Adaptive Microprocessor with Miniature Accelerator using LLVM Infrastructure and FPGA: The Case of ARM Cortex-M0

(Appendix B)

Ehsan Ali(ehssan.aali@gmail.com), Wanchalerm Pora (wanchalerm.p@chula.ac.th)
Chulalongkorn University of Thailand
Last update: 28th July 2021

Listing 1: FFT Cooley-Tukey Algorithm in C++

```
void fft(CArray& x)
{
  // DFT
  unsigned int N = x.size(), k = N, n;
  double thetaT = 3.14159265358979323846264338328L / N;
  Complex phiT = Complex(cos(thetaT), -sin(thetaT)), T;
  while (k > 1)
  {
    n = k;
    k >>= 1;
    phiT = phiT * phiT;
    T = 1.0L;
    for (unsigned int l = 0; l < k; l++)
    {
      for (unsigned int a = l; a < N; a += n)
      {
        unsigned int b = a + k;
        Complex t = x[a] - x[b];
        x[a] += x[b];
        x[b] = t * T;
      }
      T *= phiT;
    }
  }
  // Decimate
  unsigned int m = (unsigned int)log2(N);
  for (unsigned int a = 0; a < N; a++)
  {
    unsigned int b = a;
    // Reverse bits
    b = (((b & 0xaaaaaaaa) >> 1) | ((b & 0x55555555) << 1));
    b = (((b & 0xcccccccc) >> 2) | ((b & 0x33333333) << 2));
    b = (((b & 0xf0f0f0f0) >> 4) | ((b & 0x0f0f0f0f) << 4));
    b = (((b & 0xff00ff00) >> 8) | ((b & 0x00ff00ff) << 8));
    b = ((b >> 16) | (b << 16)) >> (32 - m);
    if (b > a)
    {
      Complex t = x[a];
      x[a] = x[b];
      x[b] = t;
    }
  }
}
```

1. FFT in C/C++

We write an FFT algorithm (shown in Listing 1) in C++ and compile and link it by issuing:

```
$ clang -lm -lstdc++ fft.cpp -o fft
```

To cross-compile for ARM using LLVM we need:

1. A **libc**. Good choices for that for baremetal are: newlib or musl.
2. **Builtins**. In LLVM, that is provided in the compiler-rt module.
3. For C++ we need 3 things:
   a. **abi library** like LLVM libcxxabi. There are also libsupc++, and libcxxrt.
   b. An **unwinder** like LLVM libunwind.
   c. A C++ standard library like LLVM libcxx.

For compiling LLVM, we can look into the project's configuration options using:

```
$ cmake -LAH | awk '{if(f)print} /-- Cache values/{f=1}'
$ ccmake ../lvm
```

A home brew formula exists at : https://github.com/eblot/homebrew-armeabi
To get a C compiler for Cortex-M0 we first build LLVM itself:

```
$ cmake -G Ninja ../llvm -DCMAKE_BUILD_TYPE=Debug -DLLVM_ENABLE_PROJECTS="clang;clang-tools-extra;lld;lldb" -
DLLVM_ENABLE_SPHINX=False -DLLVM_INCLUDE_TESTS=False -DLLVM_TARGET_ARCH=ARM -
DLLVM_TARGETS_TO_BUILD=ARM -DLLVM_INSTALL_UTILS=ON -DLLVM_DEFAULT_TARGET_TRIPLE=arm-none-eabi -
DCMAKE_CROSSCOMPILING=ON -DLLDB_USE_SYSTEM_DEBUGSERVER=ON -DCMAKE_INSTALL_PREFIX=/home/esi/arm-
none-eabi -DBUILD_SHARED_LIBS=ON -DLLVM_BUILD_DOCS=OFF -DLLVM_ENABLE_BINDINGS=OFF -
DLLVM_ENABLE_DOXYGEN=OFF
```

