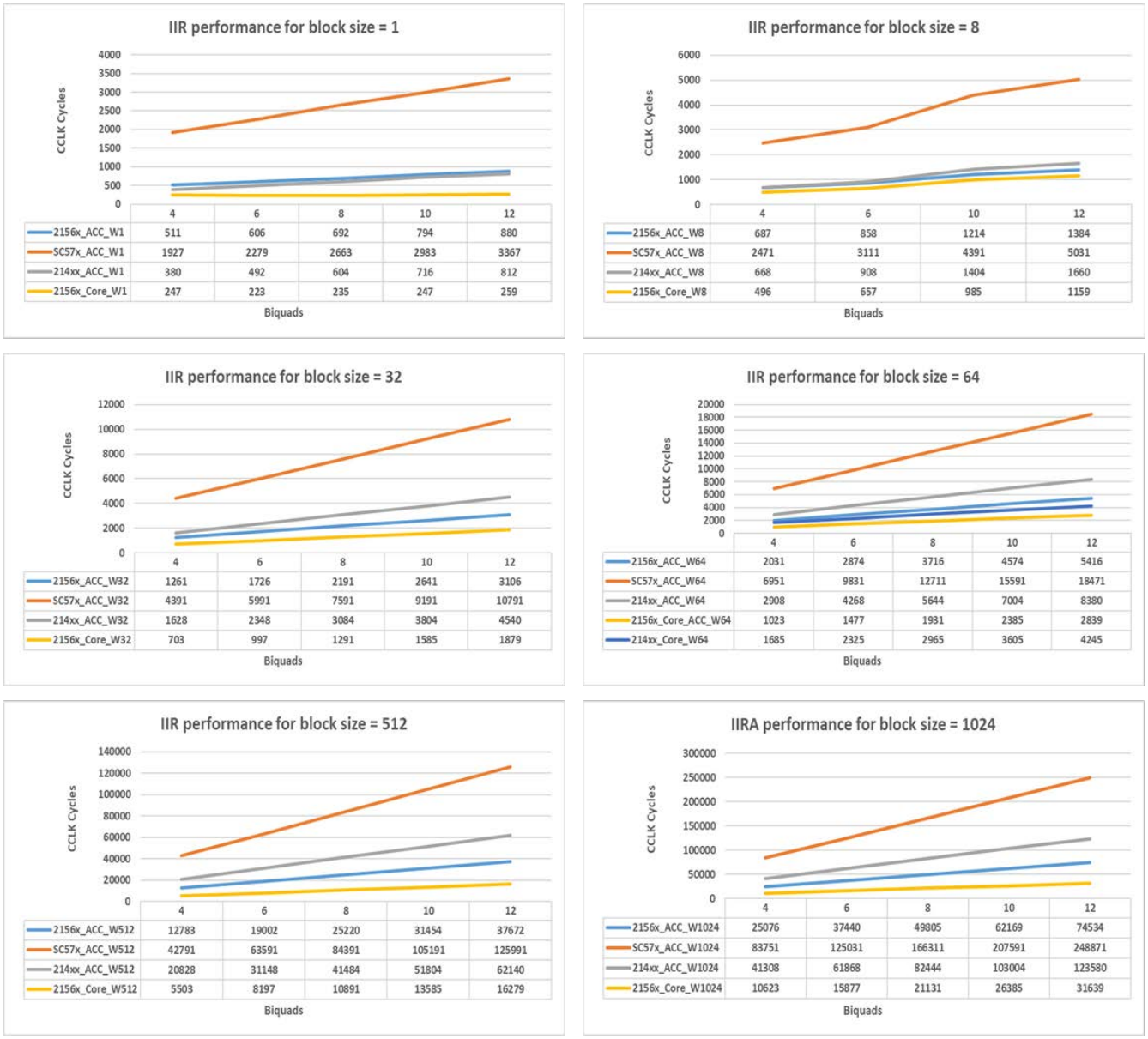
IIR Performance

This section contains the following information:

- Measured performance of the ADSP-2156x IIRA for different filter parameters and comparison with other cases
- Measured IIRA Compute Efficiency (CE) and contributing factors
- Effect of buffer placement (L1/L2/L3) on IIRA performance

Comparison with ADSP-SC57x Accelerator, ADSP-2156x Core, and ADSP-214xx Accelerator

Figure 7 and Figure 8 show the measured IIR performance in core cycles and time units (µs), respectively, for all the cases. Figure 9 shows the performance factor of the ADSP-2156x IIRA as compared to the other cases for different window size values and 6 biquads.



IIR performance for block size = 1	4	6	8	10	12
2156x_ACC_W1	511	606	692	794	880
SC57x_ACC_W1	1927	2279	2663	2983	3367
214xx_ACC_W1	380	492	604	716	812
2156x_Core_W1	247	223	235	247	259

IIR performance for block size = 8	4	6	10	12
2156x_ACC_W8	687	858	1214	1384
SC57x_ACC_W8	2471	3111	4391	5031
214xx_ACC_W8	668	908	1404	1660
2156x_Core_W8	496	657	985	1159

IIR performance for block size = 32	4	6	8	10	12
2156x_ACC_W32	1261	1726	2191	2641	3106
SC57x_ACC_W32	4391	5991	7591	9191	10791
214xx_ACC_W32	1628	2348	3084	3804	4540
2156x_Core_W32	703	997	1291	1585	1879

IIR performance for block size = 64	4	6	8	10	12
2156x_ACC_W64	2031	2874	3716	4574	5416
SC57x_ACC_W64	6951	9831	12711	15591	18471
214xx_ACC_W64	2908	4268	5644	7004	8380
2156x_Core_ACC_W64	1023	1477	1931	2385	2839
214xx_Core_W64	1685	2325	2965	3605	4245

