Build system for gcc toolchain Introduction

Shiva Chen

Checkout source by build system

- **Marvell internal mgcc build-system git repo**
  - **git clone** git://10.19.133.151/mgcc/build-system.git
- **git branch –a**

```
remotes/origin/MGCC_4.8.4_RELEASE_2015.01
remotes/origin/MGCC_4.8_MASTER_2012.12
remotes/origin/MGCC_4.9_MASTER_2014.04
remotes/origin/master
```

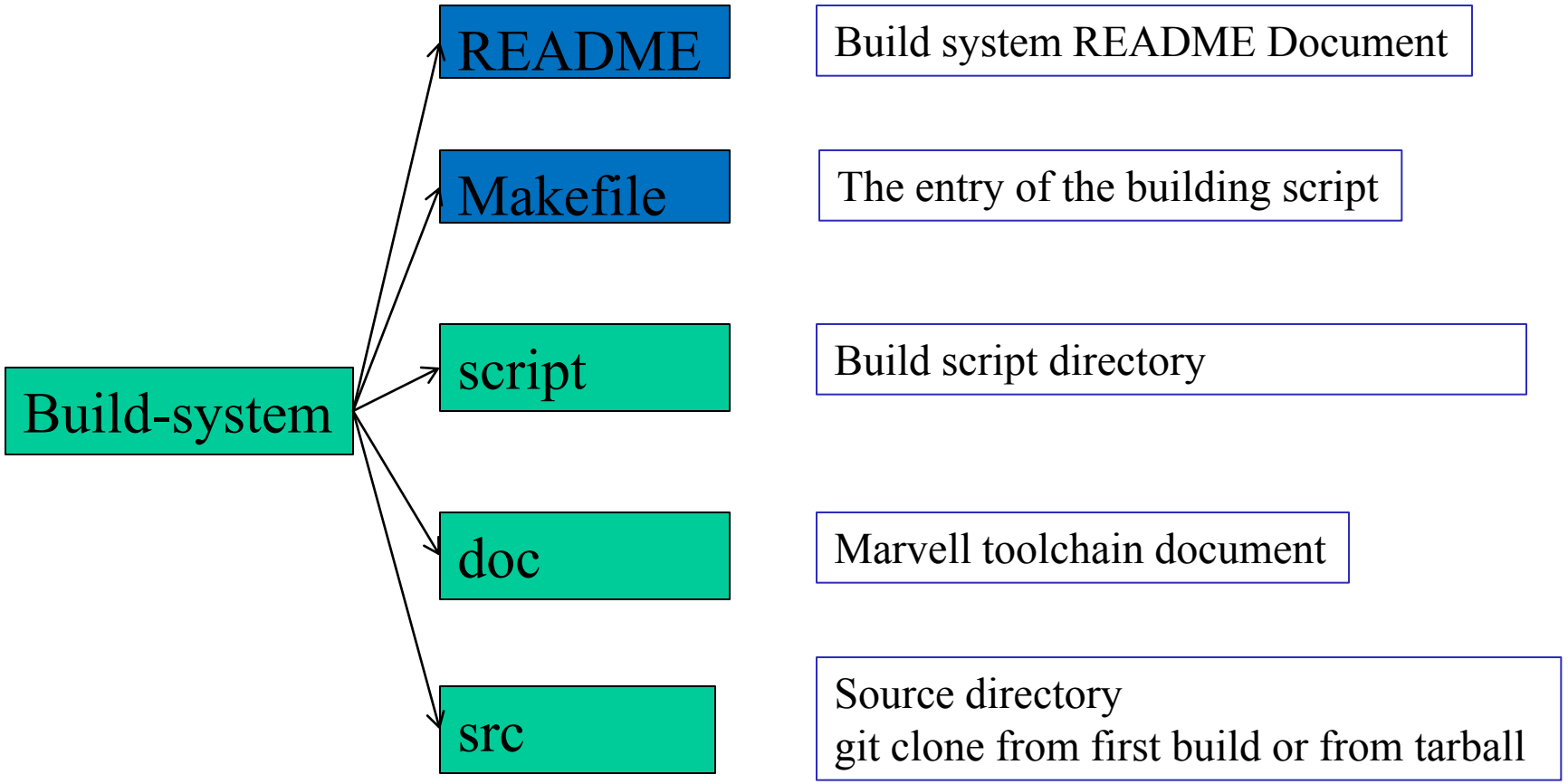
- **Each branch is use to build specify toolchain**
  - **E.g. branch “MGCC_4.9_MASTER_2014.04” use to build MGCC 4.9 master toolchains**

