ARC HS56/HS57D/HS58 Processors

Highlights

- Combination dual-issue, 32-bit RISC + DSP processor
- Delivers up to 5700 DMIPS and 9880 CoreMark per core at 1.9 GHz on 16ff (worst case conditions, single-core configuration)
- ARCv2DSP ISA adds over 150 DSP instructions
- Easy DSP programming support with Metaware C/C++ Compiler
- Feature-rich DSP software library for easy algorithm programming
- One 32x32, quad 16x16, or dual 32x16 MACs/cycle
- Fixed point, vector/SIMD DSP processing support
- Based on advanced ARCv2 ISA
- Support for custom instructions
- Support for up to 16MB of close coupled memory and direct mapping of peripherals
- Dual- and quad-core versions for higher performance

Target Applications

- Wireless baseband
- Mid-range home audio
- Voice/speech processing
- Home automation
- ADAS automotive systems
- Home networking

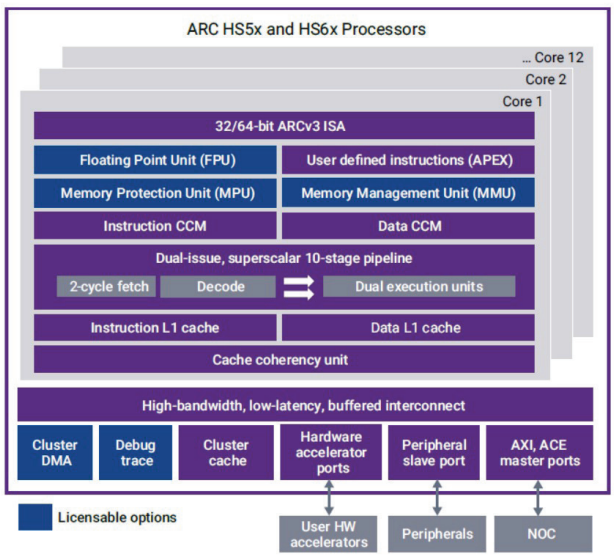
32-Bit Processors for High-Performance Embedded Applications

The DesignWare® ARC® HS45D and HS47D processor feature a dual-issue, 32-bit, RISC + DSP architecture for use in embedded applications where high-performance and high clock speed plus signal processing are required. The cores can be clocked at up to 1.9 GHz in 16ff processes (worst case, single core, base configuration) and offer outstanding performance delivering 3.0 DMIPS/MHz and 5.2 CoreMark/MHz with a small area footprint and low power consumption.

The processors are based on the advanced ARCv2DSP instruction set architecture (ISA) and pipeline, which provides leadership performance-efficiency and code density, and more than 150 DSP instructions. For applications requiring higher performance, dual- and quad-core versions of the HS45D and HS47D cores are available.

The ARC HS45D features close coupled memory (CCM) and is optimized for use in applications where real-time, deterministic behavior is required. The HS47D is designed for high-performance embedded applications that require cache and includes all of the features of the HS45D plus support for up to 64K Level 1 (L1) instruction and data cache.

The processors are designed to be used in applications such as wireless baseband, voice/speech processing, home audio, automotive systems, and other high-end embedded applications that require signal processing.



ARCv2DSP ISA

The DesignWare ARC HS45D and HS47D processor cores implement the DSP enhanced ARCv2DSP 16-/32-bit RISC ISA, which supports advanced processor capabilities and incorporates 150+ DSP instructions that improve DSP performance (Table 1). The architecture and pipeline are designed to meet the needs of next-generation system-on-chip (SoC) applications and enable the development of a full range of 32-bit processor cores, from low-end, extremely power-efficient embedded DSP cores to high-performance solutions that are binary compatible and designed with common pipeline elements. The ARCv2DSP ISA enables the implementation of complex heterogeneous SoCs with processor cores precisely targeted to meet the specific performance and power requirements for each instance on an SoC, while offering the same software programmer's model to simplify program development and task partitioning.

