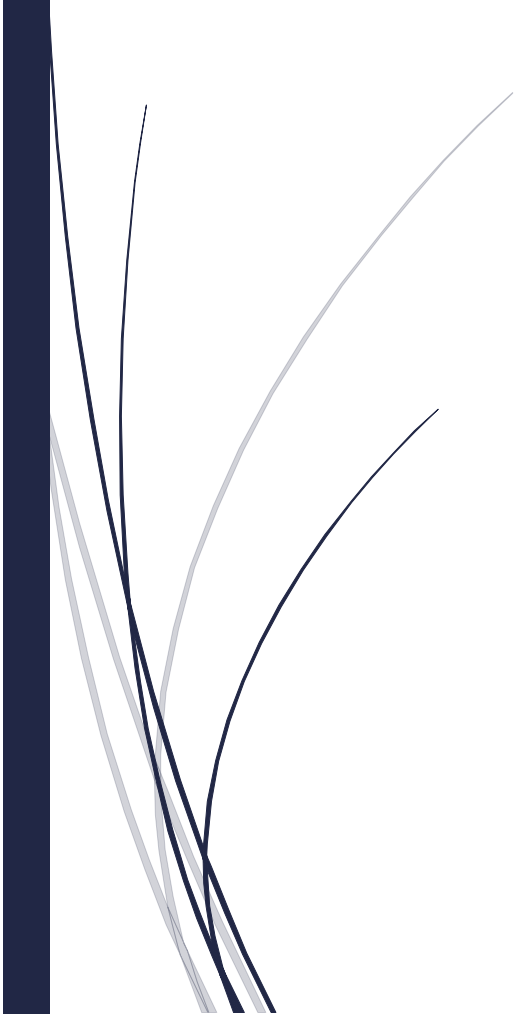
6.4.2026

FFT IP

IrisCores.com



Vojtěch Terš

1 Revision history

Version	Date	Description
1.0	17.4.2023	Initial release with “full” capabilities
2.0	14.8.2023	1. Consolidated Indexing cores 2. Improved timing on Indexing 3. Reduced latency of Indexing
3.0	11.9.2023	Support for Cooley-Tukey Decimation in Time (DIT). Defaults still to DIF (Decimation in Frequency). This feature has currently no direct benefits while doing standard FFT transforms. In fact, it is slightly slower and requires more FPGA resources. It is recommended to keep “DIF” architecture.
4.0	4.11.2023	Added support for high-speed Pipelined architecture.
5.0	19.11.2023	Added proprietary Floating-Point support
6.0	20.11.2023	Removed backpressure on stage indexers to reduce resource usage especially for Pipelined architecture.
7.0	25.11.2023	Added burst flagging across FFT Loops. Conditionally reverted to support backpressure within Indexer.
7.1	24.8.2024	Improved documentation, added RX/TX Count registers. Minor bugfixes and code base adjustments.
8.0	5.9.2024	Added support for CDC on AXI4-Lite and Padding Support for FFTs.
8.1	6.9.2024	Removed Unnecessary latches from Pipelined version
8.2	4.12.2024	Radix8 base now properly propagates the DSP Attributes. Reduced Latency and resource usage during data dumping. Introduced 3 new generics to allow for improved timing especially for R4 / R8.
9.0.3	31.8.2025	This is major redesign with several important enhancements and updates with primary focus on new RSDF architecture and various resource optimizations: 1. Reduced DSP Count for Pipelined Architecture 2. Reduced DSP Processing Resources for Mixed-Radix Transforms. 3. Reduced ROM Memory requirements for Twiddle Factors for PIPE and R2SDF architectures. 4. Reduced RAM Memory requirements for PIPE architecture. 5. Stage Indexing Resource Logic & Latency Opt. 6. DSP Processing Resource Logic Opt. 7. Digit-Reversal Resource Logic Opt. 8. Added optional bitmask estimation for fixed point data across FFT Stages to streamline any dynamic-range problems diagnosis.
9.0.4	9.9.2025	Updated resource estimates, added port descriptions and missing register definition in regard to the bit-masking.
10.0.0	6.4.2026	Architecture Updates: - Finalized Pipelined SSR Architecture Feature Updates: - Support for 3 or 4 – way complex Multiplier - Support of Cyclic Prefix Insertion - Minimum Transform Size Settings - Custom Scaler Width Configuration - RTL-Based Reconfiguration Others: - Timing Optimization Sweeps - Resource Optimization Sweeps - Throughput Optimization Sweeps

	- Codebase refactoring and Enhancements - Minor Bugfixes

2 ABBREVIATIONS

Abbreviation	Full Term
BTF	Butterfly
AXI	Advanced eXtensible Interface
CDC	Clock Domain Crossing
CP	Cyclic Prefix
DFT	Discrete Fourier Transform
DIF	Decimation In Frequency
DIT	Decimation In Time
DR	Dynamic Range
DSP	Digital Signal Processing
ENUM	Enumeration
FFT	Fast Fourier Transform
FI	Fixed Point
FP	Floating Point
IP	Intellectual Property
ISI	Inter-Symbol Interference
LSB	Least Significant Bit
LTE	Long-Term Evolution
MSB	Most Significant Bit
OFDM	Orthogonal Frequency Division Multiple Access
R2	Radix-2
R4	Radix-4
R8	Radix-8
RAM	Random-Access Memory
ROM	Read-Only Memory
RTL	Register-Transfer Level
SDF	Single-Path Delay Feedback
SDR	Software-Defined Radio
SL	Std Logic
SLV	Std Logic Vector
SSR	Super Sample Rate

3 REFERENCES

1) AMBA AXI Protocol Specification “ARM IHI 0022”

3.1 FEATURES:

1. **Available architectures**
   - **1.1.** BASE (Iteration-based Processing Engine)
   - **1.2.** RSDF (Continuous Streaming Pipeline)
   - **1.3.** PIPE (Super Sample Rate Pipeline)
2. **Supported Implementation Styles**
   - **2.1.** DIF – Decimation in Frequency
   - **2.2.** DIT – Decimation in Time
3. **Base Scalability parameters**
   - **3.1.** 100% Vendor-Independent VHDL-2008 Source Code
   - **3.2.** Parameterized Base Radix (2 / 4 / 8)
   - **3.3.** Transform size configurable up to 65,536 points
   - **3.4.** Fixed-Point and Floating-point Arithmetic's Support
     - 3.4.1. Note: Floating-Point arithmetic requires third-party or vendor-specific external IP libraries.
   - **3.5.** Natural Output Order by default
   - **3.6.** Configurable Data and Twiddle phase width (up to 32-bit)
   - **3.7.** Industry-standard AXI4-Stream (Data) and AXI4-Lite (Control) Interfaces
   - **3.8.** Advanced DSP handling: Overflow detection, saturation logic, and Invalid Packet Detection
   - **3.9.** Configurable Fixed-Point Rounding modes
     - 3.9.1. Truncate
     - 3.9.2. Round Half-Up
     - 3.9.3. Round Half-Even
   - **3.10.** Runtime Dynamic Per-Packet Reconfiguration
     - 3.10.1. Transform Size
     - 3.10.2. Transform Direction (Forward / Inverse)
     - 3.10.3. Scaling Schedule
     - 3.10.4. Cyclic Prefix
4. **Optional features**
   - **4.1.** Complex Multiplier Architecture Selection (3-DSP or 4-DSP optimization)
   - **4.2.** Optional Cyclic Prefix (CP) Insertion
   - **4.3.** Optional Fixed-Point Dynamic Range Estimation
   - **4.4.** Optional Clock Domain Crossing (CDC) synchronization for AXI4-Lite

3.2 LIMITATIONS

1. **AXI4-Lite Access:**
   1.1. The control interface strictly requires 32-bit read/write accesses. Unaligned (Ie. Address of 0x001) or partial (8-bit / 16-bit) accesses are not supported.
2. **Floating-Point Arithmetic:**
   2.1. Floating-point support is implemented using external, proprietary 3rd-party IPs (the core provides the necessary pipeline routing and control, but not the raw floating-point DSP logic).
3. **Phase and Data Width Limits:**
   3.1. The maximum supported Data and Twiddle factor phase widths are 32 bits.
4. **Cyclic Prefix Constraints:**
   4.1. Hardware-automated Cyclic Prefix (CP) insertion within the BASE architecture strictly requires the DIT (Decimation in Time) implementation style. The Digit Reversal engine has to be enabled for CP to work properly.
   4.2. Configurations with AXIS_SAMPLES_M > 1 must specify a multiple of this value. For example, PIPE-R8 architecture must be aligned to multiples of 8.
5. **AXI4-Stream Strobing Restrictions:**
   5.1. Certain constraints apply to the partial strobing of the input/output AXI4-Stream interfaces (Detailed in the Interface Specifications chapter)
6. **Fixed-Point Formatting:**
   6.1. Specific alignment and format restrictions apply to the Fixed-Point processing mode (Detailed in the Arithmetic & Precision chapter)
7. **Minimum Transform Size:**
   7.1. The minimum allowable transform size (NFFT) is bounded by the selected Base Radix.
      7.1.1. For BASE_RADIX = 4, NFFT=2 is not supported
      7.1.2. For BASE_RADIX = 8, NFFT=2 and NFFT=4 is not supported.
8. **Mixed-Radix Throughput Throttling:**
   8.1. High-throughput configurations may temporarily throttle down if the requested transform size cannot be factored purely by the Base Radix (e.g., executing a 32-point transform on a Radix-4 architecture requires a mixed 4×4×2 decomposition stage)
9. **Throughput guarantees:**
   9.1. RSDF (R2) is guaranteed for continuous 100% gapless throughput for N_{FFT} >= 8.
   9.2. PIPE-R2 is guaranteed for continuous 100% gapless throughput for N_{FFT} >= 64.
   9.3. PIPE-R4 is guaranteed for continuous 100% gapless throughput for N_{FFT} >= 128.
   9.4. PIPE-R8 is guaranteed for continuous 100% gapless throughput for N_{FFT} >= 256.
10. **Expected Synthesis Warnings:**
   10.1. The IP is highly parameterized. Depending on the configuration, certain internal interfaces, ports, or logic blocks may be partially or completely unused. This will result in standard synthesis warnings (e.g., "Signal disconnected" or "Logic optimized away") which are expected and safe to ignore.