Checkout source by build system

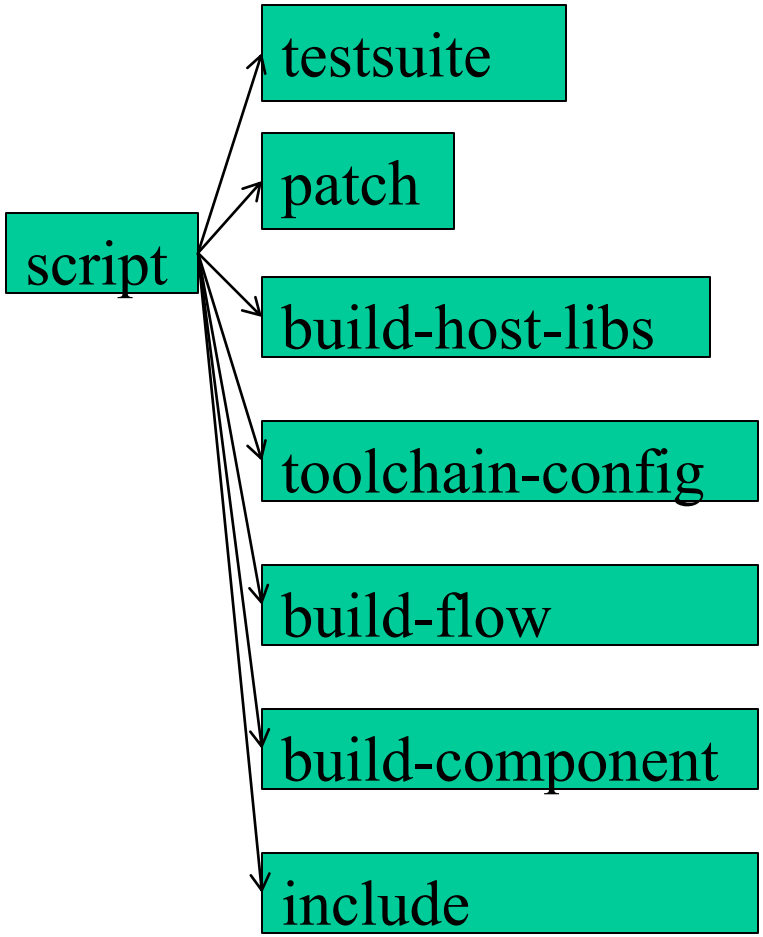
- **make {toolchain-target}**
  - **will automatically git clone source before build the toolchain**
  - **Important:**
    - git checkout the build-system branch you want before build any toolchain

- **For the user get build-system by tarball**
  - **E.g. Marvell_5.1.1_toolchain_20150622-src.tar.xz**
    - tar xf *.tar.xz
    - cd Marvell_5.1.1_toolchain_20150622-src
    - Then you could start to build toolchains
      - E.g . make armv7-marvell-eabihf-hard

Build-system folder structure



Build-system folder structure



Toolchain testsuite script directory

All patch need for the component (linux/..)

Build host-libs (mpc/mpfr/..)
needs to build gcc

List the toolchain build-system
Support to build

Build flow the toolchain
Ex: 1. binutils 2. gcc-first …

How to config/build each component
(binutils/gcc/..)

