Neosyn · IP Core

DATASHEET · REV A (PRELIMINARY)

Fixed-Point Divider & Barrel Shifter

Q16.16 division and runtime shifts in portable logic

• PRODUCTION READY

Product code: NSN-DSP-DIVSHIFT · Implemented in C⏚ · Source-included · Verilog / VHDL output

The Neosyn Divider core supplies the integer-math building blocks an FPGA datapath needs when a plain hardware / is unavailable or too expensive: a fixed-point Q16.16 divider and a 32-bit barrel shifter that shifts by a run-time amount.

The divider is shipped in two implementations of the same restoring long-division algorithm, so the area/latency trade-off is a design choice rather than a rewrite. The combinational variant unrolls all 48 steps and returns a quotient every cycle. The sequential variant reuses a single bit-serial stage under an FSM: roughly 40× fewer LUTs for roughly 48× the latency.

Every block is pure portable logic — no vendor divider primitive and no DSP block — so the core synthesizes unchanged on any FPGA, including small DSP-less devices. Supplied as readable C⏚ source, so the word width and fixed-point format can be retargeted.

KEY FEATURES

- Q16.16 division
- Combinational & sequential
- ≈40× area trade-off
- 32-bit barrel shifter
- SLL / SRL / SRA
- No DSP blocks
- No vendor primitives
- Source-included

— Architecture

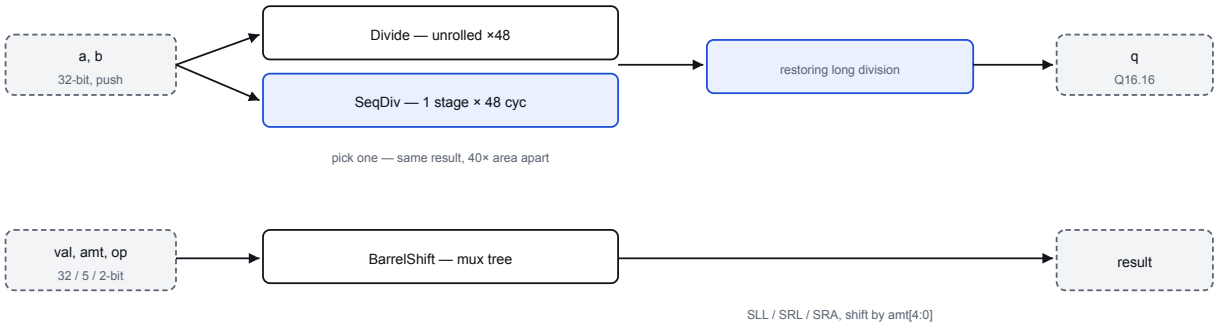

Figure 1. Core organization. The two divider variants implement the same Q16.16 restoring long division and are interchangeable at the interface; only area and latency differ. The barrel shifter is an independent combinational block.

1 Overview

The divider computes q = (a << 16) / b — that is, the quotient a/b expressed in Q16.16 fixed point. Integer inputs therefore yield the rational a/b as a Q16.16 value, and Q16.16 inputs yield the Q16.16 quotient. Operands are treated as non-negative magnitudes.

Both variants derive from one bit-serial restoring long-division step. The combinational variant unrolls 48 of those steps into a single long combinational path; the sequential variant instantiates the step once and iterates it under an FSM, reading operands only when idle and asserting its result when the division completes.

The barrel shifter performs a logical-left, logical-right, or arithmetic-right shift of a 32-bit value by a run-time amount, as a mux tree that conditionally shifts by each power of two. All internal shifts are by literal constants, which is what makes it portable RTL.

2 Features

- **Format.** Q16.16 fixed point; q = (a << 16) / b on 32-bit operands.
- **Two divider variants.** Combinational (1 result/cycle) and sequential (≈40× smaller, ≈48 cycles), interface-compatible.
- **Barrel shifter.** 32-bit SLL / SRL / SRA by a run-time amount amt[4:0]; one result per cycle.
- **Portability.** No DSP blocks and no vendor primitives — synthesizes on any FPGA, including DSP-less parts.
- **Flow control.** Push handshake throughout; the sequential divider back-pressures its producer.
- **Adaptability.** Readable C⏚ source — retarget word width or fixed-point format.

3 Functional description

Refer to Figure 1.

3.1 Combinational divider (Divide)

All 48 restoring-division steps are unrolled into one combinational path, producing a quotient every cycle. This is the area-heavy choice: roughly 10,050 LUTs and 33 flip-flops, with a long combinational path that will dominate timing closure. Use it when throughput matters more than area.