We then build *newlib* (Some patched need to be applied):

```
$ CC_FOR_TARGET=/home/esi/arm-none-eabi/bin/clang AR_FOR_TARGET=/home/esi/arm-none-eabi/bin/llvm-ar
NM_FOR_TARGET=/home/esi/arm-none-eabi/bin/llvm-nm RANLIB_FOR_TARGET=/home/esi/arm-none-eabi/bin/llvm-ranlib
READELF_FOR_TARGET=/home/esi/arm-none-eabi/bin/llvm-readelf CFLAGS_FOR_TARGET="--target=armv6m-none-eabi -
mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -g -Os -ffunction-sections -fdata-sections -fno-stack-protector -
fvisibility=hidden -Wno-unused-command-line-argument" AS_FOR_TARGET=/home/esi/arm-none-eabi/bin/clang ../configure --
host=x86_64-linux-gnu --build=x86_64-linux-gnu --target=armv6m-none-eabi --prefix=/home/esi/arm-none-eabi --disable-newlib-supplied-
syscalls --disable-newlib-fvwrite-in-streamio --disable-newlib-fseek-optimization --disable-newlib-wide-orient --enable-newlib-nano-malloc
--disable-newlib-unbuf-stream-opt --enable-lite-exit --enable-newlib-global-atexit --disable-newlib-nano-formatted-io --disable-newlib-
fvwrite-in-streamio --enable-newlib-io-c99-formats --enable-newlib-io-float --disable-newlib-io-long-double --disable-nls

$ make \& make -j1 install;
```

After that we build *compiler-rt*:

```
        $ cmake -G Ninja ../compiler-rt -DCMAKE_INSTALL_PREFIX=/home/esi/arm-none-eabi/armv6m-none-eabi -
DCMAKE_TRY_COMPILE_TARGET_TYPE=STATIC_LIBRARY -DCMAKE_SYSTEM_PROCESSOR=arm -
DCMAKE_SYSTEM_NAME=Generic -DCMAKE_CROSSCOMPILING=ON -DCMAKE_CXX_COMPILER_FORCED=TRUE -
DCMAKE_BUILD_TYPE=Debug -DCMAKE_C_COMPILER=/home/esi/arm-none-eabi/bin/clang -
DCMAKE_CXX_COMPILER=/home/esi/arm-none-eabi/bin/clang++ -DCMAKE_LINKER=/home/esi/arm-none-eabi/bin/clang -
DCMAKE_AR=/home/esi/arm-none-eabi/bin/llvm-ar -DCMAKE_RANLIB=/home/esi/arm-none-eabi/bin/llvm-ranlib -
DCMAKE_C_COMPILER_TARGET=armv6m-none-eabi -DCMAKE_ASM_COMPILER_TARGET=armv6m-none-eabi -
DCMAKE_SYSROOT=/home/esi/arm-none-eabi/armv6m-none-eabi/ -DCMAKE_SYSROOT_LINK=/home/esi/arm-none-eabi/armv6m-
none-eabi/ -DCMAKE_C_FLAGS="--target=armv6m-none-eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -
g -Os -ffunction-sections -fdata-sections -fno-stack-protector -fvisibility=hidden -Wno-unused-command-line-argument" -
DCMAKE_ASM_FLAGS="--target=armv6m-none-eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -g -Os -
ffunction-sections -fdata-sections -fno-stack-protector -fvisibility=hidden -Wno-unused-command-line-argument" -
DCMAKE_CXX_FLAGS="--target=armv6m-none-eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -g -Os -
ffunction-sections -fdata-sections -fno-stack-protector -fvisibility=hidden -Wno-unused-command-line-argument" -
DCMAKE_EXE_LINKER_FLAGS=-L/home/esi/arm-none-eabi/lib -DLLVM_CONFIG_PATH=/home/esi/arm-none-eabi/bin/llvm-config
-DLLVM_DEFAULT_TARGET_TRIPLE=armv6m-none-eabi -DLLVM_TARGETS_TO_BUILD=ARM -DLLVM_ENABLE_PIC=OFF -
DCOMPILER_RT_OS_DIR=baremetal -DCOMPILER_RT_BUILD_BUILTINS=ON -DCOMPILER_RT_BUILD_SANITIZERS=OFF -
DCOMPILER_RT_BUILD_XRAY=OFF -DCOMPILER_RT_BUILD_LIBFUZZER=OFF -DCOMPILER_RT_BUILD_PROFILE=OFF -
DCOMPILER_RT_BAREMETAL_BUILD=ON -DCOMPILER_RT_DEFAULT_TARGET_ONLY=ON -
DCOMPILER_RT_INCLUDE_TESTS=OFF -DCOMPILER_RT_USE_LIBCXX=ON -DUNIX=1
```