Invoke testsuite script
Tarball commands implementation
Detect host setting (x86/i386) (CPU numb[illegible])

Help commands

- **make**
  - **Show all help commands**

```
Marvell mgcc-4.8.4 toolchains building system
The following commands show more details:

  make help-target                list all toolchain targets
  make help-native-target         list all native toolchain targets
  make help-build                 show building commands
  make help-test                  show testing commands
  make help-misc                  show misc commands
  make help-all                   show all commands
```

Help commands

- **make help-target**
  - **Show all toolchains could build**

```
Build the mgcc crossing-toolchain:

  make <TARGET_NAME>

The valid <TARGET_NAME> is:

  arm-marvell-eabi_all
        armv5-marvell-eabihf-hard
        armv5-marvell-eabi-soft
        armv5-marvell-eabi-softfp
        armv7-marvell-eabihf-hard
        armv7-marvell-eabi-soft
        armv7-marvell-eabi-softfp
  armeb-marvell-eabi_all
        armebv5-marvell-eabihf-hard
        armebv5-marvell-eabi-soft
        armebv5-marvell-eabi-softfp
        armebv7-marvell-eabihf-hard
        armebv7-marvell-eabi-soft
        armebv7-marvell-eabi-softfp
  arm-marvell-eabi-mingw32_all
        armv7-marvell-eabihf-hard-mingw32
```


MARVELL®

Help commands

- **make help-build**
  - **Show all build options**

```
Build the mgcc toolchain:

  make <TARGET_NAME> [<OPTIONS> ...]
                                build <TARGET_NAME> toolchain with <OPTIONS>
  make clean                    remove all build-working directories
  make clean TOOLCHAIN=<TARGET_NAME>
                                remove the <TARGET_NAME> build-working directory

The valid <OPTIONS> is:

  VERBOSE=1                     show more building flow information
  MDEBUG=1                      build with debug mode for developers
  MAXPAGESIZE=[64K|32K]         linker's max page size (default: 64K)
  MPDF=1                        generate toolchain PDF documents
  NCPU=-j<N>                    make parallel with <N> tasks (default: host's co
re number)
  ENABLE_FORTRAN=1              build FORTRAN compiler
  STAGE=<STRING>                add <STRING> to GCC version (default: release)
  EXTRA=1                       build compiler with extra-lib
  TARGET_TUNE=<TUNE>            build compiler with specify target tune
  TARGET_FPU=<FPU>              build compiler with specify target fpu
```

Help commands

- **make help-test**
  - **Show all commands to run toolchain testsuite**

```
Run the package's testsuite:

  make check-[gcc|binutils|gdb|all] [<OPTIONS>]
                                test the packages and put the log and summary
in the 'testsuite/result/' path
  make test-makefile            create Makefiles in the 'testsuite/result/' pa
th
  make test-report              produce testing reports

The following <OPTIONS> are for check-gcc:

  TOOLCHAIN=<TARGET_NAME>       test the <TARGET_NAME> toolchain (default: all
 built ones)
  TOOLCHAIN_LIST="<TARGET_NAME1> <TARGET_NAME2> ..."
                                test the specified toolchain list
  THUMB=1                       test THUMB code (gcc -mthumb)
  ARCH=armv7-r                  test ARMv7-R code (gcc -march=armv7-r)
  REMOTE=<ARM_MACHINE>          test executables on the remote board (default:
 QEMU)

The QEMU_BIN_PATH environment variable tells building script
where the QEMU user mode is. If not setting, use the one
```

Help commands

- **make help-misc**
  - **Show other commands**

```
The following commands are for developers:

  make source                    git clone source before building toolchains
  make update                    git pull building script and package source
  make release                   build all toolchains of the release
  make tarball                   tar all toolchains for distributing
  make tarball TAR_SRC=1         tar all toolchains and package source
  make forall CMD=<COMMAND>      change to each package's source directory a
nd do the <COMMAND>
```