4 Core Overview

The Iris FFT IP Core is an enterprise-grade, highly parameterized hardware implementation of the Discrete Fourier Transform (DFT). Built upon the highly efficient Cooley-Tukey algorithm, the core natively supports both Decimation-in-Time (DIT) and Decimation-in-Frequency (DIF) decomposition. Designed specifically to break the limitations of standard, vendor-locked EDA tools, the IP serves as a comprehensive ecosystem. It offers unparalleled Performance/Power/Area (PPA) trade-offs, making it the ultimate DSP processing engine for next-generation Software-Defined Radio (SDR), 5G/6G telecommunications, Radar, Electronic Warfare (EW) systems and similar applications.

To meet diverse system requirements, the IP provides three distinct, independent underlying architectures. Mathematically, the core computes N/R butterfly operations per transform stage (where N is the Transform Size and R is the Base Radix), scaled across the following topologies:

1. **The BASE Architecture:** An iterative processing engine computing one butterfly (Radix-2, Radix-4, or Radix-8) per clock cycle. It offers an optimal balance between minimal logic utilization and execution time for embedded systems.
2. **The RSDF Architecture:** A Radix-2 Single Delay Feedback topology that unrolls the transform stages across multiple processing elements. It guarantees continuous data streaming of up to one sample per clock cycle.
3. **The PIPE Architecture:** A Super-Sample Rate (SSR) fully pipelined architecture offering the highest processing bandwidth. Depending on IP configuration, used device fabric and clock rate, it is capable of transforming massively parallel data streams (up to ~3.5 GSPS in Radix-8 mode).

The built-in DSP butterfly processing engines feature a versatile data path supporting both Fixed-Point and Single-Precision Floating-Point implementations (with configurable latency routing for external floating-point libraries). To ensure maximum Signal-to-Noise Ratio (SNR) and mathematical fidelity in Fixed-Point mode, the internal processing within each FFT stage is computed using extended, full data-width precision. The inevitable bit-growth inherent to the FFT butterflies is strictly mitigated at the very end of each stage, where the core applies user-configured dynamic scaling, truncation, and rounding schemes before passing the data to the next stage.

Beyond standard implementation styles and architectural decisions, the IP offers an advanced premium architectural extension, that allows the Bit/Digit reversal engines to be fully omitted from the DSP chain. This option drastically reduces resource usage and power consumption while simultaneously improving FMAX. It is ideal for applications, that aim to compute convolution via optimal FFT (DIF) → Complex Multiply → IFFT (DIT) data path.

The IP natively supports other essential industry-standard EDA features such as dynamic runtime reconfiguration on a per-packet basis (Transform Size, Direction and Scaling Schedule).

Packaged in 100% vendor-independent VHDL-2008 source code and wrapped in standard AXI4-Stream data interfaces, the IP guarantees seamless integration regardless of the selected FPGA fabric or a custom ASIC silicon die, thus aggressively challenging other standard EDA-provided IP solutions.

4.1 FIXED – POINT DATA PROCESSING

4.1.1Data Format

The IP core exclusively operates on signed, two's complement arithmetic. To prevent undetected hardware overflows during complex number rotations, the input data must adhere to a specific fractional format. Throughout this documentation, the: **QX.Y(u/s)**,notation is used, where:

1. **X** Represents total amount of all bits
2. **Y** Represents total amount of **fractional bits**
3. **u/s** Signedness (**u** = unsigned, **s** = signed)

Twiddle Factor Storage:

1. Fixed-Point arithmetic: Twiddle factors are permanently stored in internal ROMs using the format **QX.(X-1)s**. This bounds the coefficients strictly within the (-1.000f to 0.999) range.
2. Floating-point arithmetic: Twiddle factors are stored natively in the IEEE-754 Single-Precision Floating-Point format.

Complex Magnitude Headroom:

1) If a complex input vector approaches the maximum fractional boundaries (e.g., 0.999 + 0.999i), its magnitude during rotation can reach ~1.414f. To safely accommodate this without hardware overflow, the numerical format inherently requires at least one integer bit alongside the sign bit.
2) **Format Example:** 16-bit Data, 14 Fractional Bits, Signed (**Q16.14s**)
   - **a)** Maximum **supplied** value : 16'h3FFF (+ 0.999f)
   - **b)** Minimum **supplied** value : 16'hC000 (- 1.000f)
   - **c)** Maximum **supported** value : 16'h7FFF (+ 1.999f)
   - **d)** Minimum **supported** value : 16'h8000 (- 2.000f)

4.1.2Bit Growth

FFT butterfly operations inherently introduce bit growth. To preserve maximum mathematical precision, the core maintains extended, full-precision bit growth throughout the entirety of a given FFT stage. The data is only scaled, truncated / rounded back to the configured DSP data path width at the very end of the stage before being forwarded downstream

1) **Sources of bit growth per FFT stage:**
   Radix-2 Stage → 1 bit
   Radix-4 Stage → 2 bits
   Radix-8 Stage → 3 bits
   Twiddle factor multiplication: Defined by the configured Twiddle Phase width
2) **Internal Scaler Sizing Example:**
   Assuming a DSP Datapath Width of 24 bits, a Twiddle Phase Width of 16 bits, and a Radix-4 stage, the internal unscaled data width right before the scaler is calculated as: **24** + **16** + log2(**4**) = **42** Bits.
3) **Simulation Magnitude Checker:**
   To assist designers in verifying their input stimulus, the IP includes a built-in magnitude checker (active exclusively during Fixed-Point simulation). This verification block assesses every incoming sample on a bit-accurate level and generates console warnings if any data violates the recommended fractional boundaries

4.1.3 Scaling Block Operations

The internal scaling component features a user-selectable scale width configuration. It forms the critical final stage of the DSP data path, actively conditioning the data before it is forwarded to the next stage. The Scaler performs the following operations:

1. **Dynamic Scaling (Shifting)** : The data is arithmetically right-shifted by a programmable value of X bits. This operation dynamically controls the overall position of the binary point to mitigate bit-growth. Due to the inherent properties of 2's complement arithmetic, an arithmetic right-shift is not strictly identical to standard integer division (e.g., division by 2, 4, 8 …).
   1.1. Performance & Design Recommendation:
   It is highly recommended to keep the configured scale width as narrow as possible (typically 2 or 3 bits maximum). Because the shifting logic effectively infers a hardware barrel-shifter on the critical DSP data path, excessively large scale widths will directly impact routing resources and may degrade the maximum achievable clock frequency.

2. **Overflow Detection & Saturation**: Before the shift is applied, the component actively evaluates the Most Significant Bits (MSBs) to detect potential overflows. If the pre-shifted value exceeds the safely representable range of the target output width, the data path automatically saturates (clips) the output to its maximum positive or minimum negative representable value. Concurrently, an overflow flag is raised and propagated.

3. **Rounding**: As the data is shifted down, the remaining Least Significant Bits (LSBs) are utilized to properly round the result and minimize quantization noise. The core implements three distinct rounding topologies, selectable via synthesis generics:

   3.1. **Truncate:** Simply discards the shifted-out LSBs. This is the most logic-efficient and fastest solution, but it introduces the highest statistical DC bias into the resulting signal.
   3.2. **Round Half-Up (Default):** Also known as standard "Schoolbook Rounding." This method rounds any value with a fractional part of 0.5 or greater up to the next integer (away from zero). This configuration offers an excellent trade-off, utilizing minimal extra logic while sufficiently reducing the DC bias for most applications.
   3.3. **Round Half-Even:** As defined by the IEEE-754 standard (often called Convergent Rounding). This method rounds ties (exactly 0.5) to the nearest even integer. This scheme perfectly balances the rounding direction, statistically eliminating the DC bias present in the Round Half-Up method, at the cost of slightly higher resource utilization.

