Neosyn · IP Core DATASHEET · REV A (PRELIMINARY)

SHA-256

SHA-256 hash core — three implementation variants

● PRODUCTION READY

Product code: NSN-SEC-SHA256 · Implemented in C⏚ · Source-included · Verilog / VHDL output

The Neosyn SHA-256 core computes the SHA-256 cryptographic hash (FIPS 180-4). It is supplied in three implementation variants — a straightforward basic version, a ROM-optimized version, and a shift-register version — which together demonstrate a complete size/speed optimization path in C⏚.

The optimized variant achieves roughly a 4× area reduction over the basic version with fewer multiplexers and simpler logic, while producing identical results.

Written from the standard specification in readable C⏚ and shipped with source, the core doubles as a worked example of the C⏚ hardware-optimization workflow.

KEY FEATURES

- FIPS 180-4 SHA-256
- 256-bit digest
- Three variants
- 4× size reduction
- From the standard spec
- Readable C⏚
- Vendor-independent RTL
- Source-included

Architecture

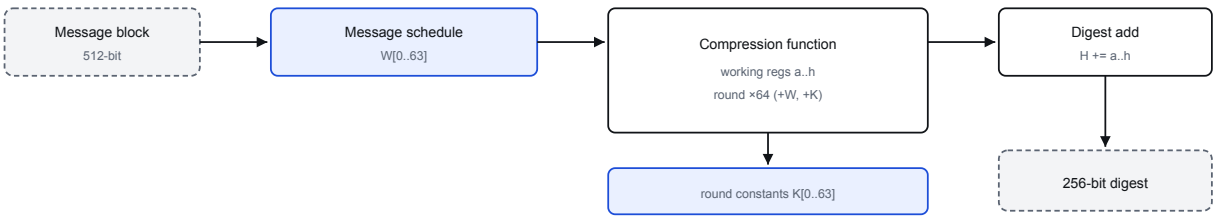

Figure 1. SHA-256 datapath. The message schedule expands each 512-bit block into W[0..63]; the compression function iterates 64 rounds over the working registers using W and the round constants K, then adds the result into the running digest.

1 Overview

The core implements SHA-256 per FIPS 180-4. Each 512-bit message block is expanded by the message schedule into the 64 words W[0..63]; the compression function then performs 64 rounds over the eight working registers (a..h) using W and the round constants K, and adds the result into the running 256-bit hash value.

Three variants trade area for simplicity, illustrating the optimization workflow.

2 Features

- **Algorithm.** SHA-256 (FIPS 180-4); 256-bit digest.
- **Variants.** Basic, ROM-optimized, and shift-register implementations.
- **Optimization.** ≈4× area reduction from basic to optimized, with fewer muxes and simpler logic.
- **Provenance.** Written directly from the standard specification.
- **Pedagogy.** A worked example of the C⏚ size/speed optimization workflow.
- **Portability.** Vendor-independent RTL generated from C⏚.

3 Functional description

Refer to Figure 1.

3.1 Message schedule

Each 512-bit block is expanded into the 64 schedule words W[0..63] used by the compression rounds.

3.2 Compression function

Sixty-four rounds update the working registers a..h using each W word and the corresponding round constant K, per the standard.

3.3 Digest

After the rounds, the working registers are added into the running hash value to produce the 256-bit digest. Variants differ in how the schedule and constants are stored and routed.

> **Note.** The three variants produce identical digests; they exist to demonstrate area/speed trade-offs in C⏚.

4 Interfaces & signals

The core consumes message blocks and produces a digest.

GROUP	DIRECTION	DESCRIPTION
Message in	in	Message block stream, with handshake
Digest out	out	256-bit hash value
Control	in	Start / init, variant selection (build-time)
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 Standards compliance

ITEM	DETAIL
Algorithm	SHA-256 — FIPS 180-4
Digest	256-bit
Block size	512-bit
Variants	basic / ROM-optimized / shift-register

6 Performance

PARAMETER	VALUE	NOTES
Digest	256-bit	
Throughput	Available on request	variant-dependent
Area	≈4× reduction (opt vs basic)	relative
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

- Validated against the FIPS-180-4 known-answer test vectors.
- All three variants produce identical digests.

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-SEC-SHA256
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).