MARVELL®

Start to build toolchain

- **make ${toolchain}**
  - **E.g. make armv7-marvell-eabihf-hard**
  - **Start building proccess**
  - **Will download package from WEB for first time building toolchain**
    - Including Marvell tuning toolchain source (gcc-src/binutils-src/…)
      - Only will download success if you have Marvell toolchains source access right
      - Or else, you could get Marvell toolchain source tarball by Marvell compiler team
        - » E.g. Marvell_4.9.3_toolchain_20150121-src.tar.xz
    - Including mpfr/ncurse/linux… needed by building mgcc toolchains.
      - Download from GNU official web site
    - Including qemu needed by running toolchain testsuite
      - Download from QEMU official web site

```
shivac@lex[12:01 PM]~/tmp/build-system$make armv7-marvell-eabihf-hard
make[1]: Entering directory `/home/shivac/tmp/build-system'
git clone MGCC_TRUNK : binutils-src.
Cloning into 'binutils-src'...
remote: Counting objects: 32826
```

Start to build toolchain

- **Output message while building the toolchain**

```
x86_64-linux-gnu gmp build complete
x86_64-linux-gnu mpfr build complete
x86_64-linux-gnu mpc build complete
x86_64-linux-gnu isl build complete
x86_64-linux-gnu cloog build complete
x86_64-linux-gnu zlib build complete
x86_64-linux-gnu ncurses build complete
x86_64-linux-gnu expat build complete
```

Build host tools needed by building toolchain
Host tools will put on build-system/tools folder

Start to build toolchain

- **Output message while building the toolchain**

```
armv7-marvell-eabi-soft install_binutils build complete
armv7-marvell-eabi-soft build_gcc_first build complete
armv7-marvell-eabi-soft build_newlib build complete
armv7-marvell-eabi-soft build_gcc_final build complete
armv7-marvell-eabi-soft build_extra_lib build complete
armv7-marvell-eabi-soft strip_host_objects build complete
armv7-marvell-eabi-soft strip_eabi_target_objects build complete
armv7-marvell-eabi-soft eabi-finish build complete
armv7-marvell-eabi-soft build complete
```

Toolchains will install in Release/install folder
Build folder will put on Release/build folder

Run gcc testsuite

- **make check-gcc TOOLCHAIN=${target}**
  - **E.g. make check-gcc TOOLCHAIN=armv7-marvell-eabihf-hard**
  - **Run gcc testsuite**

- **First time run testsuite will build flowing component**

```
building tclsh
building expect
building runtest
building qemu-2.2.0
```

- **make check-gcc**
  - **Run gcc testsuite for all toolchains have been build if without specify TOOLCHAIN**
    - Run gcc testsuite for all toolchains in “Release/install” folder

Run gcc testsuite

- **Each toolchain testsuite result will put in test-result/{target}**
  - **E.g. ls test-result/armv7-marvell-linux-gnueabihf-hard-4.8.4_i686/**

```
config.log   g++.sum          libgomp.sum     libstdc++.sum
gcc.log      libatomic.log    libitm.log      Makefile
gcc.sum      libatomic.sum    libitm.sum
g++.log      libgomp.log      libstdc++.log
```

  - ***.sum**
    - testsuite summary file for each testing item
  - ***.log**
    - testsuite compile/run log for each testing item
  - **Makefile**
    - Testing driver
      - Auto generate while first time run testing for the toolchain

Run gcc testsuite

- **Testing Driver for each toolchain**
  - **When will create ?**
    - first time running testsuite for the toolchain
  - **What's inside Testing driver**
    - Testing driver will export several environment valuable which needed for running testsuite
      - TOOLCHAIN_PATH
        - » Toolchain absolute path
      - QEMU_PATH
        - » Qemu absolue path
      - DEJAGNU
        - » Indicate master.exp location
        - » master.exp will specify board configure files location
      - SYSROOT
        - » Toolchain sysroot path
        - » pass to qemu let qemu could find toolchain runtime library correctly
    - Run testsuite commands
      - 1. cd {toolchain_gcc_final_build_folder}
      - 2. make check RUNTESTFLAGS="--target boards {board configure file}"