The above steps provide us ***libclang_rt.builtins-armv6m.a*** and ***libc.a*** in /home/esi/arm-none-eabi/armv6m_none-eabi/lib.

When we compile files for Cortex-M0 we must set /home/esi/arm-none-eabi/armv6m-none-eabi as the *sysroot*.

We now can compile C programs for ARM Cortex-M0 by issuing:

```
        $ /home/esi/arm-none-eabi/bin/clang fft.c --sysroot=/home/esi/arm-none-eabi/armv6m-none-eabi --target=armv6m-none-eabi -
mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -g -Os -ffunction-sections -fdata-sections -fno-stack-protector -
fvisibility=hidden -Wno-unused-command-line-argument -L/home/esi/arm-none-eabi/armv6m-none-eabi/lib -o fft
```

We then compile *libcxx*:

```
        $ cmake -G Ninja ../libcxx -DCMAKE_INSTALL_PREFIX=/home/esi/arm-none-eabi/armv6m-none-eabi -
DCMAKE_TRY_COMPILE_TARGET_TYPE=STATIC_LIBRARY -DCMAKE_SYSTEM_PROCESSOR=arm -
DCMAKE_SYSTEM_NAME=Generic -DCMAKE_CROSSCOMPILING=ON -DCMAKE_CXX_COMPILER_FORCED=TRUE -
DCMAKE_BUILD_TYPE=Debug -DCMAKE_C_COMPILER=/home/esi/arm-none-eabi/bin/clang -
DCMAKE_CXX_COMPILER=/home/esi/arm-none-eabi/bin/clang++ -DCMAKE_LINKER=/home/esi/arm-none-eabi/bin/clang -
DCMAKE_C_COMPILER_AR=/home/esi/arm-none-eabi/bin/llvm-ar -DCMAKE_C_COMPILER_RANLIB=/home/esi/arm-none-
eabi/bin/llvm-ranlib -DCMAKE_CXX_COMPILER_AR=/home/esi/arm-none-eabi/bin/llvm-ar -
DCMAKE_CXX_COMPILER_RANLIB=/home/esi/arm-none-eabi/bin/llvm-ranlib -DCMAKE_C_COMPILER_TARGET=armv6m-none-
eabi -DCMAKE_CXX_COMPILER_TARGET=armv6m-none-eabi -DCMAKE_SYSROOT=/home/esi/arm-none-eabi/armv6m-none-eabi/
-DCMAKE_SYSROOT_LINK=/home/esi/arm-none-eabi/armv6m-none-eabi/ -DCMAKE_C_FLAGS="--target=armv6m-none-eabi -
mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -g -Os -ffunction-sections -fdata-sections -fno-stack-protector -
fvisibility=hidden -fno-use-cxa-atexit -Wno-unused-command-line-argument -D_LIBUNWIND_IS_BAREMETAL=1 -
D_GNU_SOURCE=1 -D_POSIX_TIMERS=1 -D_LIBCPP_HAS_NO_LIBRARY_ALIGNED_ALLOCATION -I/home/esi/arm-none-
eabi/armv6m-none-eabi/include" -DCMAKE_CXX_FLAGS="--target=armv6m-none-eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-
enums -mfloat-abi=soft -g -Os -ffunction-sections -fdata-sections -fno-stack-protector -fvisibility=hidden -fno-use-cxa-atexit -Wno-unused-
command-line-argument -D_LIBUNWIND_IS_BAREMETAL=1 -D_GNU_SOURCE=1 -D_POSIX_TIMERS=1 -
D_LIBCPP_HAS_NO_LIBRARY_ALIGNED_ALLOCATION -I/home/esi/arm-none-eabi/armv6m-none-eabi/include" -
DCMAKE_EXE_LINKER_FLAGS="-L/home/esi/arm-none-eabi/armv6m-none-eabi/lib/" -DLLVM_CONFIG_PATH=/home/esi/arm-
none-eabi/bin/llvm-config -DLLVM_TARGETS_TO_BUILD=ARM -DLLVM_ENABLE_PIC=OFF -
DLIBCXX_ENABLE_ASSERTIONS=OFF -DLIBCXX_ENABLE_SHARED=OFF -DLIBCXX_ENABLE_FILESYSTEM=OFF -
DLIBCXX_ENABLE_THREADS=OFF -DLIBCXX_ENABLE_MONOTONIC_CLOCK=OFF -
DLIBCXX_ENABLE_ABI_LINKER_SCRIPT=OFF -DLIBCXX_ENABLE_EXPERIMENTAL_LIBRARY=ON -
DLIBCXX_INCLUDE_TESTS=OFF -DLIBCXX_INCLUDE_BENCHMARKS=OFF -DLIBCXX_USE_COMPILER_RT=ON -
DLIBCXX_CXX_ABI=libcxxabi -DLIBCXX_CXX_ABI_INCLUDE_PATHS=/home/esi/workspace/llvm-project/libcxxabi/include -
DLIBCXXABI_ENABLE_STATIC_UNWINDER=ON -DLIBCXXABI_USE_LLVM_UNWINDER=ON -DUNIX=1 -
DLIBCXX_TARGET_TRIPLE=armv6m-none-eabi -DLIBCXX_SYSROOT=/home/esi/arm-none-eabi/armv6m-none-eabi
```