IIR performance for block size = 512	4	6	8	10	12
2156x_ACC_W512	12783	19002	25220	31454	37672
SC57x_ACC_W512	42791	63591	84391	105191	125991
214xx_ACC_W512	20828	31148	41484	51804	62140
2156x_Core_W512	5503	8197	10891	13585	16279

IIRA performance for block size = 1024	4	6	8	10	12
2156x_ACC_W1024	25076	37440	49805	62169	74534
SC57x_ACC_W1024	83751	125031	166311	207591	248871
214xx_ACC_W1024	41308	61868	82444	103004	123580
2156x_Core_W1024	10623	15877	21131	26385	31639

Figure 7: IIR Performance in Core Cycles Across Window Size and Biquad Values

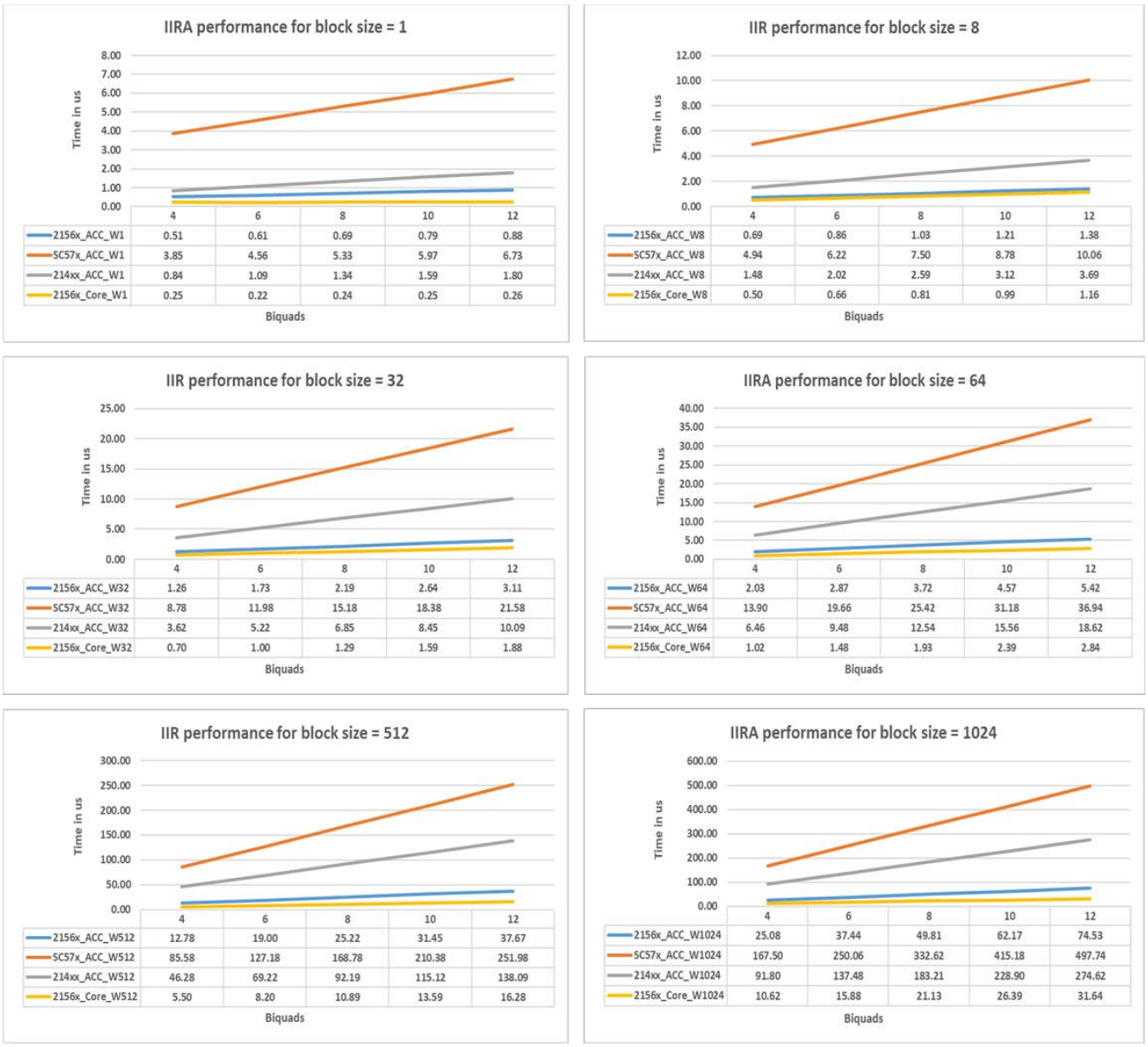


Figure 8: IIR Performance in Time (μs) Across Window Size and Biquad Values

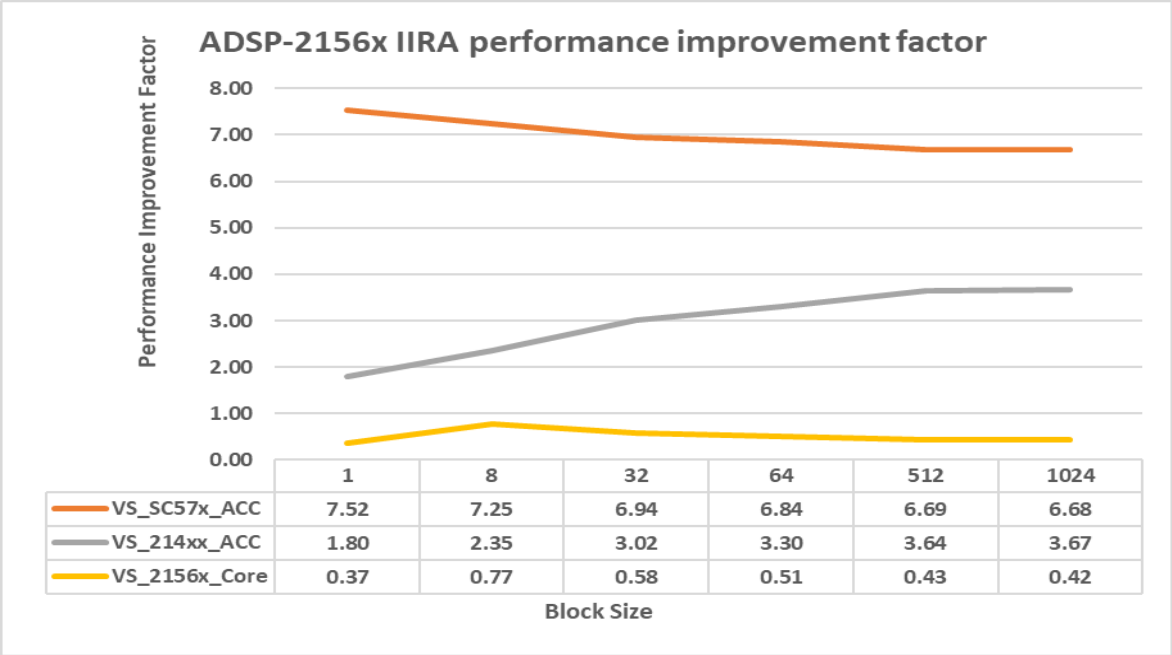


Figure 9: ADSP-2156x IIR Performance Improvement Factor for 6 Biquads Considering Time

The performance numbers shown in Figure 9 correspond to a selected number of combinations of window size and number of biquad sections. For other combinations, the following example code provided with this application note [4] can be used to measure the IIR performance on the evaluation boards for the four cases:

1. ADSP_2156x_IIRA_Performance
2. ADSP_SC57x_IIRA_Performance
3. ADSP_214xx_IIRA_Performance
4. ADSP_2156x_IIR_Core_Performance

The following observations relate to the IIR performance numbers shown in Figure 9 :