4.1.4 Default Scaling Schedule

There is a critical numerical implication when the core is configured to use fixed-point signed arithmetic. During the multiplication of two signed numbers, an additional integer bit is inherently created in the result to cover the specific worst-case scenario of multiplying two minimum negative values: **-1.000** x **-1.000 = +1.000.**

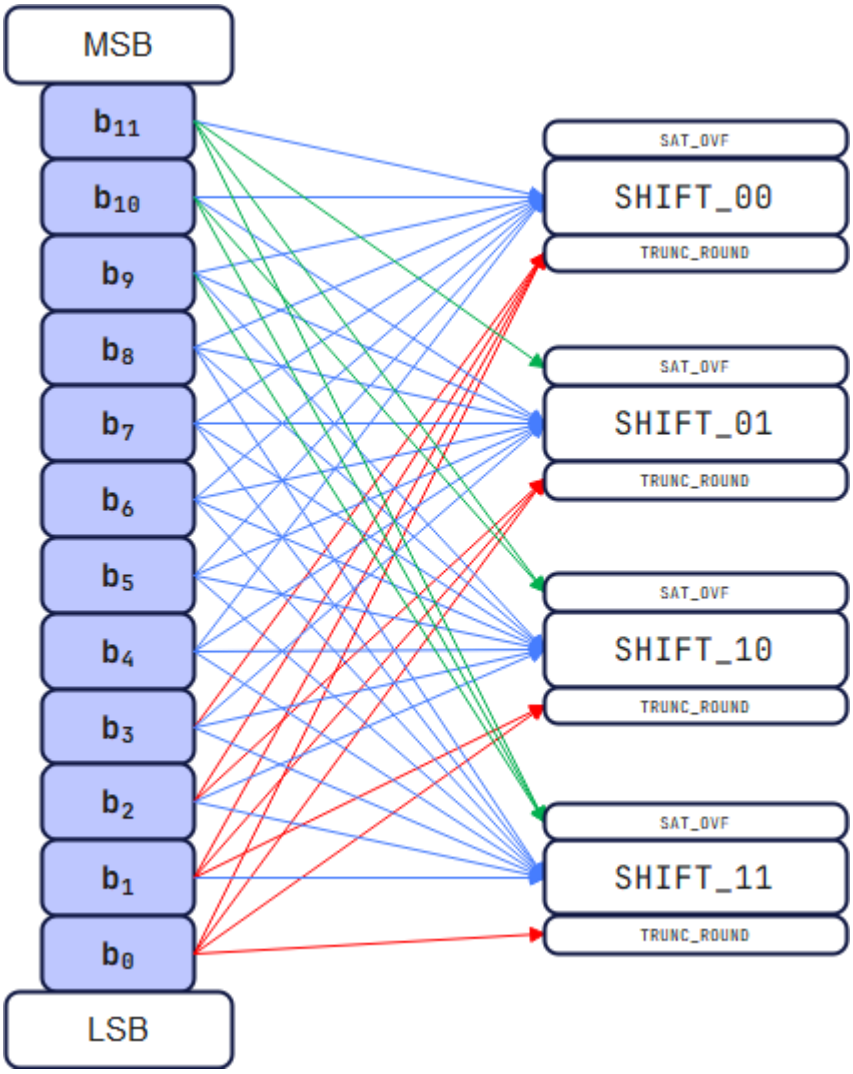
The input data format, for example **Q16.15s** (16-bits total, 15 fractional bits, signed), leaves a representable range of [**-1.000, +0.999**] and cannot hold the **+1.000** result. Applying the generic fixed-point multiplication rule "**Q(A.B)s * Q(C.D)s = Q(A+C).(B+D)s**" results in: Q16.15s * Q16.15s = **Q32.30s.**

While this expanded format correctly represents the +1.000 value, the intermediate data must be truncated and realigned back to the uniform DSP data path width, as each subsequent FFT stage requires a consistent input format. Therefore, to properly realign the binary point, the default scaling schedule **must always be set to a minimum shift of 1** for active processing stages.

4.1.5 Conversion Example 12-bit to 8-bit

The diagram below illustrates the internal scaling operation for a 12-bit intermediate result being converted and mapped back to an 8-bit core data path width.

1. **BLUE Arrows (Target Datapath):** Indicate the final 8 data bits selected for the output. The configured shift value (0, 1, 2, or 3) determines exactly which 8 bits are extracted from the 12-bit intermediate vector.
2. **GREEN Arrows (Overflow Evaluation):** Show the Most Significant Bits (MSBs) that are evaluated prior to the shift. If these bits indicate a value too large (or too negative) to fit securely within the 8-bit output window, the selected BLUE bits will be forcefully saturated to their maximum/minimum bounds
3. **RED Arrows (Truncation & Rounding):** Show the Least Significant Bits (LSBs) that are shifted out of the data path. If a rounding scheme is enabled (e.g., Round Half-Up), the most significant bit of this truncated RED segment is actively evaluated to round the final BLUE output

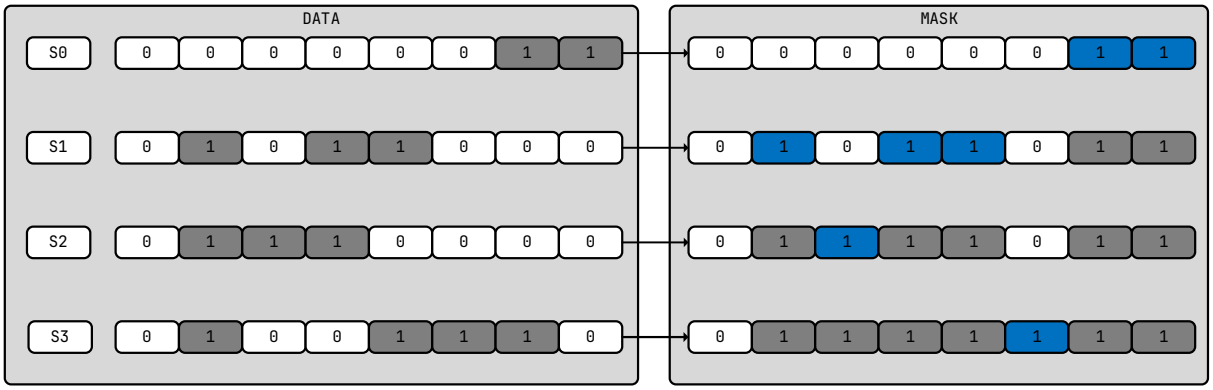


Architectural Insight: DIF vs. DIT Shifter Utilization

Each stage of the transform can be assigned an independent shift value via the scaling schedule, with a 1-bit shift being the standard recommendation. The Decimation-in-Time (DIT) architecture inherently requires double the amount of these scaling shifters compared to Decimation-in-Frequency (DIF). Without these additional shifters in DIT, the complex multipliers would become excessively large and severely degrade timing performance. For this reason, while DIT is fully supported, **the DIF architecture is should be selected for optimal logic utilization and maximum routing performance**.

4.2 Fixed Point Dynamic Range Estimation

1) **Overview & Diagnostic Purpose**
   The core optionally features a hardware-accelerated profiling tool designed to examine the actual utilized dynamic range of the data across all individual FFT stages. While the Scaling component actively diagnoses and reports critical overflow events, it cannot natively detect sub-optimal signal scaling or quantization noise issues. To definitively diagnose issues resulting in underflow or localized dynamic range reduction, the IP utilizes an optional Bitmask Extractor.
2) **Hardware Extraction Mechanism**
   This component acts as an inline diagnostic probe positioned immediately after the scaling component in each stage. It actively gathers and logically accumulates all toggling bits within a given transform. The resulting final bitmask represents precisely which data bits were actively utilized to transfer valid signal energy.
   The bitmask is evaluated cumulatively across the entire block of data samples for a given transform length. For instance, the diagnostic mask of a 1024-point FFT is the logical aggregate of all 1024 processed samples within that specific stage. Internally, the extractor automatically resets its local bitmask concurrently with the last sample of the current transform packet.



3) **AXI4-Lite Integration & Latching**
   To alleviate the polling requirements on the external host processor, the AXI4-Lite Top-Level Wrapper actively manages these extractions. It captures and securely latches the generated bitmasks from all active stages at the end of the transform. These values are held statically in the register map until the host software explicitly issues a manual "clear" command via the control interface, allowing for post-mortem analysis of the signal chain.
4) **Diagnostic Interpretation**
   Under optimal, full-scale signal conditions, all data bits within the data path should actively toggle. Therefore, the presence of constant zero (0) bits starting from the Most Significant Bit (MSB) downwards directly correlates to a wasted dynamic range and potential signal-to-quantization-noise (SQNR) degradation.
   a) **Example:** A read-out mask of 12'b**0000_1111_1111** indicates that out of the 12 available data path bits, the upper **4** MSBs remained completely static. Only 8 bits were actively utilized by the signal, highlighting a severely reduced dynamic range that may require upstream gain adjustments or a revised core scaling schedule.

