Linux Arc Documentation

The kernel development community

Jan 15, 2023

CONTENTS

CHAPTER

ONE

LINUX KERNEL FOR ARC PROCESSORS

1.1 Other sources of information

Below are some resources where more information can be found on ARC processors and relevant open source projects.

- https://embarc.org - Community portal for open source on ARC. Good place to start to find relevant FOSS projects, toolchain releases, news items and more.
- https://github.com/foss-for-synopsys-dwc-arc-processors - Home for all development activities regarding open source projects for ARC processors. Some of the projects are forks of various upstream projects, where "work in progress" is hosted prior to submission to upstream projects. Other projects are developed by Synopsys and made available to community as open source for use on ARC Processors.
- Official Synopsys ARC Processors website - location, with access to some IP documentation (Programmer's Reference Manual, AKA PRM for ARC HS processors) and free versions of some commercial tools (Free nSIM and MetaWare Light Edition). Please note though, registration is required to access both the documentation and the tools.

1.2 Important note on ARC processors configurability

ARC processors are highly configurable and several configurable options are supported in Linux. Some options are transparent to software (i.e cache geometries, some can be detected at runtime and configured and used accordingly, while some need to be explicitly selected or configured in the kernel's configuration utility (AKA "make menuconfig").

However not all configurable options are supported when an ARC processor is to run Linux. SoC design teams should refer to "Appendix E: Configuration for ARC Linux" in the ARC HS Databook for configurability guidelines.

Following these guidelines and selecting valid configuration options up front is critical to help prevent any unwanted issues during SoC bringup and software development in general.

1.3 Building the Linux kernel for ARC processors

The process of kernel building for ARC processors is the same as for any other architecture and could be done in 2 ways:

- Cross-compilation: process of compiling for ARC targets on a development host with a different processor architecture (generally x86_64/amd64).
- Native compilation: process of compiling for ARC on a ARC platform (hardware board or a simulator like QEMU) with complete development environment (GNU toolchain, dtc, make etc) installed on the platform.

In both cases, up-to-date GNU toolchain for ARC for the host is needed. Synopsys offers prebuilt toolchain releases which can be used for this purpose, available from:

- Synopsys GNU toolchain releases: https://github.com/foss-for-synopsys-dwc-arc-processors/toolchain/releases
- Linux kernel compilers collection: https://mirrors.edge.kernel.org/pub/tools/crosstool
- Bootlin's toolchain collection: https://toolchains.bootlin.com

Once the toolchain is installed in the system, make sure its "bin" folder is added in your `PATH` environment variable. Then set `ARCH=arc` & `CROSS_COMPILE=arc-linux` (or whatever matches installed ARC toolchain prefix) and then as usual `make defconfig && make`.

This will produce "vmlinux" file in the root of the kernel source tree usable for loading on the target system via JTAG. If you need to get an image usable with U-Boot bootloader, type `make uImage` and `uImage` will be produced in `arch/arc/boot` folder.

CHAPTER

TWO

FEATURE STATUS ON ARC ARCHITECTURE

Subsystem	Feature	Kconfig	Status	Desc
core	cBPF-JIT	HAVE_CBPF_JIT	TODO	arch
core	eBPF-JIT	HAVE_EBPF_JIT	TODO	arch
core	generic-idle-thread	GENERIC_SMP_IDLE_THREAD	ok	arch
core	jump-labels	HAVE_ARCH_JUMP_LABEL	ok	arch
core	thread-info-in-task	THREAD_INFO_IN_TASK	TODO	arch
core	tracehook	HAVE_ARCH_TRACEHOOK	ok	arch
debug	debug-vm-pgtable	ARCH_HAS_DEBUG_VM_PGTABLE	ok	arch
debug	gcov-profile-all	ARCH_HAS_GCOV_PROFILE_ALL	TODO	arch
debug	KASAN	HAVE_ARCH_KASAN	TODO	arch
debug	kcov	ARCH_HAS_KCOV	TODO	arch
debug	kgdb	HAVE_ARCH_KGDB	ok	arch
debug	kmemleak	HAVE_DEBUG_KMEMLEAK	ok	arch
debug	kprobes	HAVE_KPROBES	ok	arch
debug	kprobes-on-ftrace	HAVE_KPROBES_ON_FTRACE	TODO	arch
debug	kretprobes	HAVE_KRETPROBES	ok	arch
debug	optprobes	HAVE_OPTPROBES	TODO	arch
debug	stackprotector	HAVE_STACKPROTECTOR	TODO	arch
debug	uprobes	ARCH_SUPPORTS_UPROBES	TODO	arch
debug	user-ret-profiler	HAVE_USER_RETURN_NOTIFIER	TODO	arch
io	dma-contiguous	HAVE_DMA_CONTIGUOUS	TODO	arch
locking	cmpxchg-local	HAVE_CMPXCHG_LOCAL	TODO	arch
locking	lockdep	LOCKDEP_SUPPORT	ok	arch
locking	queued-rwlocks	ARCH_USE_QUEUED_RWLOCKS	TODO	arch
locking	queued-spinlocks	ARCH_USE_QUEUED_SPINLOCKS	TODO	arch
perf	kprobes-event	HAVE_REGS_AND_STACK_ACCESS_API	ok	arch
perf	perf-regs	HAVE_PERF_REGS	TODO	arch
perf	perf-stackdump	HAVE_PERF_USER_STACK_DUMP	TODO	arch
sched	membarrier-sync-core	ARCH_HAS_MEMBARRIER_SYNC_CORE	TODO	arch
sched	numa-balancing	ARCH_SUPPORTS_NUMA_BALANCING	—	arch
seccomp	seccomp-filter	HAVE_ARCH_SECCOMP_FILTER	TODO	arch
time	arch-tick-broadcast	ARCH_HAS_TICK_BROADCAST	TODO	arch
time	clockevents	!LEGACY_TIMER_TICK	ok	arch
time	context-tracking	HAVE_CONTEXT_TRACKING	TODO	arch
time	irq-time-acct	HAVE_IRQ_TIME_ACCOUNTING	TODO	arch
time	virt-cpuacct	HAVE_VIRT_CPU_ACCOUNTING	TODO	arch

Table 1 – continued from p

Subsystem	Feature	Kconfig	Status	Desc
vm	batch-unmap-tlb-flush	ARCH_WANT_BATCHED_UNMAP_TLB_FLUSH	TODO	arch
vm	ELF-ASLR	ARCH_HAS_ELF_RANDOMIZE	TODO	arch
vm	huge-vmap	HAVE_ARCH_HUGE_VMAP	TODO	arch
vm	ioremap_prot	HAVE_IOREMAP_PROT	ok	arch
vm	PG_uncached	ARCH_USES_PG_UNCACHED	TODO	arch
vm	pte_special	ARCH_HAS_PTE_SPECIAL	ok	arch
vm	THP	HAVE_ARCH_TRANSPARENT_HUGEPAGE	ok	arch