- **The ADSP-2156x IIRA performance is around 6.7x the ADSP-SC57x IIRA performance.** The ADSP-2156x IIRA runs at CCLK (1 GHz), whereas the ADSP-SC57x IIRA runs at SCLK, which is a maximum of CCLK/4 (125 MHz). The reduction in the improvement factor from 8x to 6.7x, as compared with the same comparison for the FIRA, is because the ADSP-2156x IIRA compute unit takes 6 cycles per biquad, whereas the ADSP-SC57x IIRA takes 5 cycles per biquad.
- **The ADSP-2156x IIRA performance is around 3.7x the ADSP-214xx IIRA performance.** The ADSP-2156x IIRA runs at CCLK (1 GHz), whereas the ADSP-214xx IIRA runs at PCLK, which is a maximum of CCLK/2 (225 MHz). The reduction in the improvement factor from 4.44x to 3.7x, as compared with the same comparison for the FIRA, is because the ADSP-2156x IIRA compute unit takes 6 cycles per biquad, whereas the ADSP-214xx IIRA takes 5 cycles per biquad.
- **The ADSP-2156x IIRA performance is around 0.42x the ADSP-2156x SHARC+ core's execution of IIR code**. The ADSP-2156x IIRA has one MAC unit, whereas the SHARC+ core featured on

ADSP-2156x processors has two MAC units. Additionally, the IIRA compute unit takes 6 cycles per biquad, whereas the SHARC+ core takes 2.5 cycles.

Compute Efficiency (CE)

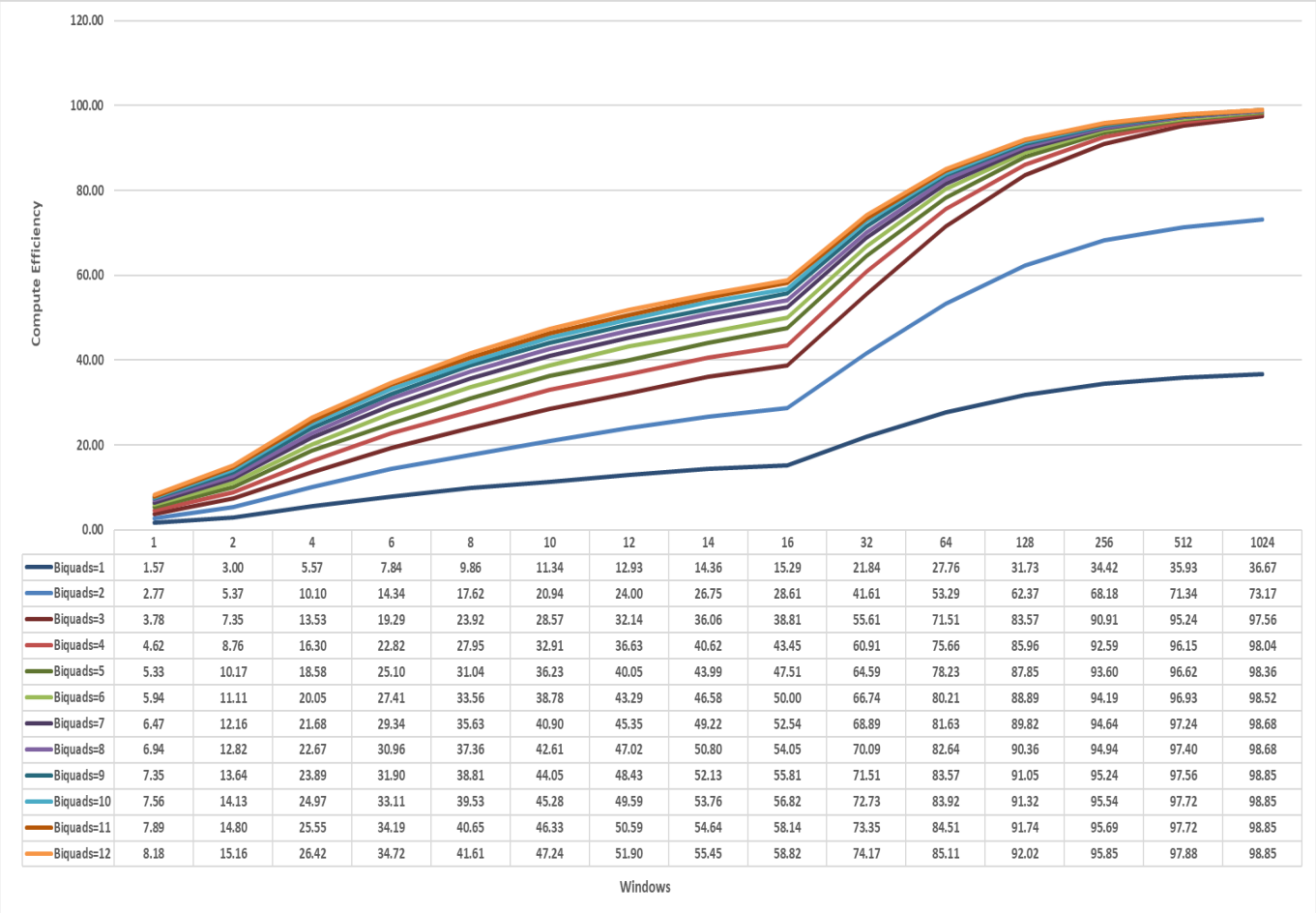
Figure 10 shows how the CE percentage for IIR varies with window size and number of biquads. As can be seen in the plot, the CE is very close to 100% for larger window size values, and it significantly reduces for small window size values. For a constant window size, the CE percentage does not vary much with number of biquads, except when the biquad count is 1 or 2, as the input/output DMA cycles dominate the compute cycles when this is the case.