4.3 EXTERNAL FLOATING-POINT INTEGRATION

1) **Integration Overview**
   To provide maximum computational performance and leverage silicon-specific DSP optimizations, the IEEE-754 Single-Precision Floating-Point arithmetic is not implemented natively within the core's generic VHDL logic. Instead, the IP utilizes a standardized wrapper architecture that relies on external, vendor-provided arithmetic IPs (e.g., AMD/Xilinx Floating-Point IP or Intel/Altera FP DSP blocks).
   ***Important Notice**:* When the IP is configured for Floating-Point processing, these external core netlists or libraries must be generated by the user and present in the compilation environment. The FFT core will not compile without them.
2) **Required Arithmetic Components**
   The IP instantiates three fundamental floating-point operations. The user must provide the corresponding IP wrappers in their project, exactly matching the component names and interface signatures below:
   a) **FP_ADD** (Floating-Point Adder)
   b) **FP_SUB** (Floating-Point Subtractor)
   c) **FP_MUL** (Floating-Point Multiplier)
3) **Standardized Component Signature**
   All three arithmetic components must adhere to a simplified, industry-standard AXI4-Stream interface. The required VHDL component declaration is strictly defined as follows:

```
component " FP_ADD | FP_SUB | FP_MUL "
port (
   aclk                    : in  sl;
   aresetn                 : in  sl;
   s_axis_a_tvalid         : in  sl;
   s_axis_a_tdata          : in  slv(31 downto 0);
   s_axis_b_tvalid         : in  sl;
   s_axis_b_tdata          : in  slv(31 downto 0);
   m_axis_result_tvalid    : out sl;
   m_axis_result_tdata     : out slv(31 downto 0)
 );
end component;
```

4) **Configurable Pipeline Latency**
   Floating-point operations inherently require deep pipelining to achieve high clock frequencies. The Iris FFT IP is designed to seamlessly absorb this latency into its internal data scheduling. The arbitrary pipeline latency of the external IPs is communicated to the FFT core via the following Top-Level generics:

   a) **DSP_FP_LAT_ADDSUB** (Defines the latency for both FP_ADD and FP_SUB)
   b) **DSP_FP_LAT_MUL** (Defines the latency for FP_MUL)

   c) **Latency Constraints & Design Recommendation:** The core supports an external IP latency ranging strictly from **1 to 10 clock cycles** (inclusive). Recommendation: The selected latency directly impacts the overall FFT processing time, block RAM utilization (due to internal delay lines), and timing closure. The default values should be carefully selected based on the target device's architecture, the required target clock frequency, and the specific recommendations provided by the silicon vendor's arithmetic library.

4.4 PARALLEL I/O INTERFACES

1) **Overview**
To accommodate high throughput requirements, specific architectural configurations of the IP support parallel data streaming. This allows the core to natively ingress (AXIS_SAMPLES_S) or egress (AXIS_SAMPLES_M) multiple complex samples within a single clock cycle. However, this parallelization is strictly governed by the selected underlying Architecture, Implementation Style (DIF/DIT), and the Base Radix.

2) **Supported Parallel Configurations**
The following table defines the valid parameter combinations for the Slave (Input) and Master (Output) AXI4-Stream interfaces:

ARCHITECTURE	IMPLEMENTATION	AXIS_SAMPLES_S	AXIS_SAMPLES_M
BASE	DIF	[1,2,4,8] [1]	[1]
	DIT	[1]	[1,2,4,8][1]
RSDF	DIF	[1]	[1]
	DIT	[1]	[1]
PIPE	DIF	[2,4,8] [2]	[2,4,8] [2]
	DIT	[2,4,8] [2]	[2,4,8] [2]

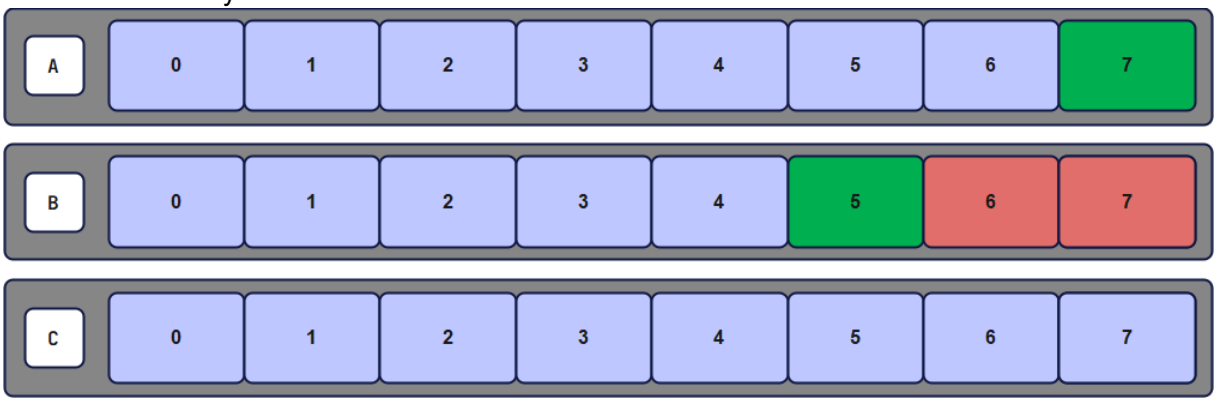
3) **Configuration Constraints**
[1]The maximum parallel sample count is mathematically bounded by the configured BASE_RADIX. For Radix-2, the maximum is 2; for Radix-4, the maximum is 4; and for Radix-8, the maximum is 8.
[2]The SSR (Super-Sample Rate) pipeline explicitly requires the parallel sample count to exactly match the configured BASE_RADIX on both interfaces. For example, if configured for Radix-4, both AXIS_SAMPLES_S and AXIS_SAMPLES_M must be set to exactly 4.

4) **AXI4-Stream Strobing & Framing Constraints**
When the IP is configured to process multiple parallel samples per clock cycle, the AXI-Stream strobe signals (TSTRB) are utilized to mark valid data words (Not Bytes) within the wide bus. To guarantee data integrity, the external driving logic must adhere to the following strict framing rules:
   a) **Continuous LSB-Aligned Strobing:**
   All valid strobes must be completely contiguous and strictly start from the Least Significant Word position (Index 0). Sparse, non-continuous, or offset strobing is mathematically invalid. Any data asserted after the first strobe discontinuity will be automatically discarded by the input framing logic.
   b) **No Empty Transfers:**
   Null transfers (asserting TVALID while all strobes are deasserted) are strictly prohibited and violate the core's internal buffering mechanisms.
   c) **Block Alignment:** The number of valid words presented per clock cycle must align with the architectural block size. Specifically, forwarding an unaligned number of samples (e.g., sending exactly 3 valid samples on a 4-sample interface) constitutes a protocol violation.
   d) **Internal Backpressure & Throttling:** Even on highly parallelized interfaces, the core's internal processing engine may dynamically throttle the data acceptance rate (by deasserting TREADY) if the configured transform size requires a mixed-radix decomposition stage to complete the packet (e.g., executing a residual Radix-2 stage at the end of a predominantly Radix-4 transform).

4.5 ZERO PADDING & PACKET FRAMING

1) **Hardware-Accelerated Zero-Padding**
   The core features an optional, dynamic hardware-accelerated zero-padding mechanism designed to gracefully handle undersized data packets. When enabled, if an early End-of-Packet (EOP) is detected via the AXI4-Stream TLAST signal before the configured transform length (NFFT) is reached, the core automatically pads the remainder of the transform buffer with zeros. To minimize latency, this internal backfilling occurs at the maximum interface speed, injecting AXIS_SAMPLES_S zero-samples per clock cycle
2) **Packet Framing & Exception Handling**
   The IP strictly monitors the alignment between the physical incoming AXI4-Stream data framing and the runtime-configured transform size. Any misalignment is safely handled to prevent bus deadlocks and is captured via the AXI4-Lite status registers:
   a) **TLAST Missing (Late or Absent EOP):** The core successfully ingested the exact number of samples required for the configured transform length, but the final sample lacked the required TLAST assertion. The core prioritizes its internal block counter, safely ignores the missing flag, and immediately commences processing the valid samples. The TLAST_MISSING flag is asserted in the status register to warn the host processor of an upstream framing violation.
3) **TLAST Unexpected (Early EOP):** The core detected a TLAST assertion on the Slave AXI-Stream interface prematurely.
   a) **If Zero-Padding is ENABLED**: The core automatically fills the remaining required data vector with zeros and executes the transform.
   b) **If Zero-Padding is DISABLED**: The core halts the transform execution and waits to consume the remaining required samples from the interface to satisfy the NFFT requirement, disregarding the premature TLAST.
   c) **Note**: In either scenario, the TLAST_UNEXPECTED flag is raised. If the early termination was an intentional architectural choice by the user (e.g., intentionally sending short packet), this flag can be safely ignored or cleared via software.