Run gcc testsuite

- **make test-report**
  - **Produce a simple testing report**
  - **If you want a testsuite report for all toolchains have been run in test-result folder**

```
------------------------------------------------------------
result/armebv7-marvell-linux-gnueabihf-hard-4.8.4_x86_64/libgomp.sum
FAIL: libgomp.graphite/force-parallel-6.c execution test
------------------------------------------------------------
result/armebv7-marvell-linux-gnueabi-softfp-4.8.4_x86_64/libgomp.sum
FAIL: libgomp.graphite/force-parallel-6.c execution test
------------------------------------------------------------
result/armv5-marvell-linux-gnueabi-soft-4.8.4_x86_64/libgomp.sum
FAIL: libgomp.graphite/force-parallel-6.c execution test
------------------------------------------------------------
result/armebv5-marvell-linux-gnueabihf-hard-4.8.4_x86_64/libgomp.sum
FAIL: libgomp.graphite/force-parallel-6.c execution test
------------------------------------------------------------
result/armv7-marvell-linux-gnueabi-soft-4.8.4_x86_64/libgomp.sum
FAIL: libgomp.graphite/force-parallel-6.c execution test
------------------------------------------------------------
result/armv5-marvell-linux-gnueabi-softfp-4.8.4_x86_64/libgomp.sum
FAIL: libgomp.graphite/force-parallel-6.c execution test
------------------------------------------------------------
```

Run gcc testsuite

- **Re-produce a single testcase**
  - **After read the testsuite report, you may find a fail case and try to re-produce it**
    - Use the testing driver “Makefile” in each test-result/{target} folder
    - syntex
      - make check RUNFLAGS={expect file}={testcase} TEST_CFLAGS={extra cflags}

```
shivac@lex[11:30 AM]~/build-system-4.8/testsuite/res
ult/armv7-marvell-linux-gnueabihf-hard-4.8.4_x86_64$
make check RUNFLAGS=conformance.exp=14775.cc
```

```
Running /home/shivac/build-system-4.8/src/gcc-src/li
bstdc++-v3/testsuite/libstdc++-dg/conformance.exp ..
.
PASS: 27_io/fpos/14775.cc (test for excess errors)
PASS: 27_io/fpos/14775.cc execution test

                === libstdc++ Summary ===

# of expected passes            2
```

Result in libstdc++.sum

MARVELL®

Run gcc testsuite

- **Run gcc testcase on remote arm host**
  - **Why need feature?**
    - Qemu user mode not fully support multi-threading.
    - Need a easy way to verify it's a toolchain bug or qemu issue
  - **Need store public key to remote host before you use the feature**
    - Use scp copy binary to remote
    - Use ssh to run binary on remote
    - Would be slower than run on qemu
      - Run on remote will scp/ssh for each testcase
  - **EX:**
    - cd testsuite/result/${target} &&
      - make check RUNFLAGS=conformance.exp=14775.cc REMOTE=armhf@robin

Build commands

- **make ${target}**
  - **Build the target toolchain**
  - **Could use “make help-target” to show target list before building toolchain**
- **make release**
  - **Build all the toolchain for the release at once**
- **make update**
  - **git pull all marvell tuning component source**
    - Including gcc-src/binutils-src/newlib-src/glibc-src/ build-system itself
  - **Don’t use the command if you get source by *-src-tar.xz tarball**

Build commands

- **make clean**
  - **Remove following folders**
    - Release/
      - Toolchain folder
    - Mar*/
      - Tarball folder
    - testsuite/result
      - Testsuite result folder

- **make clean TOOLCHAIN={target}**
  - **Remove specify toolchain install/build folder**

Build options

- **MDEBUG=1**
  - **E.g. make armv7-marvell-eabihf-hard MDEBUG=1**
    - Build the toolchain with –g3 –O0
      - Build toolchain with diable optimization and enable debug info
- **VERBOSE=1**
  - **E.g. make armv7-marvell-eabihf-hard VERBOASE=1**
    - To show building flow
- **MAXPAGESIZE=[64K|32K]**
  - **E.g. make armv7-marvell-eabihf-hard MAXPAGESIZE=32K**
    - Specify toolchain max page size
    - Default max page size is 64K