Buffer Placement

Figure 11 shows how the core cycles change when the input/output/coefficient buffers are placed in L1/L2/L3 memory with a window size equal to 1024 and the number of biquads is increased. It also shows a plot of the performance factor when the buffers are in L1 memory as compared to when they are in L2/L3 memory.

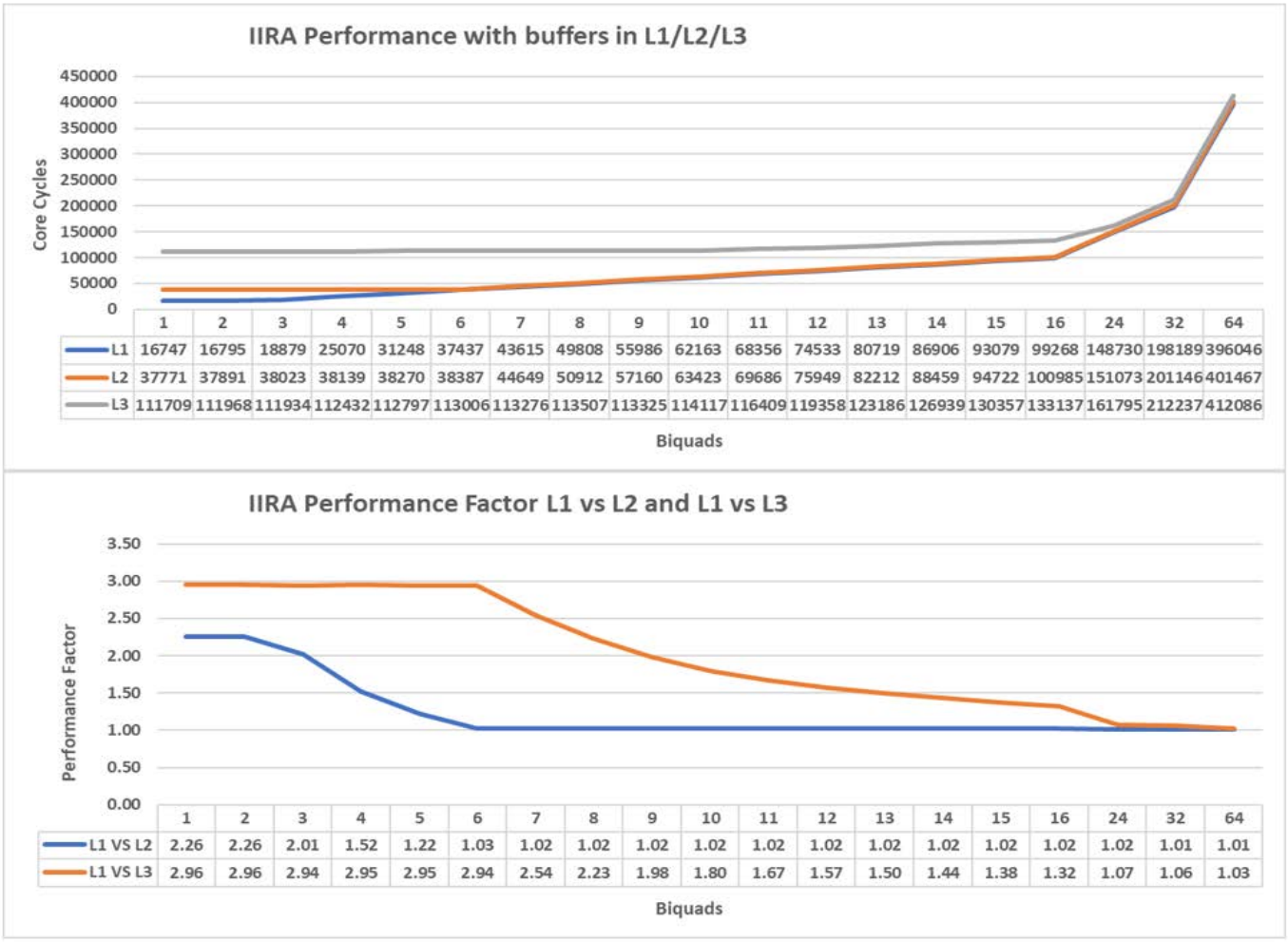
The following important observations can be noted from the plots:

- As the number of biquads is increased, the core cycles remain constant until point (N), at which point the core cycles start increasing proportionally. This change happens because the input/output DMA cycles dominate the compute cycles when the biquad count is low.
- As the memory access latency increases from L1 to L2 memory and then further to L3 memory, the value of N increases, and the performance factor trends towards 1.0. Placing buffers in L2 and L3 memory instead of L1 memory might be expensive for biquad values less than N and comparable for biquad values greater than or equal to N.