4) **Framing Examples -** The following diagram illustrates the core's framing behavior across different boundary conditions:



a) **Case A – Ideal Framing (Standard Operation):** The core is configured for an 8-point FFT. Exactly 8 valid samples are transferred, and the TLAST signal correctly accompanies the final (8th) sample. The packet is processed seamlessly with no status flags raised.

b) **Case B – Early Termination (Unexpected TLAST):** The core is configured for an 8-point FFT, but TLAST is prematurely asserted on the 6th sample. This instantly triggers the TLAST_UNEXPECTED condition. If zero-padding is enabled, the core autonomously flushes the remaining 2 positions (samples 7 and 8) with zeros and begins processing. If disabled, the core stalls processing and waits for 2 additional valid samples from the upstream logic.

c) **Case C – Absent Termination (Missing TLAST):** The core is configured for an 8-point FFT and receives exactly 8 valid samples, but the TLAST signal is entirely absent on the 8th sample. This triggers the TLAST_MISSING event. The core autonomously assumes the packet is complete and initiates processing to prevent a pipeline stall.

4.6 CYCLIC PREFIX INSERTION

1. **Overview & Application**
   Cyclic Prefix (CP) insertion is a hardware-accelerated feature designed primarily for OFDM-based telecommunication and SDR systems (e.g., 5G, LTE, WLAN). When enabled, the core autonomously duplicates the final NCP samples of a computed transform (typically an IFFT) and prepends them to the beginning of the output packet. This hardware automation offloads the surrounding logic from managing complex buffer routing to mitigate Inter-Symbol Interference (ISI).

2. **Dynamic Configuration**
   The length of the Cyclic Prefix (NCP) is not statically bound. It can be dynamically reconfigured on a per-packet basis via the AXI4-Lite control interface or the native RTL CFG_PREFIX port.

3. **Architectural Impacts & Resource Costs**
   Because the CP operation inherently alters the linear flow of output data, its hardware footprint varies significantly based on the selected core architecture:

   a. **BASE Architecture (DIT Implementation Only):** Due to the iterative nature of the BASE core, this feature is **virtually resource-free**. By utilizing intelligent read-pointer manipulation within the centralized RAM, the controller natively fetches the prefix before flushing the rest of the transform, requiring absolutely zero additional memory primitives. (**Note**: CP is not supported in BASE DIF implementations).

   b. **RSDF & PIPE Architectures:** In these continuous-streaming topologies, data is egressed immediately as it is computed. To prepend the physical "tail" of a stream to its "head", the core must inherently buffer the output. Consequently, enabling CP in pipelined architectures will automatically infer additional deep memory buffers (BRAMs/URAMs), directly increasing the overall resource footprint and core latency.

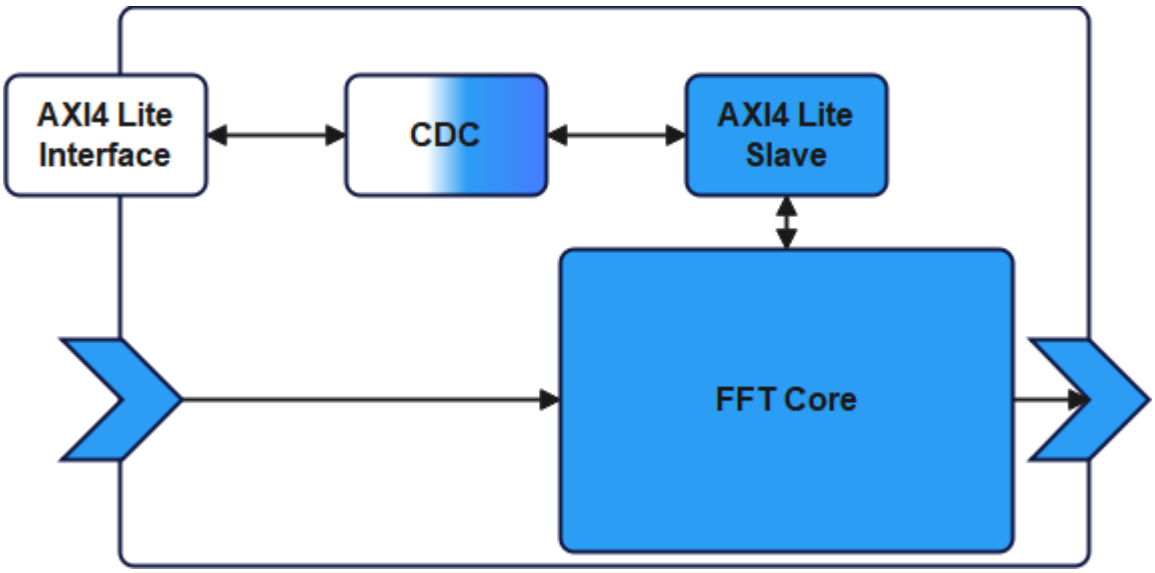
4. **Known Limitations & Constraints**

   a. **Natural Order Prerequisite:** Cyclic Prefix insertion inherently demands that the underlying output data is linear. Therefore, the CP feature must be disabled for configurations, that do bypass the bit/digit-reversal engines.
   b. **Length Boundary**: The configured CP length (NCP) must strictly be less or equal to the currently configured transform size (NFFT).

5. **Architectural Insight: Throughput & AXI4-Stream Backpressure**
   By prepending N_{CP} samples, the total egress packet length inherently expands to $N_{FFT} + N_{CP}$ samples. Because the core must spend additional clock cycles flushing this expanded packet to the M_AXIS interface, the effective continuous throughput into the IP is reduced accordingly as the core dynamically de-asserts S_AXIS_TREADY signal.

4.7 CLOCK DOMAIN CROSSING

1) **Dual-Clock Architecture**
   The IP is designed with a flexible, dual-clock architecture to seamlessly integrate into complex environments. The core utilizes two distinct clock domains:
   a) **FACLK: (Processing Clock):** The primary, high-speed clock driving the internal DSP data path, scaling logic, and AXI4-Stream data interfaces
   b) **ACLK (Control Clock):** The standard clock driving the AXI4-Lite configuration and status interface.
2) **Optional Built-In CDC Synchronization**
   To simplify system integration when FACLK and ACLK operate asynchronously, the core features an optional, integrated Clock Domain Crossing (CDC) module dedicated to the AXI4-Lite interface.
   a) Asynchronous Mode (AXIL_CDC_EN == 1):
   Safely bridges the AXI4-Lite transactions across the asynchronous clock boundary. The number of synchronization flip-flop stages is fully parameterized via the generic configuration (**AXIL_CDC_STAGES**), defaulting to 3 stages to ensure a robust Mean Time Between Failures (MTBF) for standard industrial applications.
   b) Synchronous Mode (AXIL_CDC_EN == 0):
   If both clock domains are synchronous (i.e., ACLK and FACLK are driven by the same clock source), the CDC logic can be entirely bypassed and optimized away during synthesis, saving logic resources and eliminating control latency.



3) **Architectural Insight: Transaction Latency**
   Enabling the AXI4-Lite CDC mechanism inherently introduces a synchronization delay to the control bus. System architects should account for this additional latency during AXI read and write operations, as the handshaking signals must safely cross the synchronization stages in both directions.
4) **Native Direct Control & Status Interface**
   While the AXI4-Lite interface provides an industry-standard method for memory-mapped control, the Iris FFT IP also natively supports a lightweight, low-latency Direct Hardware

Interface. This is specifically designed for deeply embedded systems or custom Finite State Machines (FSMs) where the overhead of an AXI4-Lite interconnect is undesirable.

By setting the Top-Level generic AXIL_INIT_MODE to FFT_CNTRL_RTL, the AXI4-Lite interface can be entirely bypassed. The core then shifts control to a dedicated native port configuration, allowing for fully dynamic, per-packet reconfiguration and status monitoring at the hardware level.

4.8 MEMORY ARCHITECTURE & FOOTPRINT

1) **Internal Memory Topologies**
   The Fast Fourier Transform is inherently a memory-bound algorithm. The IP uses a mixture of RAM and ROM memories for internal data buffering, algorithmic reordering, and phase factor storage. The core strictly utilizes standard architectural compression and pipeline optimization techniques across all its architectures (BASE, RSDF, PIPE) to minimize the physical memory footprint.
2) **Hardware Optimization & Pipeline Registers**
   To guarantee an optimal timing closure and achieve the highest possible maximum clock frequency (FMAX) on modern FPGA fabrics, the core is explicitly designed to automatically infer secondary pipeline registers at the output of all RAM and ROM components.
3) **Physical Primitive Mapping & Benchmarks**
   The precise number of physical hardware primitives (e.g., Block RAMs, UltraRAMs, or M20Ks) required to implement these memory structures is highly variable. It depends entirely on the selected generic configuration (FFT Size, Datapath Width), the target silicon architecture, and the synthesis tool's ability to efficiently pack logical arrays into physical blocks.
   a) **Implementation Results:** For exact resource utilization tables, primitive counts, and validated timing reports across various device families and configurations, please refer to the official **IP Implementation Benchmarks** available on the product website.