Instruction Set Summary

The ARCv2DSP ISA supports the following classes of DSP instructions/operations: basic saturating arithmetic, vector unpacking/packing, accumulator operations, vector 16x16 MAC, quad 16x16 MAC, dual 16x8 MAC, dual 32x16 MAC, 32x32 MAC and complex (16+16)x(16+16) MAC.

Operation	Mnemonics	Description
Arithmetic		
Absolute value	ABSS, ABSSH, VABS2H, VABSS2H	(Non-) saturating absolute value
Add	ADDS, ADCS, VADD2H, VADDS2H, VADD4B	(Non-)saturating add
Add-subtract	VADDSUB2H, VADDSUBS2H	(Non-) saturating add-subtract
Min/max	VMIN2H, VMAX2H	Minimum/maximum
Negate	NEGS, NEGSH, VNEG2H, VNEGS2H	(Non-) saturating negative
Normalize	VNORM2H	Vector normalize
Subtract	SUBS, SBCS, VSUB2H, VSUBS2H, VSUB4B	(Non-) saturating subtract
Subtract-add	VSUBADD2H, VSUBADDS2H	(Non-) saturating subtract-add
MUL/MAC operations		
16X16 integer	MPYW, MPYW_S, MPYUW, MPYUW_S	Signed/unsigned 16x16 MUL
32x16 fractional	MPYWHFM, MPYWHFMR, MACWHFM, MACWHFMR, MSUBWHFM, MSUBWHFMR, MPYWHFL, MPYWHFLR, MACWHFL, MACWHFLR, MSUBWHFL, MSUBWHFLR	32x16 fractional MUL/MAC/MSUB, saturating, (non-) rounding
32x16 integer	MPYWHL, MACWHL, MPYWHUL, MACWHUL, MPYWHKL, MACWHKL, MPYWHKUL, MACWHKUL	Signed/unsigned 32x16 MUL/MAC
32x32 fractional	MPYF, MPYFR, MACF, MACFR, MSUBF, MSUBFR, MPYDF, MACDF, MSUBDF	32x32 fractional MUL/MAC/MSUB, saturating, (non-) rounding
32x32 integer	MPY, MPY_S, MPYU, MAC, MACU, MPYM, MPYMU, MPYD, MPYDU, MACD, MACDU	Signed/unsigned 32x32 MUL/MAC
Dual 16x8	DMPYHBL, DMPYHBM, DMACHBL, DMACHBM	Dual 16x8 signed MUL/MAC, inner-product style
Vector 16x16 fractional	VMPY2HF, VMPY2HFR, VMAC2HF, VMAC2HFR, VMPY2HWF, VMAC2HNFR, VMSUB2HF, VMSUB2HFR, VMSUB2HNFR	Vector/SIMD 16x16 fractional MUL/MAC/MSUB, saturating, (non-) rounding
Vector 16x16 integer	VMPY2H, VMPY2HU, VMAC2H, VMAC2HU	Vector/SIMD 16x16 integer MUL/MAC, signed/unsigned
Dual 16x16 fractional	DMPYHF, DMPYHFR, DMACHF, DMACHFR, DMPYHWF	Dual 16x16 fractional MUL/MAC, inner-product style, saturating, (non-) rounding
Dual 16x16 integer	DMPYH, DMPYHU, DMACH, DMACHU	Dual 16x16 integer MUL/MAC, inner-product style, signed/unsigned