	1	2	4	6	8	10	12	14	16	32	64	128	256	512	1024
Biquads=1	1.57	3.00	5.57	7.84	9.86	11.34	12.93	14.36	15.29	21.84	27.76	31.73	34.42	35.93	36.67
Biquads=2	2.77	5.37	10.10	14.34	17.62	20.94	24.00	26.75	28.61	41.61	53.29	62.37	68.18	71.34	73.17
Biquads=3	3.78	7.35	13.53	19.29	23.92	28.57	32.14	36.06	38.81	55.61	71.51	83.57	90.91	95.24	97.56
Biquads=4	4.62	8.76	16.30	22.82	27.95	32.91	36.63	40.62	43.45	60.91	75.66	85.96	92.59	96.15	98.04
Biquads=5	5.33	10.17	18.58	25.10	31.04	36.23	40.05	43.99	47.51	64.59	78.23	87.85	93.60	96.62	98.36
Biquads=6	5.94	11.11	20.05	27.41	33.56	38.78	43.29	46.58	50.00	66.74	80.21	88.89	94.19	96.93	98.52
Biquads=7	6.47	12.16	21.68	29.34	35.63	40.90	45.35	49.22	52.54	68.89	81.63	89.82	94.64	97.24	98.68
Biquads=8	6.94	12.82	22.67	30.96	37.36	42.61	47.02	50.80	54.05	70.09	82.64	90.36	94.94	97.40	98.68
Biquads=9	7.35	13.64	23.89	31.90	38.81	44.05	48.43	52.13	55.81	71.51	83.57	91.05	95.24	97.56	98.85
Biquads=10	7.56	14.13	24.97	33.11	39.53	45.28	49.59	53.76	56.82	72.73	83.92	91.32	95.54	97.72	98.85
Biquads=11	7.89	14.80	25.55	34.19	40.65	46.33	50.59	54.64	58.14	73.35	84.51	91.74	95.69	97.72	98.85
Biquads=12	8.18	15.16	26.42	34.72	41.61	47.24	51.90	55.45	58.82	74.17	85.11	92.02	95.85	97.88	98.85

Figure 10: IIRA Compute Efficiency (CE) for Different Filter Parameters



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	24	32	64
L1	16747	16795	18879	25070	31248	37437	43615	49808	55986	62163	68356	74533	80719	86906	93079	99268	148730	198189	396046
L2	37771	37891	38023	38139	38270	38387	44649	50912	57160	63423	69686	75949	82212	88459	94722	100985	151073	201146	401467
L3	111709	111968	111934	112432	112797	113006	113276	113507	113325	114117	116409	119358	123186	126939	130357	133137	161795	212237	412086

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	24	32	64
L1 VS L2	2.26	2.26	2.01	1.52	1.22	1.03	1.02	1.02	1.02	1.02	1.02	1.02	1.02	1.02	1.02	1.02	1.02	1.01	1.01
L1 VS L3	2.96	2.96	2.94	2.95	2.95	2.94	2.54	2.23	1.98	1.80	1.67	1.57	1.50	1.44	1.38	1.32	1.07	1.06	1.03

Figure 11: ADSP-2156x FIRA Performance Comparison for Buffers in L1/L2/L3 Memory

Programming the Accelerators

The ADSP-2156x accelerators can be programmed using the device drivers available with the CCES installation package.