4.9 DSP RESOURCE UTILIZATION & FOOTPRINT

1) **Modular Processing Engine**
   The IP is built upon a highly optimized, modular DSP computational engine (comprising mostly the Complex Multiplier block and the Butterfly routing logic). The total physical footprint of the core depends on how the selected top-level architecture instantiates this foundational engine:
   a) **BASE Architecture**: Instantiates exactly one physical engine, which is reused iteratively across all stages. The logic footprint remains highly compact and constant regardless of the configured FFT size.
   b) **RSDF Architecture:** Instantiates exactly one R2 engine per FFT stage. The total multiplier footprint scales logarithmically with the transform size ($\log_2(N)$)
   c) **PIPE Architecture**: Instantiates one R2,R4 or R8 engine per FFT stage. The total multiplier footprint scales logarithmically with the transform size ($\log_R(N)$)

2) **Complex Multiplier (CMUL) Footprint**
   The core utilizes by default a 4-way complex multiplication architecture. Each required CMUL instance inherently consumes 4 dedicated hardware multipliers and 2 adders/subtractors. The number of parallel CMULs required is dictated by the selected Base Radix:

	CMUL	MULT	ADD/SUB
Radix – 2	1	(1x4) = 4	(1 x 2) = 2
Radix – 4	3	(3x4) = 12	(3 x 2) = 6
Radix – 8	7	(7x4) = 28	(7 x 2) = 14

3) **FFT Butterfly Engine Footprint**
   To compute the FFT butterfly itself, the IP employs a highly efficient split-radix approach to process higher-order radixes (R4/R8) with minimal mathematical overhead.

	MULT	ADD/SUB
Radix – 2	0	4
Radix – 4	0	(4 x 4) = 16
Radix – 8	4	(3 x 4 x 4) + 4 = 52

4) **Total Core Arithmetic Summary**
   Combining the CMUL and Butterfly logic, the total baseline arithmetic footprint for a single processing engine instance resolves to:
   a) Radix-2 Configuration: 4 Multipliers + 6 Adders
   b) Radix-4 Configuration: 12 Multipliers + 22 Adders
   c) Radix-8 Configuration: 32 Multipliers + 66 Adders

(**Note**: These baseline values exclude supplementary logic required for sign manipulation, fixed-point rounding, or data scaling, which may infer additional LUTs or DSP slices depending on the synthesis tool and IP configuration.)

5) **Implementation Constraints & Device Mapping:**
   a) **DSP Block Mapping**: Only the multiplier blocks are explicitly marked via VHDL generics to infer dedicated DSP slices. The adders/subtractors are typically mapped to standard logic fabric (LUTs/CARRY chains) unless optimized otherwise by the synthesizer.
   b) **Word Width Limitations**: Large data path widths may require cascading multiple DSP blocks to implement a single mathematical multiplier. For example, the AMD Xilinx

DSP48E2 slice natively supports a 27 x 18-bit multiplication. Consequently, configuring the core with a 32 x 32-bit data path will force the synthesis tool to cascade up to 4 DSP blocks per multiplier, drastically increasing the overall DSP utilization and potentially degrading the maximum clock frequency (FMAX).

c) **Design Recommendation:** To maximize performance and limit resource explosion, it is highly recommended to strictly adhere to data path and twiddle widths that fit within the target silicon's native DSP slice dimensions

4.10BASE-RADIX ARCHITECTURE

1) Processing Flow & Datapath

The BASE Architecture implements a resource-efficient, iterative processing engine built around a centralized internal RAM. The operation follows a strict, four-phase sequential state machine:

a) **Ingress (Data Load):** Data received over the Slave AXI4-Stream (S_AXIS) interface is continuously written into the internal RAM buffer. During this phase, the core's processing engine is halted.

b) **Execution (In-Place Processing):** Once the complete packet (defined by the configured FFT size and the TLAST marker) is safely buffered, the input interface asserts backpressure (S_AXIS_TREADY is deasserted) to prevent data overwrite. The controller then executes the FFT stages sequentially. For each stage, it reads intermediate data from the RAM, fetches the corresponding Twiddle factors from the ROM, computes the complex butterfly operations, applies the designated scaling/truncation, and writes the results back into the RAM array in-place.

c) **Cyclic Prefix Insertion (Optional, DIT Only):** Following the completion of all transform stages, if the Cyclic Prefix (CP) feature is enabled and the core is configured for the DIT implementation, the state machine initiates the CP protocol. Instead of physically copying data, the controller dynamically manipulates the internal RAM read pointers to fetch and egress the final N_{CP} samples before unloading the primary transform data, effectively prepending the prefix without incurring any additional memory buffering overhead

d) **Egress (Data Unload):** The core flushes the fully processed transform data from the internal RAM to the Master AXI4-Stream (M_AXIS) output. Once the full packet egress is complete (including the prepended Cyclic Prefix, if applicable), the core releases the input backpressure and is immediately ready to accept a new packet.

4.11 RSDF ARCHITECTURE

1. **Overview & Continuous Streaming**
   The RSDF Architecture is specifically engineered for high-throughput applications requiring continuous, uninterrupted data streaming. Unlike the iterative BASE topology, the RSDF architecture processes data "on the fly," natively supporting a continuous ingress and egress rate of exactly 1 complex sample per clock cycle (AXIS_SAMPLES_S == AXIS_SAMPLES_M == 1).

2. **Pipelined Datapath & Unrolled Processing**
   To achieve this continuous throughput, the RSDF topology completely unrolls the FFT algorithm into a deep physical pipeline:

   a. **Dedicated Stage Resources**: Unlike iteratively using a single DSP engine in the BASE Radix Architecture, the RSDF instantiates the same DSP Processing block across all required stages. Each stage is equipped with its own dedicated Radix-2 Butterfly computing engine, independent Twiddle Factor ROM, and internal RAM tailored for the given stage.
   b. **Concurrent Transform Execution**: The architecture might consume several configuration packets in advance to fully utilize its DSP pipeline. This is a standard and expected behavior. It is not possible to cancel a transform that is already scheduled for execution.

3. **Performance Guarantees & Constraints**
   a. **Throughput Guarantee**: The core guarantees 100% continuous streaming throughput (no internal backpressure or dead-cycles between packets) strictly for transform sizes greater than or equal to 8 points at minimum.
   b. **Small Transform Stalls**: Executing ultra-small transforms ($N_{FFT} < 8$) back-to-back may introduce inherent pipeline stalls. The core fully supports these smaller sizes and will compute them correctly, but continuous 1-sample-per-clock throughput is not mathematically guaranteed, and the core may dynamically throttle the input via S_AXI_TReady.
   c. **Standard Features Integration**: The RSDF data path uniformly supports all universal core features, including Front-end Zero Padding, invalid packet signalization, and dynamic Datapath Scaling (Shifting) across both DIF and DIT implementations.

4. **Cyclic Prefix (CP) Support in RSDF**
   The hardware-accelerated Cyclic Prefix feature is fully supported within the RSDF architecture. However, designers must be aware of the architectural trade-offs:
   a. **Memory Overhead**: Because data is streaming continuously, prepending the cyclic prefix requires the core to infer additional deep memory buffers at the egress stage to temporarily hold the stream's "tail" before it can be transmitted. If CP is not required for the application, it should be disabled via generics to conserve silicon area.
   b. **Digit-Reversal Dependency**: As with all continuous architectures, the Cyclic Prefix logic requires the output data to be linear. Consequently, CP insertion is strictly prohibited if the core's internal bit/digit-reversal engine is bypassed.

4.12PIPE ARCHITECTURE

1. **Overview & Super-Sample Rate (SSR) Paradigm**

   The PIPE Architecture represents the highest performance tier of the IP, designed strictly for ultra-broadband and multi-gigahertz throughputs. Fundamentally diverging from both the iterative BASE and the standard streaming RSDF topologies, this architecture implements a proprietary **Super-Sample Rate (SSR)** processing engine.

   Beneath the surface, the core utilizes the same highly optimized DSP computational blocks, but instantiates them to form a massively parallel, multi-lane data path. The internal routing and stage-to-stage data communication interfaces are designed similarly to modern Network-on-Chip (NoC) crossbars with performance in mind, allowing the core to process multiple complex samples concurrently within a single clock cycle at the highest rate.

2. **Strict I/O Framing & Strobe Management**
   Because the core ingests and egresses an entire vector of samples per clock cycle, certain protocol rules have to be obeyed to allow the core to operate as described.

   a. **AXIS Bus Sizing:** The AXI4-Stream parallel sample count must strictly match the selected algorithmic Base Radix: BASE_RADIX == AXIS_SAMPLES_S == AXIS_SAMPLES_M.
   b. **Data Validity & Strobing:** Unlike single-sample architectures, downstream slaves must actively monitor the M_AXIS_TSTRB (Strobe) signal. Depending on the configured transform size and IP parametrization, not all parallel words presented on the wide data path bus may contain valid transform data. The core however guarantees certain strobing patterns:

      i. **No Null Transfers (Empty Strobes):** The core never forwards any empty strobes.
      ii. **Contiguous LSB-Aligned Strobing**: All strobes transmitted from the core start at index 0.
      iii. **Power-of-2 Word Alignment:** The core only asserts strobes for exactly 2, 4, or 8 valid words; odd or non-power-of-2 quantities (1, 3, 5, 6, 7) are never generated.