Build options

- **MPDF=1**
  - **E.g. make armv7-marvell-eabihf-hard MPDF=1**
    - To generate toolchain document
      - gcc.pdf, binutils.pdf,…
    - We would need generate toolchain document for the release toolchain

- **EXTRA=1**
  - **E.g. make armv7-marvell-eabihf-hard EXTRA=1**
    - To build toolchain with Marvell extra library
    - Including
      - qemu hint startup code
        - To insert specific assembly code to start/end record trace on qemu
        - Enable by compile marvell gcc with –specs= movlrlr.specs
      - Assembly library
      - Gcc plugins to dump optimization flags

Build options

- **NCPU=-j<N>**
  - **E.g. make armv7-marvell-eabihf-hard NCPU=-j1**
    - make parallel with <N> tasks
    - Default <N> use host's core number to speedup building
- **TARGET_TUNE=<TUNE>**
  - **E.g. make armv7-marvell-eabihf-hard TARGET_TUNE=marvell-whitney**
    - To build toolchain with specific target tune
- **TARGET_FPU=<FPU>**
  - **E.g. make armv7-marvell-eabihf-hard TARGET_FPU=neon**
    - To build toolchain with specific target fpu
- **ENABLE_FORTRAN=1**
  - **E.g. make armv7-marvell-eabihf-hard ENABLE_FORTRAN=1**
    - To build toolchain with fortran compiler

Test commands

- **make check-binutils TOOLCHAIN={target}**
  - **Run binutils testsuite for the toolchain**
- **make check-binutils**
  - **Run binutils testsuite for all toolchain in Release/install folder if without specify TOOLCHAIN**
- **make check-gdb TOOLCHAIN={target}**
  - **Run gdb testsuite for the toolchain**
  - **Currently, only support mgcc-4.8**
    - Still have fail test case need gdb expert to verify
- **make test-report**
  - **Create simple testsuite report**
    - report-* : all test item report
    - report-fail-*: only report fail test case

Test options

- **Make check-gcc THUMB=1**
  - **E.g. make check-gcc TOOLCHAIN=armv7-marvell-eabihf-hard THUMB=1**
    - Run gcc-testsuite with –mthumb to test thumb code generation
    - Only the toolchain support thumb mode will run when THUMB=1
      - armv7/armv8 support thumb2
      - armv5 only soft toolchain support thumb mode

- **Make check-gcc ARCH=armv7-r**
  - **E.g. make check-gcc TOOLCHAIN=armv7-marvell-eabihf-hard ARCH=armv7-r**
    - Run gcc testsuite with –march=armv7-r to test armv7-r code generation
    - Only the toolchain support armv7-r will run when ARCH=armv7-r
      - armv7 eabi toolchain support armv7-r code generation

MISC commands

- **make tarball**
  - **Tar all toolchains to Marvell_${date} folder**
- **Make tarball TAR_SRC=1**
  - **Tar all toolchains and source to Marvell_${date} folder**
- **make forall CMD="command"**
  - **To operate command for all component source**
  - **E.g. make forall CMD="git tag test-release"**

Status so for

- **Not support**
  - **build mgcc-4.6**
- **Support**
  - **Build gcc-4.8/4.9/5**
- **Testsuite status**
  - **Gcc-4.8 gcc/binutils testsuite run on qemu user mode**
    - Fully testing
    - We already know and list the case would FAIL on qemu user mode but PASS on real board
  - **Gcc-4.9 gcc/binutils testsuite run on qemu user mode**
    - We won't release mgcc4.9, so not fully testing yet
  - **Gcc-5 gcc/binutils testsuite run on qemu user mode**
    - On-going
  - **Gdb testsuite not fully testing yet.**

Build script git repo

- **Contributor :**
  - **git clone ssh://gitosis@10.19.133.151/mgcc/build-system.git**
- **Read only:**
  - **git clone git://10.19.133.151/mgcc/build-system.git**
  - **Could access without gitosis access right**
  - **Could not git push commit back to git repo**

Thank you

MARVELL®