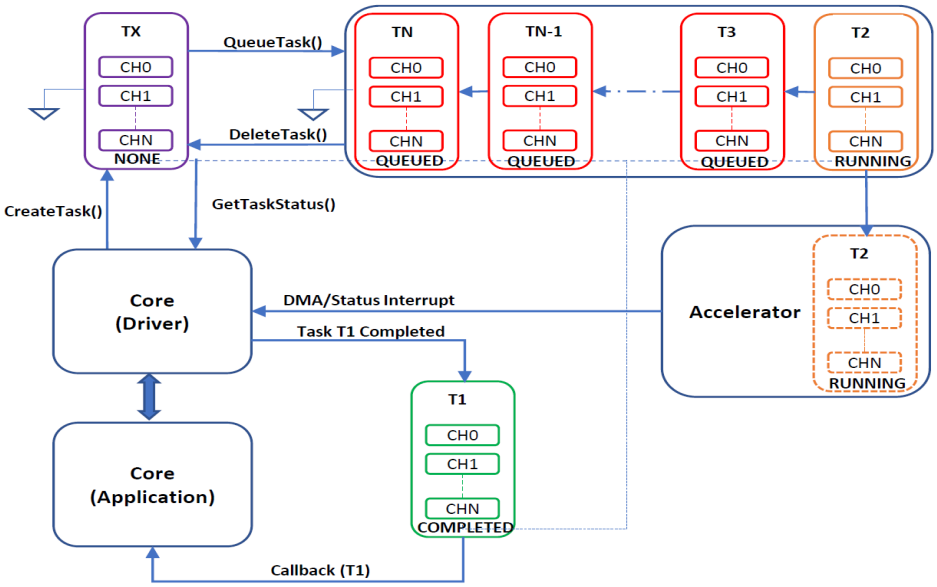


Figure 12: ADSP-2156x Accelerator Device Driver Structure

The programming model is the same for both the FIR and IIR accelerators. As shown in Figure 12, the accelerator drivers follow a queueing programming model. Abstractions of task and queue are used in the model. A task contains a set of channels which, when given to the accelerator, are processed sequentially. A queue is a set of tasks maintained on a first-in-first-processed basis which the accelerator processes in sequentially. Applications can create tasks for the accelerator using the `adi_fir(/iir)_CreateTask()` and queue the same when required using the `adi_fir(/iir)_QueueTask()`. The driver maintains a processing queue of the tasks given/queued by the application for the accelerator and sequentially schedules the same to the accelerator. The applications can register a user-defined callback using `adi_fir(/iir)_RegisterCallback()`. The driver uses the callback to notify the application on completion of the tasks/channels in a task. The driver also maintains the status of all the tasks the application has queued. The application can also query the status of the task as required using the `adi_fir(/iir)_GetFir(/Iir)TaskStatus()`.Once the task which was queued is completed by the accelerator, the task is removed from the processing queue. The driver maintains the information of all the tasks which are completed/created. Hence, the application can reuse the tasks which were created by simply queuing it again using `adi_fir(/iir)_QueueTask()`.

The accelerator can be configured to operate in legacy mode or ACM using static configuration. In legacy mode, the driver maintains a software queue (linked list) of the tasks queued by the application and sequentially schedules the same to the accelerator. In ACM, the driver utilizes the halt feature in hardware to implement a hardware queuing mechanism. There are additional channel-customizable features available in ACM like interrupts/triggers and more. Refer to `ADI_FIR(/IIR)_CHANNEL_INFO` in the *ADSP-2156x SSDD API Reference Manual for SHARC+ Core*[3] in the CCES help for more details. The programming model of the driver is the same for both legacy and ACM operation.

The `FIR_Multi_Channel_Processing` and `IIR_Multi_Channel_Processing` code examples provided with this application note[4] (also available with the ADSP-21569 EZ-Kit *Board Support Package*) can be used as a reference to understand how to use the ADSP-2156x accelerator device drivers.

Table 1 summarizes the accelerator driver benchmarks measured on the ADSP-21569 EZ-Kit evaluation board. The `ADSP_2156x_FIRA_Driver_Benchmark` and `ADSP_2156x_IIRA_Driver_Benchmark` code examples supplied with this application note[4] were used to obtain these measurements. Note that these numbers were measured with all buffers in L1 memory.

Operation	Description	Measured Core Cycles			
		FIRA		IIRA	
		Legacy	ACM	Legacy	ACM
CreateTask()	Constant Overhead	76	148	114	120
	Per Channel Overhead	306	320	302	327
QueueTask()	Pushing a task to the empty queue	300	371	299	371
	Pushing a task to the non-empty queue	195	329	188	321
Interrupt Overhead (Round trip cycles including SEC interrupt dispatcher latency)	For a single task in queue	429	423	430	426
	For more than one task in queue	503	595	523	571

Table 1: ADSP-2156x Accelerator Driver Benchmarks

Additional overhead for cache flushing/invalidation can occur when the buffers are placed in L2/L3 memories.

Accelerator Usage Models

This section describes the various models to use the accelerator optimally for different application scenarios. The ideal and most optimum solution is to offload all the FIR/IIR tasks from the core to the accelerators and allow the core to do something else in parallel; however, this solution may not be feasible for all the scenarios, particularly when the core needs to use the output from the accelerator for further processing and has no other independent tasks to finish in parallel. For these cases, this note discusses the accelerator usage models shown in Figure 13.

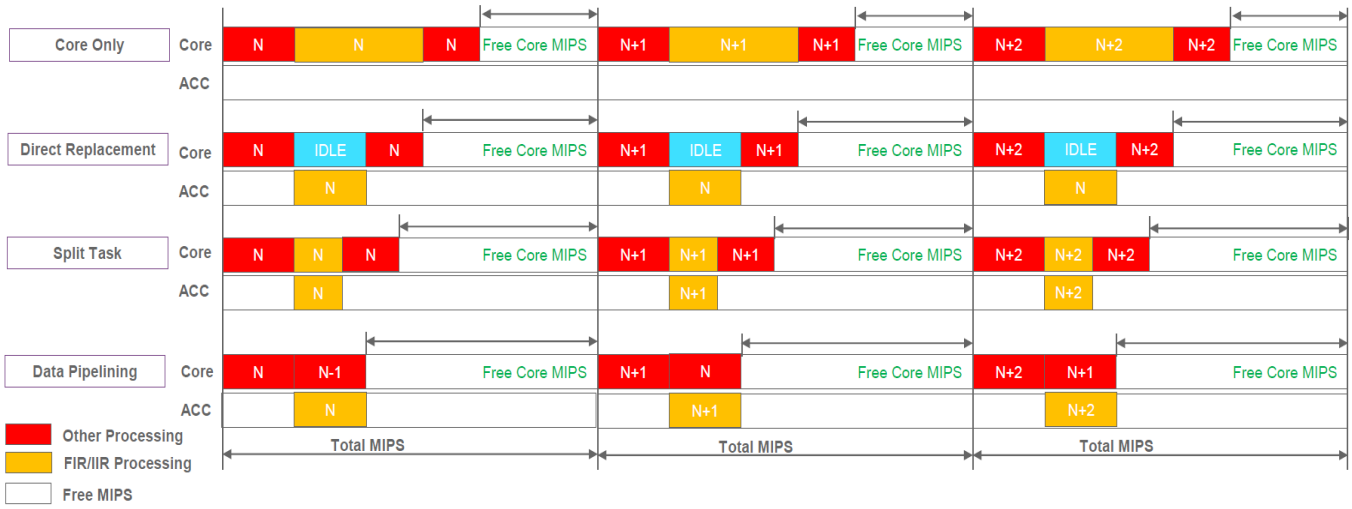


Figure 13: Accelerator Usage Models

Direct Replacement

This model is the most straightforward and easy-to-use. The core FIR/IIR processing is directly replaced by the accelerator, and the core simply waits for the accelerator to finish the job. This model is useful only when the FIRA is used because it is faster than the core.