Then we compile *libunwind*:

```
        $ cmake -G Ninja ../libunwind -DCMAKE_INSTALL_PREFIX=/home/esi/arm-none-eabi/armv6m-none-eabi -
DCMAKE_TRY_COMPILE_TARGET_TYPE=STATIC_LIBRARY -DCMAKE_SYSTEM_PROCESSOR=arm -
DCMAKE_SYSTEM_NAME=Generic -DCMAKE_CROSSCOMPILING=ON -DCMAKE_CXX_COMPILER_FORCED=TRUE -
DCMAKE_BUILD_TYPE=Debug -DCMAKE_C_COMPILER=/home/esi/arm-none-eabi/bin/clang -
DCMAKE_CXX_COMPILER=/home/esi/arm-none-eabi/bin/clang++ -DCMAKE_LINKER=/home/esi/arm-none-eabi/bin/clang -
DCMAKE_AR=/home/esi/arm-none-eabi/bin/llvm-ar -DCMAKE_RANLIB=/home/esi/arm-none-eabi/bin/llvm-ranlib -
DCMAKE_C_COMPILER_TARGET=armv6m-none-eabi -DCMAKE_CXX_COMPILER_TARGET=armv6m-none-eabi -
DCMAKE_SYSROOT=/home/esi/arm-none-eabi/armv6m-none-eabi/ -DCMAKE_SYSROOT_LINK=/home/esi/arm-none-eabi/armv6m-
none-eabi/ -DCMAKE_C_FLAGS="--target=armv6m-none-eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -g -
Os -ffunction-sections -fdata-sections -fno-stack-protector -fvisibility=hidden -fno-use-cxa-atexit -Wno-unused-command-line-argument -
D_LIBUNWIND_IS_BAREMETAL=1 -D_GNU_SOURCE=1 -D_POSIX_TIMERS=1 -
D_LIBCPP_HAS_NO_LIBRARY_ALIGNED_ALLOCATION -I/home/esi/arm-none-eabi/armv6m-none-eabi/include -
D_LIBCPP_HAS_NO_THREADS=1" -DCMAKE_CXX_FLAGS="--target=armv6m-none-eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -
fshort-enums -mfloat-abi=soft -g -Os -ffunction-sections -fdata-sections -fno-stack-protector -fvisibility=hidden -fno-use-cxa-atexit -Wno-
unused-command-line-argument -D_LIBUNWIND_IS_BAREMETAL=1 -D_GNU_SOURCE=1 -D_POSIX_TIMERS=1 -
D_LIBCPP_HAS_NO_LIBRARY_ALIGNED_ALLOCATION -I/home/esi/arm-none-eabi/armv6m-none-eabi/include -
D_LIBCPP_HAS_NO_THREADS=1" -DCMAKE_EXE_LINKER_FLAGS="-L/home/esi/arm-none-eabi/armv6m-none-eabi/lib/" -
DLLVM_CONFIG_PATH=/home/esi/arm-none-eabi/bin/llvm-config -DLLVM_ENABLE_PIC=OFF -
DLIBUNWIND_ENABLE_ASSERTIONS=OFF -DLIBUNWIND_ENABLE_PEDANTIC=ON -DLIBUNWIND_ENABLE_SHARED=OFF
-DLIBUNWIND_ENABLE_THREADS=OFF -DLLVM_ENABLE_LIBCXX=TRUE -DUNIX=1
```