Continued on next page

Operation	Mnemonics	Description
Complex (16+16)x(16+16)	CMPYHFMR, CMPYHNFR, CMPYHFR, CMPYCHNFR, CMPYCHFR, CMACHNFR, CMACHFR, CMACCHNFR, CMACCHFR	(16+16)x(16+16) complex MUL/MAC, rounding
FFT butterfly (16+16) x(16+16)	CBFLYHF0R, CBFLYHF1R	First half and second half of 2-cycle FFT butterfly
Saturation and rounding		
Saturate	SATH, SATF	Saturate 32b to 16b/32b
Round	RNDH	Round and saturate 32b to 16b
Shifting		
Saturating shifts	ASLS, ASRS, VASLS2H, VASRS2H	Saturating left/right shift
Shifts with rounding	ASRSR, VASRSR2H	Right shift with rounding
Non-saturating shifts	VASL2H, VASR2H, VLSR2H	Non-saturating left/right shift
Vector unpacking/packing		
16b to 32b packing	VALGN2H, VPACK2HL, VPACK2HM	Pack two 16b into 2x16 vector
Replicate	VREP2HL, VREP2HM	Replicate 16b in 2x16 vector
8b to 16b unpacking	VEXT2BHL, VEXT2BHM, VSEXT2BHL, VSEXT2BHM, VEXT2BHLF, VEXT2BHMF	Expand two bytes to 2x16 vector
16b to 8b packing	VPACK2HBL, VPACK2HBM, VPACK2HBLF, VPACK2HBMF	Pack two 16b values into two upper/lower bytes
Permute	VPERM	Permute bytes and sign extend
Divide and square root		
Divide	DIV, DIVU, DIVF	Integer/fractional divide
Remainder	REM, REMU	Signed/unsigned integer remainder
Square root	SQRT, SQRTF	Integer/fractional square root
Accumulator operations		
Get/set	GETACC, SETACC	Get/set accumulator value
Flag	FLAGACC	Copy accumulator status flags
Left-shift	ASLACC, ASLSACC	(non-)saturating shift left
Normalize	NORMACC	Normalize accumulator
Auxiliary operations		
XY address pointer update	MODIF	Apply modifier
64-bit source operations		
Quad 16x16 fractional	QMPYHF, QMACHF	Quad 16x16 fractional MUL/MAC, inner-product style, saturating, non-rounding
Quad 16x16 integer	QMPYH, QMPYHU, QMACH, QMACHU	Quad 16x16 integer MUL/MAC, inner-product style, signed/unsigned

Operation	Mnemonics	Description
Dual 32x16 fractional	DMPYWHF, DMACWHF	Dual 32x16 fractional MUL/MAC, inner-product style, saturating, non-rounding
Dual 32x16 integer	DMPYWH, DMPYWHU, DMACWH, DMACWHU	Dual 32x16 integer MUL/MAC, inner-product style, signed/unsigned
Quad 16b add/subtract	VADD4H, VADDS4H, VSUB4H, VSUBS4H, VADDSUB4H, VADDSUBS4H, VSUBADD4H, VSUBADDS4H	Quad 16b add/subtract /add-subtract/ subtract-add (non-) saturating
Dual 32b add/subtract	VADD2, VADDS2, VSUB2, VSUBS2, VADDSUB, VADDSUBS, VSUBADD, VSUBADDS	Dual 32b add/subtract/add-subtract/ subtract-add (non-) saturating

Table 1: ARCv2DSP ISA DSP instructions

HS45D and HS47D Features

- High-speed, dual-issue, 10-stage pipeline
- Up to 16 MB instruction and data close coupled memory (CCM)
- 64-bit loads and stores
- 4 KB to 64 KB instruction and data cache (HS47D)
- Register file supports 3 write ports and 4 read ports, and up to 8 contexts
- One 32x32, quad 16x16 or dual 32x16 MACs/clock
- Radix-4 hardware divider
- Up to 240 interrupts, with up to 16 configurable preemption levels
- Fast context switch with up to 8 register files
- Native ARM® AMBA® AXI™ or AHB-Lite™ interfaces
- JTAG and Compact JTAG (cJTAG) debug interface

Dual-Issue Pipeline

The high-performance, dual-issue, 10-stage pipeline is optimally balanced to achieve very high embedded performance with very low power consumption. The pipeline is designed to issue up to two instructions per clock (in order) and features two-cycle access to memory allowing the processor to achieve higher clock speeds and making it less sensitive to memory size. The pipeline supports out-of-order retirement, sophisticated branch prediction (with early correction of mispredicted branches) and a late-stage ALU that improves instruction throughput. The processor is supported with a 32-bit hardware multiply (and multiply-accumulate), vector addition and subtraction, and a radix-4 hardware divider. A number of separately licensed options are available including: an IEEE 754-compliant floating point unit, real-time trace debug, and a cluster DMA.