3. **Performance Guarantees & Mixed-Radix Penalties**
   Similar to the RSDF architecture, the SSR pipeline may consume multiple configuration packets in advance to keep its massive data path fully saturated.
   a. **Throughput Guarantee:** To guarantee continuous streaming with zero backpressure, the configured transform size (N_{FFT}) must meet the following minimums (**Note**: Smaller sizes are fully supported as well and will compute accurately, but will inherently throttle the S_AXIS interface):
      i. **Radix-2:** $N_{FFT} >= 64$
      ii. **Radix-4:** $N_{FFT} >= 128$
      iii. **Radix-8:** $N_{FFT} >= 256$
   b. Mixed-Radix Penalties:

To achieve 100% utilization of the SSR hardware, the transform size must be a pure mathematical power of the configured Base Radix (R^N). If a "Mixed-Radix" transform is executed (e.g., executing a Radix-2 fallback stage on an 8-lane Radix-8 architecture), the wide data path is forced to operate at fractional capacity.

i. **Example**: Executing an 8^N x 4 decomposition on a Radix-8 core drops the sustainable throughput to 50%. Executing an 8^N x 2 decomposition drops it drastically to 25% as the 8-lane hardware waits for a 2-lane operation to complete.

4. SSR Cyclic Prefix (CP) Constraints

The hardware Cyclic Prefix is fully supported in the PIPE architecture, retaining the same physical memory overheads and digit-reversal dependencies as the RSDF core. However, due to the multi-lane nature of the data path, the configured Cyclic Prefix length (N_{CP}) must be an exact integer multiple of the Base Radix.

5. **Hardware Optimization Insight: Managing Wide-Bus Synthesis**

The SSR architecture inherently relies on exceptionally wide data paths and massive combinatorial commutators to route parallel samples. System architects are strongly advised to think critically about their requested **DSP_DATA_WIDTH**, as overly wide words multiplied across 8 parallel lanes will cause rapid routing congestion and FMAX degradation on the FPGA fabric

a. **Synthesis Tip:** To alleviate routing pressure, always configure the **MIN_SIZE** generic to the actual minimum transform size your application requires. The core utilizes this parameter during synthesis to aggressively prune out unnecessary wide multiplexers, which might drastically improve timing closure and save logic resources.

4.13 LATENCY AND PERFORMANCE

1) **Mathematical Latency Estimation**
   The theoretical minimum latency is dictated by the mathematics of the Cooley-Tukey algorithm. The number of required processing stages (S) for a given transform size (NFFT) and Base Radix (R) is calculated as:

$$S = log_R(N_{FFT})$$

   During each stage, the core computes NFFT/R butterfly operations. Since the BASE architecture computes exactly one butterfly per clock cycle, the absolute theoretical minimum execution latency (L_{IDEAL}) in clock cycles is

$$L_{IDEAL} = \left(\frac{N_{FFT}}{R}\right) x\ S$$

2) **Hardware Performance & Latency Benchmarks**
   Due to the physical realities of the hardware - such as RAM read / write access times, DSP pipeline filling / draining, internal handshaking and state machine transitions - the actual clock cycle latency will always exceed the theoretical minimum.

   Because latency metrics vary significantly based on the selected IP configuration (Fixed-Point vs. Floating-Point, DIF vs. DIT, and the chosen Architecture), exhaustive cycle-accurate latency tables are maintained interactively on the product website.

3) **Performance Optimization Guidelines:**
   a) **Mixed-Radix Penalties:** Transforms that are not a pure mathematical power of the configured Base Radix (e.g., executing a 32-point FFT on a Radix-4 core, calculated as 4x4x2) incur a processing penalty. To maximize throughput, it is highly recommended to strictly utilize transform sizes that match the R^N form.
   b) **DIF vs. DIT Execution:** The Decimation-in-Time (DIT) architecture is generally slower in Fixed-Point implementation due to the mandatory insertion of additional pipeline delay stages required to manage the data scaling.

4.14 GENERIC PARAMETERS

1) Core Architecture & Capabilities

ARCH_TYPE	String	Top-level architecture selection: "BASE", "RSDF", or "PIPE"
IMPL_TYPE	String	Algorithm implementation style: "DIT" or "DIF"
BASE_RADIX	Natural	Mathematical base radix: 2, 4, or 8
MIN_SIZE	Natural	Minimum Supported Size (N_{FFT})
MAX_SIZE	Natural	Maximum Supported Size (N_{FFT})
CYCLIC_PREFIX_EN	Natural	Enables hardware Cyclic Prefix logic (0 = Disabled, 1 = Enabled)

2) DSP Arithmetic & Datapath Precision

DSP_DATA_WIDTH	Natural	Data width for both Real and Imaginary components
DSP_TWIDDLE_WIDTH	Natural	Twiddle factor (Phase) resolution width
DSP_PRECISION	String	Processing arithmetic type: "FIXED" or "FLOAT"
DSP_FI_SCALE_WIDTH	Natural	Width of the Fixed-Point internal scaler configuration
DSP_FI_ROUND_MODE	Rounding_t	Fixed-Point rounding mode: TRUNCATE, ROUND_HALF_UP, or ROUND_HALF_EVEN
DSP_FI_MULT_ARCH	String	Complex multiplier DSP topology: "CMUL_X3" or "CMUL_X4"
DSP_FI_DR_MASK_EN	Natural	Enables Fixed-Point dynamic range extraction logic
DSP_FP_LAT_MUL	Natural	Latency of the external Floating-Point Multiplier IP
DSP_FP_LAT_ADDSUB	Natural	Latency of the external Floating-Point Adder/Subtractor IP

3) AXI4-Stream Interfaces

AXIS_SAMPLES_S	Natural	Number of parallel complex samples on the Slave (Input) interface
AXIS_SAMPLES_M	Natural	Number of parallel complex samples on the Master (Output) interface

4) AXI4-Lite Control & Configuration Interface

AXIL_INIT_MODE	SL	Selects the active control interface: FFT_CNTRL_AXI or FFT_CNTRL_RTL
AXIL_ADDR_WIDTH	NATURAL	AXI4-Lite memory-mapped address bus width
AXIL_DATA_WIDTH	NATURAL	AXI4-Lite data bus width (Must be strictly set to 32)
AXIL_CDC_EN	NATURAL	Enables built-in Clock Domain Crossing (CDC) for the control interface
AXIL_CDC_STAGES	NATURAL	Number of synchronization flip-flop stages for the AXI4-Lite CDC

5) Hardware Synthesis Attributes

ATTR_RAM_TYPE	STRING	Preferred implementation primitive for internal Data RAM
ATTR_ROM_TYPE	STRING	Preferred implementation primitive for Twiddle ROM
ATTR_USE_DSP	STRING	Enforces explicit DSP slice inference for multipliers

4.15 PORT DEFINITIONS

1. **Clocks & Resets**

ACLK	AXI4 – Lite Infrastructure clock	N/A
FACLK	Primary Core / DSP Processing Clock	N/A
ARESETn	AXI4-Lite Infrastructure Reset (Active Low, Synchronous)	ACLK
FRESETn	Primary Core Reset (Active Low, Synchronous)	FACLK

2. **AXI4-Lite Read Address Channel**

S_AXIL_ARADDR	Read Address. This is the address that the master wants to read from	ACLK
S_AXIL_ARREADY	Read Address Ready. The slave asserts this to indicate that it's ready to accept a new read address	ACLK
S_AXIL_ARVALID	Read Address Valid. The master asserts this to indicate that a valid read address is available	ACLK

3. **AXI4-Lite Read Data Channel**

S_AXIL_RDATA	Read Data. This is the data that the slave is sending to the master in response to a read request	ACLK
S_AXIL_RRESP	Read Data Status. This indicates the status of the read transaction.	ACLK
S_AXIL_RVALID	Read Data Valid. The slave asserts this to indicate that valid read data and response are available	ACLK
S_AXIL_RREADY	Read Data Ready. The master asserts this to indicate that it's ready to accept the read data and response	ACLK

4. **AXI4-Lite Write Address Channel**

S_AXIL_AWADDR	Write Address. This is the address that the master wants to write to	ACLK
S_AXIL_AWREADY	Write Address Ready. The slave asserts this to indicate that it's ready to accept a new write address	ACLK
S_AXIL_AWVALID	Write Address Valid. The master asserts this to indicate that a valid write address is available	ACLK

5. **AXI4-Lite Write Data Channel**

S_AXIL_WDATA	Write Data. This is the data that the master is writing to the slave	ACLK
S_AXIL_WSTRB	Write Data Byte Strobes These signals indicate which bytes of the WDATA bus are valid	ACLK
S_AXIL_WVALID	Write Data Valid. The master asserts this to indicate that valid write data and strobes are available	ACLK
S_AXIL_WREADY	Write Data Ready. The slave asserts this to indicate that it's ready to accept the write data	ACLK