Split Task

In this model, the overall FIR/IIR processing job is divided between the core and the accelerator. This model is especially useful when there are multiple channels available to be processed in parallel. Based on a rough timing estimation, the total number of channels can be divided between the core and the accelerator in such a way that both finish at approximately the same time. In such cases, this approach is better than the direct replacement approach. The split task model uses the idle time of the core to finish the overall FIR/IIR processing job faster than the accelerator processing the complete job alone.

Data Pipelining

In this model, data flow between the core and accelerator can be pipelined in such a way that both can work in parallel on different data frames. As shown in Figure 13, the core processes the Nth frame and then initiates the accelerator's processing of this frame. The core then continues in parallel to further process the N-1th frame output produced by the accelerator in the previous frame. This sequence allows the complete offloading of the FIR/IIR processing task to the accelerator at the cost of additional output latency. The pipeline stages and, consequently, the output latency can increase depending upon the number of such FIR/IIR processing stages in the complete processing chain.

Table 2 compares the usage models.

Approach	Description	Performance Advantage	Constraints
Direct Replacement	Core processing is replaced by accelerator, core waits until the accelerator completes the job.	+	Suitable only for FIRA, as it is faster than the core.
Split Task	FIR/IIR processing of multiple channels is split between the core and the accelerator.	++	Can't be used for single-channel processing.
Data Pipelining	Data between the core and accelerator is pipelined to allow the core and accelerator to run in parallel	+++	Costs additional latency of at least one frame.

Table 2: Accelerator Usage Model Comparison

Real-Time Implementation of an Example FIR/IIR Use Case

Figure 14 shows an example real-time use case for FIR/IIR processing of 12-channel audio data.

- Each channel passes through a 512-tap FIR filter followed by a five-band IIR equalizer.
- For simplification, the FIR filter is an all-pass filter (the b0 coefficient is one, and all other coefficients are zero).

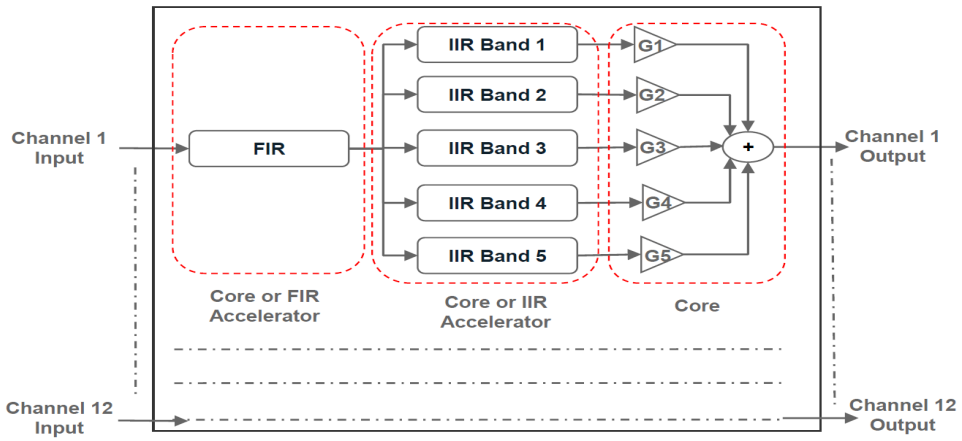


Figure 14: FIR/IIR Real-Time Use Case

- Each IIR band is an 8th order IIR filter implemented with four cascaded biquad stages. The IIR band details are as shown in Table 3. The output of each band is multiplied with the respective gain (from 0.0 min to 1.0 max) and added together to generate the final output. The ***IIRGains*** flag can be set or cleared dynamically using the CCES memory browser to respectively bypass or enable processing.

Band No.	Lower Cutoff in Hz	Higher Cutoff in Hz	Filter Type
1	-	250	Low Pass
2	250	500	Band Pass
3	500	2000	Band Pass
4	2000	4000	Band Pass
5	4000	-	High Pass

Table 3: IIR Bands Details

- The block size is 256 samples per channel.
- The example code uses only two-channel audio input and output on the ADSP-21569 EZ-Kit evaluation board. The input data for the remaining 10 channels is "zero". Since only the performance is being measured, this limitation does not affect the results.
- All buffers are placed in internal (L1) memory.
- The ***BypassProcessing*** flag can be set or cleared dynamically using the CCES memory browser to respectively bypass or enable processing.
- The ***cycles_array*** array can be observed in the expressions window to see the periodic cycle count snapshot trace. This information can be used to calculate MIPS details.

The `Direct_Replacement`, `Pipelined`, and `Split_Task example code` supplied with the application note [4] implements this case in real-time on the ADSP-21569 EZ-Kit evaluation board for the different usage models. The core and accelerator performance numbers are measured for the *core only* case and for each accelerator usage model along with the resultant core MIPS savings for each model. Figure 15 shows the breakup of the cycle/MIPS measurements for both the core and the accelerator with the cycle snapshots taken at different instances of the overall processing time. It also shows the total MIPS consumed and total free MIPS for both the core and the accelerator.