Configurable Options

The processors support a broad range of configurable options, enabling optimization for a specific application's performance, power and size requirements. The included DesignWare ARChitect configuration tool features a graphical interface and produces verified RTL and synthesis scripts that are compatible with industry-standard design flows.

L1 Cache Memory

The ARC HS47D features separate instruction and data L1 cache that can be independently configured for 4 K, 8 K, 16 K, 32 K or 64 K size. The instruction and data cache is fixed to 2-way associativity, and a user-selectable line size of 32, 64 or 128 bytes. The caches can be individually configured to support line locking and invalidate, and to offer debug visibility. For dual- and quad-core configurations the L1 data cache implements the MOESI protocol and supports cache to cache transfers.

Close Coupled Memory

The ARC HS45D and HS47D support 512 B to 16 MB of close coupled memory (CCM) for both instruction and data. The CCM is implemented as separate memory spaces for the Instruction Close Coupled Memory (ICCM) and Data Close Coupled Memory (DCCM). Both ICCM and DCCM have optional support for error-correcting code (ECC) to increase application reliability.

Register File and Program Counter

The HS45D and HS47D register file is configurable, with 16 or 32 32-bit registers. The register file can be constructed from flip-flops, and supports three write ports and four read ports. Some of the registers have defined special purposes like stack pointers (r28), link registers (r29,- r31) and loop counters. The register file can be expanded to up to a total of 60 registers using the extension registers (r32–r59). The ARCv2 16-bit instructions can directly access registers r0–r15 and can indirectly access the remaining registers. The program counter is read only and located at r63 in the register file. The program counter is a 32-bit word aligned register for use as a source operand in all instructions supporting PC-relative addressing.

Bus Interfaces

The processors have native support for the ARM AMBA AXI or AHB-Lite bus protocols. This is a build-time option with the ARM AMBA AXI interface as the default selection. The AXI bus is user configurable with 32-, 64- or 128-bit widths to improve system throughput.

Interrupts and Exceptions

The processors support up to 240 interrupts and exceptions with 16 nested levels of priority. Designers can select any number of interrupts up to 240 at build time. The interrupts are serviced through a jump table that allows higher flexibility in the location and implementation of the interrupt vectors. Interrupts can be triggered by hardware or software.

64-Bit Load and Store

The processors feature 64-bit load double and store double instructions that can be optionally included in the processor at build time. These are single instructions that load or store 64-bits of data to and from register pairs. There is no additional cycle penalty due to the wider and banked DCCM that support non-aligned loads and stores.

Fast Context Switch

For applications that require a deterministic and short latency context switch, the HS45D and HS47D processors can be configured by the user with up to eight register files. This is a build-time option and allows a fast context switch without the need to save processor registers to the stack.

ARC Processor EXtension (APEX) Interface

The processors are designed to be extendable with the addition of custom instructions. These instruction extensions may include more processor and auxiliary registers, new instructions, and additional condition code tests. Custom instructions enable designers to efficiently add their proprietary hardware to the processor to further increase application performance. The processors support 64-bit APEX instructions.

Low-Latency Auxiliary Port

The low-latency auxiliary port offers fast access to on-chip peripherals and memory. The port supports single-cycle read and write register access bringing these peripherals close to the processor and significantly reducing the latency that accompanies accessing peripherals and memory over a multi-layer AMBA bus interface.

In addition, system performance is improved because this processor-to- peripheral bus traffic is moved to the auxiliary bus. The auxiliary port has a 32-bit address space and can support all of the peripheral registers on an SoC.