6. **AXI4-Lite Write Response Channel**

S_AXIL_BREADY	Response Ready. The master asserts this to indicate that it's ready to accept the write response	ACLK
S_AXIL_BRESP	Response Status. This indicates the status of the write transaction	ACLK
S_AXIL_BVALID	Response Valid. The slave asserts this to indicate that a valid write response is available	ACLK

7. AXI4-Stream Data Receive Interface

S_AXI_TLast	Transfer Last: Master asserts this signal on the last beat of a packet	FACLK
S_AXI_TReady	Transfer Ready: Core asserts it is ready to accept data	FACLK
S_AXI_TVld	Transfer Valid: Master asserts that valid data is present on the bus	FACLK
S_AXI_TStrb	Transfer Strobe: Indicates which data words within the bus are valid	FACLK
S_AXI_TData_Real	Transfer Data: Real component(s)	FACLK
S_AXI_TData_Imag	Transfer Data: Imaginary component(s)	FACLK

8. AXI4-Stream Data Transmit Interface

M_AXI_TReady	Transfer Ready: Downstream slave asserts it is ready to accept data	FACLK
M_AXI_TVld	Transfer Valid: Core asserts that valid data is present on the bus	FACLK
M_AXI_TLast	Transfer Last: Core asserts this signal on the last beat of a packet	FACLK
M_AXI_TStrb	Transfer Strobe: Indicates which data words within the bus are valid	FACLK
M_AXI_TData_Real	Transfer Data: Real component(s)	FACLK
M_AXI_TData_Imag	Transfer Data: Imaginary component(s)	FACLK

9. Dynamic RTL – Reconfiguration Interface

CFG_VALID	Core Reconfiguration Handshake	FACLK
CFG_READY	Core Reconfiguration Handshake	FACLK
CFG_SIZE	Transform Size	FACLK
CFG_SCALES	Scaling / Shift Schedule	FACLK
CFG_PREFIX	Cyclic Prefix Size	FACLK
CFG_DIRECTION	Transform Direction	FACLK
CFG_PADDING_EN	Enables padding of packets	FACLK
CFG_SATURATION_EN	Enables saturation for scalers	FACLK
ST_OVF_REAL	Signalizes overflow for real data type	FACLK
ST_OVF_IMAG	Signalizes overflow for imag data type	FACLK
ST_TLAST_MISSING	Tlast Missing during packet reception	FACLK
ST_TLAST_UNEXPECTED	Tlast unexpected during packet reception	FACLK

4.16 MEMORY MAP

The IP's Memory map requires 4KB of memory-mapped space ands is divided into several regions where each region has a fixed amount of **32 x 32-bit** registers as described in the address decode below:

Mapping:	**Region**	**Region's Registers**	**Byte Alignment**
Address Bits:	[11:7]	[6:2]	[1:0]

1. Base Region: 0x000 (Byte Offset 0x00)
2. Bitmask Region: 0x001 (Byte Offset 0x80)

4.16.1 Base Region

Address	**Register**	**Bits**	**Description**	**Reset Value**
0x00	CORE_ID	[31:0]	Reads as 0x65796573	N/A
0x04	CORE_VERSION	[15:0]	IP Minor Version Number	N/A
		[31:16]	IP Major Version Number	N/A
0x08	GENERIC_0	[7:0]	DSP_DATA_WIDTH	N/A
		[15:8]	DSP_TWIDDLE_WIDTH	N/A
		[19:16]	Log_2(BASE_RADIX)	N/A
		[24:20]	Log_2(MIN_SIZE)	N/A
		[31:25]	Log_2(MAX_SIZE)	N/A
0x0C	GENERIC_1	[7:0]	DSP_FP_LAT_MUL	N/A
		[15:8]	DSP_FP_LAT_ADDSUB	N/A
		[19:16]	AXIS_SAMPLES_S	N/A
		[23:20]	AXIS_SAMPLES_M	N/A
		[27:24]	DSP_FI_SCALE_WIDTH	N/A
		[31:25]	DSP_FI_ROUND_MODE 0 → TRUNCATE 1 → ROUND_HALF_UP 2 → ROUND_HALF_EVEN	N/A
0x10	GENERIC_2	[1:0]	ARCH_TYPE 0 → BASE 1 → PIPE 2 → RSDF	N/A
		[2]	IMPL_TYPE 0 → DIT 1 → DIF	N/A
		[7:4]	DSP_PRECISION 0 → Fixed - Point 1 → Floating Point	N/A
0x14	GENERIC_3	[31:0]	Reserved	N/A
0x18	CNTRL	[3:0]	FFT Transform Size[1]	0x0
		[22]	Control Mode 0 → AXI4-Lite 1 → RTL - Based	Bound by Generics
		[23]	FI Mask Freeze	0x0
		[24]	FI Mask Reset	0x0
		[25]	FFT Padding Enable	0x0
		[26]	RX / TX / Drop Sample count reset	0x0

		[27]	Single Step compute: Process 1 transform and halt (Unless BIT 28 is set)	0x0
		[28]	Continuous Processing Enable	0x0
		[29]	FFT Direction 0 → Forward FFT Transform 1 → Inverse FFT Transform	0x0
		[30]	Status Latch Reset: Write 1 to clear OVF/TLAST flags	0x0
		[31]	Saturation Enable: (Fixed-Point only)	0x0
0x1C	SCALE_0	[31:0]	Scale Value [31:0][2]	0x0
0x20	SCALE_1	[31:0]	Scale Value [63:32] [2]	0x0
0x24	SCALE_2	[31:0]	Scale Value [95:64] [2]	0x0
0x28	SCALE_3	[31:0]	Scale Value [127:96] [2]	0x0
0x2C	CYCLIC_PREFIX	[31:0]	Cyclic Prefix Size	0x0
0x30	STATUS	[0]	Processing Ready[3]	0x1
		[1]	Real Data Overflow (Latched)	0x0
		[2]	Imaginary Data Overflow (Latched)	0x0
		[3]	TLAST Missing Means that the core has received enough samples to start the transform, but no end-of-packet has been delivered to the core.	0x0
		[4]	TLAST Unexpected Means that the core received end-of-packet flag prior having enough samples to start the transform.	0x0
0x34	RX_COUNT	[X:0] [4]	Amount of samples received on S-AXIS	0x0
0x38	TX_COUNT	[X:0] [4]	Amount of samples send on M-AXIS	0x0
0x3C	DROP_COUNT	[X:0] [4]	Amount of dropped words on S-AXIS due to protocol violation.	0x0

1. **Transform Size Encoding (CNTRL[3:0])**
   The desired transform size (N_{FFT}) must be written to the control register as an encoded index, rather than the raw point size. The core samples this configuration strictly at the start of a new packet. The mathematical relation is defined as:
   $$\lceil log_2(N_{FFT}) \rceil - 1$$
2. **FSM Handshake** (Processing Enable vs. Ready) : Firmware must adhere to the following sequence to govern the core's state machine safely:
   a. After hardware reset, the core remains inactive. The STATUS[0] (Processing Ready) bit defaults to 1, indicating the IP is idle.
   b. The host writes configuration data (Size, Direction, Scales) to the corresponding registers.
   c. The host asserts CNTRL[28] (Processing Enable) to 1.
   d. The core immediately de-asserts STATUS[0] to 0 (Busy) and begins accepting / processing AXI-Stream data.
   e. If CNTRL[28] remains 1, the core continuously processes back-to-back packets as soon as possible. If it is de-asserted, the core will finish the currently active packet, halt, and re-assert STATUS[0] to 1

3. **Shift / Scaling Schedule (SCALE_0123)**
   For Fixed-Point implementations, the internal bit-growth scaling is managed via a virtual vector distributed across four 32-bit registers (SCALE_0 to SCALE_3). The vector is divided into contiguous fields of DSP_FI_SCALE_WIDTH bits. Each field dictates the physical right-shift value applied at the output of a sequentially corresponding FFT stage.

   a. Bits [(1 x W-1) : 0 x W]: Shift value for FFT Stage 1
   b. Bits [(2 x W-1) : 1 x W]: Shift value for FFT Stage 2
      ... (where W = DSP_FI_SCALE_WIDTH)

4. **Diagnostic Counters** (RX_COUNT, TX_COUNT, DROP_COUNT)
   To strictly preserve FPGA resources in production environments, the diagnostic sample and drop counters are controlled via the core's public VHDL package definition, rather than entity generics. If TOP_SAMPLE_COUNTS or TOP_DROP_COUNT are set to 0 in the package configuration, the associated AXI4-Lite registers will be physically tied off and read as zero.

4.16.2 Bitmask Region

This region is read-only and dedicated to dynamic - range / bitmask estimation for a fixed-point arithmetic's across various FFT stages if the feature is enabled through the generics of the IPs. Each register's data width is equal to the Data width the IP is configured to use. The below table shows the registers used:

Address Byte Offset	FFT Stage ID Value
0x00	0
0x04	1
0x08	2
0x0C	3
0x10	4
…….	…….