Core Only

Snapshot No.	Description	Core State	Accelerator State	Cycles	MIPS
1	Function Entry			53	0.01
2	Misc			27	0.01
3	FIR Processing			822221	154.17
4	IIR Processing			175642	32.93
5	Multiply gain and add IIR Output			26405	4.95
6	Function Exit			47	0.01
7	Free core MIPS			4308851	807.91
Total MIPS					1000.00
Accelerator MIPS Usage		0.00	Core MIPS Usage		192.07
Free Accelerator MIPS		1000.00	Free Core MIPS		807.93

Direct Replacement

Snapshot No.	Description	Core State	Accelerator State	Cycles	MIPS
1	Function Entry			53	0.01
2	Misc			27	0.01
3	FIR Task Queue			301	0.06
4	FIR Processing			435293	81.62
5	IIR Task Queue			2615	0.49
6	IIR Processing			398365	74.69
7	Multiply gain and add IIR Output			26420	4.95
8	Function Exit			47	0.01
9	Free core MIPS			4470298	838.18
Total MIPS					1000.00
Accelerator MIPS Usage		156.31	Core MIPS Usage		161.84
Free Accelerator MIPS		843.69	Free Core MIPS		838.16

Split Task

Snapshot No.	Description	Core State	Accelerator State	Cycles	MIPS
1	Function Entry			53	0.01
2	FIR Task Queue			302	0.06
3	FIRA Processing of 7 channels, Core FIR Processing 5 channels			254238	47.67
4	FIRA Done Callback - IIR queue task			1596	0.30
5	IIRA Processing of 7 channels, Core FIR Processing of 5 channels			259560	48.67
6	IIRA Processing of 7 channels, Core IIR Processing of 5 channels			73212	13.73
7	Multiply gain and add IIR Output			24918	4.67
8	Function Exit			47	0.01
9	Free core MIPS			4719491	884.90
Total MIPS					1000.00
Accelerator MIPS Usage		110.06	Core MIPS Usage		115.11
Free Accelerator MIPS		889.94	Free Core MIPS		884.89

Data Pipelining

Snapshot No.	Description	Core State	Accelerator State	Cycles	MIPS
1	Function Entry			52	0.01
2	Multiply gain and add IIR Output			26438	4.96
3	FIR Task Queue			302	0.06
4	Function Exit			54	0.01
5	FIRA processing, free core MIPS			435133	81.59
6	FIRA Done Callback - IIR queue task			2616	0.49
7	IIRA CH1 processing, free core MIPS			31041	5.82
8	IIR CH1 Callback			36	0.01
9	IIRA CH2 processing, free core MIPS			33339	6.25
10	IIR CH2 Callback			33	0.01
11	IIRA CH3 processing, free core MIPS			33339	6.25
12	IIR CH3 Callback			33	0.01
13	IIRA CH4 processing, free core MIPS			33406	6.26
14	IIR CH4 Callback			33	0.01
15	IIRA CH5 processing, free core MIPS			33339	6.25
16	IIR CH5 Callback			33	0.01
17	IIRA CH6 processing, free core MIPS			33339	6.25
18	IIR CH6 Callback			33	0.01
19	IIRA CH7 processing, free core MIPS			33414	6.27
20	IIR CH7 Callback			33	0.01
21	IIRA CH8 processing, free core MIPS			33339	6.25
22	IIR CH8 Callback			33	0.01
23	IIRA CH9 processing, free core MIPS			33415	6.27
24	IIR CH9 Callback			33	0.01
25	IIRA CH10 processing, free core MIPS			33339	6.25
26	IIR CH10 Callback			33	0.01
27	IIRA CH11 processing, free core MIPS			33340	6.25
28	IIR CH11 Callback			33	0.01
29	IIRA CH12 processing, free core MIPS			33270	6.24
30	IIR CH12 Callback			33	0.01
	Free core MIPS			4470343	838.19
Total MIPS					1000.00
Accelerator MIPS Usage		156.27	Core MIPS Usage		5.60
Free Accelerator MIPS		843.73	Free Core MIPS		994.40

Busy
Free
Waiting

Figure 15: Cycles/MIPS Breakup for Different Accelerator Usage Models

Table 4 summarizes the MIPS usage and free MIPS for both the core and the accelerator for the *Core Only* case and for different accelerator usage models along the core MIPS saving numbers. From a performance perspective, the *Data Pipelining* model is the best, resulting in saving approximately 186 core MIPS, followed by the *Split Task* model (saving 77 core MIPS) and then the *Direct Replacement* model (saving 30 core MIPS). These results align with the previous discussion about the accelerator usage models.

Usage Model	MIPS Usage		Free MIPS		Core MIPS Saving
	Core	ACC	Core	ACC	
Core Only	192.07	0.00	807.93	1000.00	-
Direct Replacement	161.84	156.31	838.16	843.69	30.23
Split Task	115.11	110.06	884.89	889.94	76.96
Data Pipelining	5.6	156.27	994.40	843.73	186.47

Table 4: MIPS Summary for Different Accelerator Usage Models

References

[1] *ADSP-21562/21563/21565/21566/21567/21569 Data Sheet.* Rev PrD, January 2019. Analog Devices, Inc.

[2] *ADSP-2156x SHARC+ Processor Hardware Reference,* Rev 0.1, October 2018. Analog Devices, Inc.

[3] *ADSP-2156x SSDD API Reference Manual for SHARC+ Core*, Version 2.0, CrossCore® Embedded Studio Help.

[4] *Associated ZIP file for EE-408: Using ADSP-2156x High Performance FIR/IIR Accelerators,* August 2019. Analog Devices, Inc.

Document History

Revision	Description
Rev 1 – June 10, 2019 *by Sanket Nayak and Mitesh Moonat*	Initial Release
Rev 2 – August 14, 2019 *by Sanket Nayak and Mitesh Moonat*	Minor edits in the document and in the real time FIR/IIR example code. Removed preliminary ADSP-214xx core benchmark data (to be included in future revision).