Multicore Versions

The HS45D and HS47D processors are available in dual-core and quad-core versions if higher performance is required for an application. The multicore versions include multiple instantiations of the processor with hardware for intercore message passing, interrupt handling, semaphores and debug. A 64-bit global real-time counter and a global interrupt distribution unit are included to aid in synchronizing multiple threads across the processors. In the multicore versions the cores can be simultaneously and selectively run, halted or reset in any combination by a debugger or a host system. The HS47D multicore versions have support for L1 cache coherency, and an optional L2 cache that is tightly connected to the cores.

Optional Features (Separately Licensed)

- RTT provides real-time trace debugging features for the ARC HS processor
- The FPU supports single- and double-precision (or both) IEEE 754-compliant math operations
- Cluster DMA programmable memory access controller
- L2 cache (HS47D only)
- Full MMU with 40-bit physical address (HS47D only)

Complete Suite of Development Tools

To facilitate rapid development, the processors are supported by a complete suite of development tools. This includes the MetaWare Development Toolkit that generates performance optimized code that takes advantage of the dual-issue pipeline that is highly efficient and ideal for deeply embedded applications, the ARC xCAM and nSIM simulators and the ARChitect configuration tool.

MIPS offers a suite of GNU tools (ARC GNU) for developers targeting the Linux operating system as well as bare metal systems. The ARC GNU Toolchain includes the GCC compiler and GDB debugger as well a number of utilities and libraries that make up a complete software toolchain.

Compile	MetaWare Compiler	• Optimize your code for size and performance • Leverage core-specific features to reduce cost and increase performance • Utilize your user defined instructions to achieve design goals
	GNU GCC Compiler	• Freely access an open source solution with the GCC compiler
Debug	MetaWare Debugger	• Easily debug multiple targets with the same user interface • Quickly profile hotspots in your code • Use scripting to increase productivity
	JTAG Debuggers	• Efficiently bring-up hardware with tools from Ashling, Green Hills and Lauterbach
	GNU GDB Debugger	• Use the open source GDB debugger to debug real and simulated targets
Deploy	nSIM Simulator	• Develop & debug software before hardware is available • Simulate large programs with very fast ARC models • Quickly optimize your software with near cycle-accurate simulation
	xCAM Simulator	• Use a 100% cycle-accurate model of your core configuration for hardware verification and optimization of critical software
	MQX Real Time OS	• Use an RTOS optimized by MIPS for ARC processors • Very small kernel size as low as 3.5 Kb • Delivered with full source code for easy reference
	Linux, SMP Linux	• Linux for ARC Processors provides all of the benefits of open source software, including complete source code and a large installed base
	ARC Development Platforms	• Develop and debug on real hardware using the MetaWare Development Toolkit or the GNU Tool Chain

Table 2: DesignWare ARC HS45D and HS47D software and development tools

Documentation

The following documentation is available for the DesignWare ARC HS4xD processors:

- ARCv2 Programmers Reference
- ARC HS4x Databook
- ARC HS4x Integration Guide
- ARC APEX Databook

Testing, Compliance and Quality

Verification of the DesignWare ARC processors follows a bottom-up verification methodology from block level through system level. Each functional block within the product follows a functional, coverage-driven test plan.

The plan includes testing for ARCv2 ISA compliance as well as state- and control- specific coverage points that have been exercised using constrained pseudo-random environments and a random instruction sequence generator.

Deliverables

The DesignWare ARC HS45D and HS47D processors are delivered as Verilog HDL in the ARChitect IP Library. The HDL is configured and output from the ARChitect IP Configurator tool. To test that the product performs as expected, a basic testbench of Customer Confidence Tests (CCT) is included.

About MIPS:

MIPS by GlobalFoundries delivers software to silicon with RISC-V for building physical AI platforms. MIPS delivers software-hardware co-design, optimized AI, and custom ASSP design and manufacturing. Together with ARC, MIPS delivers the open, standards-based processor IP portfolio for embedded applications. Physical AI is built on MIPS.

For more information, visit **www.mips.com/arc.**