3.2 Sequential divider (SeqDiv)

A single bit-serial stage is iterated 48 times by an FSM, giving an identical result for roughly 243 LUTs and 198 flip-flops — about a 40× LUT reduction. Operands are accepted only when the divider is idle and the quotient is asserted on completion, so push back-pressure lets a producer stream operands at the divider's natural rate.

3.3 Barrel shifter (BarrelShift)

A mux tree conditionally shifts by 1, 2, 4, 8 and 16, each stage gated by one bit of amt[4:0], giving any shift of 0..31 in one cycle. op selects SLL (0), SRL (1) or SRA (2); SRA replicates the sign bit. Roughly 229 LUTs and 33 flip-flops.

> **Note.** The divider operates on non-negative magnitudes. Apply and re-apply operand signs externally if signed division is required. Resource figures are yosys 0.33 `synth_xilinx` estimates against 7-series primitives, not a place-and-route report; treat them as relative guidance and re-characterize for your target.

4 Interfaces & signals

Each block is an independent task with a push handshake on every port.

GROUP	DIRECTION	DESCRIPTION
a, b	in	Divider operands, 32-bit, push handshake
q	out	Q16.16 quotient, 32-bit
val, amt	in	Shifter value (32-bit) and shift amount (5-bit)
op	in	Shift operation — 0 = SLL, 1 = SRL, 2 = SRA
result	out	Shifted value, 32-bit
Clock / reset	in	Core clock domain and synchronous reset

Detailed signal-level pinout (per-bit widths, polarities, timing) is available on request and ships with the core.

5 Arithmetic definition

ITEM	DETAIL
Quotient	q = (a << 16) / b
Format	Q16.16 fixed point
Operands	Non-negative magnitudes, 32-bit
Algorithm	Restoring long division, 48 steps
Shift ops	SLL / SRL / SRA, 0..31
Handshake	Push (valid + back-pressure)

6 Performance

PARAMETER	VALUE	NOTES
Divide — throughput	1 result / cycle	combinational
Divide — area	≈10,050 LUTs, 33 FF	yosys synth_xilinx
SeqDiv — latency	≈48 cycles / result	sequential
SeqDiv — area	≈243 LUTs, 198 FF	yosys synth_xilinx
BarrelShift	1 result / cycle, ≈229 LUTs	yosys synth_xilinx
DSP blocks / BRAM	None	portable to DSP-less parts
Max clock (f_{MAX})	Available on request	per target device

7 Resource utilization

Representative utilization per target FPGA family is provided on request from current synthesis reports.

TARGET FAMILY	LUTS / CELLS	REGISTERS	BLOCK RAM	F_{MAX}
Lattice ECP3	on request	on request	on request	on request
AMD/Xilinx 7-series	on request	on request	on request	on request
Intel Cyclone	on request	on request	on request	on request

Figures are supplied per project from characterised synthesis runs to avoid quoting unverified numbers.

8 Verification & validation

- Q16.16 known-answer tests on the combinational divider (7/7) and on the sequential divider (7/7), plus SLL/SRL/SRA known-answer tests on the barrel shifter (8/8).
- Every test is run on both backends — cycle-accurate Verilog simulation (Icarus) and the C⏚ simulator — as an automated regression gate on every build.
- Known-answer vectors are computed independently of the hardware by a reference script shipped with the core (tools/divider_kat.py).

9 Deliverables

- C⏚ source for the core (readable, modifiable)
- Generated synthesizable Verilog (VHDL on request)
- Self-checking testbench
- Integration guide and this datasheet
- Email integration support per the licensed tier

10 Ordering & licensing

ITEM	DETAIL
Product code	NSN-DSP-DIVSHIFT
License	Single-project or perpetual; full C⏚ source included
Pricing	Quoted per use (project, volume, support tier) — contact Neosyn
Support	Email integration support; custom development available
Contact	neosyn.io/contact · info@neosyn.io

11 Revision history

REV	DATE	CHANGE
A	2026	Preliminary datasheet (Neosyn / C⏚ release).

Disclaimer. This document is preliminary and provided for information only. Specifications, features and figures are subject to change without notice. Resource, timing and latency figures marked “available on request” are supplied per target device from characterised synthesis reports. The IP core is licensed, not sold, and is described as hardware; it is not certified for safety- or life-critical use and is used at the licensee’s own risk. All trademarks are the property of their respective owners and are referenced for descriptive purposes only. © 2026 Neosyn. All rights reserved.