Then we compile libcxxabi:

```
$ cmake -G Ninja ../libcxxabi -DCMAKE_INSTALL_PREFIX=/home/esi/arm-none-eabi/armv6m-none-eabi -
DCMAKE_TRY_COMPILE_TARGET_TYPE=STATIC_LIBRARY -DCMAKE_SYSTEM_PROCESSOR=arm -
DCMAKE_SYSTEM_NAME=Generic -DCMAKE_CROSSCOMPILING=ON -
DCMAKE_CXX_COMPILER_FORCED=TRUE -DCMAKE_BUILD_TYPE=Debug -
DCMAKE_C_COMPILER=/home/esi/arm-none-eabi/bin/clang -DCMAKE_CXX_COMPILER=/home/esi/arm-none-
eabi/bin/clang++ -DCMAKE_LINKER=/home/esi/arm-none-eabi/bin/clang -DCMAKE_AR=/home/esi/arm-none-eabi/bin/llvm-
ar -DCMAKE_RANLIB=/home/esi/arm-none-eabi/bin/llvm-ranlib -DCMAKE_C_COMPILER_TARGET=armv6m-none-eabi -
DCMAKE_CXX_COMPILER_TARGET=armv6m-none-eabi -DCMAKE_SYSROOT=/home/esi/arm-none-eabi/armv6m-none-
eabi/ -DCMAKE_SYSROOT_LINK=/home/esi/arm-none-eabi/armv6m-none-eabi/ -DCMAKE_C_FLAGS="--target=armv6m-
none-eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -g -Os -ffunction-sections -fdata-sections -
fno-stack-protector -fvisibility=hidden -fno-use-cxa-atexit -Wno-unused-command-line-argument -
D_LIBUNWIND_IS_BAREMETAL=1 -D_GNU_SOURCE=1 -D_POSIX_TIMERS=1 -
D_LIBCPP_HAS_NO_LIBRARY_ALIGNED_ALLOCATION -I/home/esi/arm-none-eabi/armv6m-none-eabi/include" -
DCMAKE_CXX_FLAGS="--target=armv6m-none-eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft
-g -Os -ffunction-sections -fdata-sections -fno-stack-protector -fvisibility=hidden -fno-use-cxa-atexit -Wno-unused-command-
line-argument -D_LIBUNWIND_IS_BAREMETAL=1 -D_GNU_SOURCE=1 -D_POSIX_TIMERS=1 -
D_LIBCPP_HAS_NO_LIBRARY_ALIGNED_ALLOCATION -I/home/esi/arm-none-eabi/armv6m-none-eabi/include" -
DCMAKE_EXE_LINKER_FLAGS="-L/home/esi/arm-none-eabi/armv6m-none-eabi/lib/" -
DLLVM_CONFIG_PATH=/home/esi/arm-none-eabi/bin/llvm-config -DLLVM_ENABLE_PIC=OFF -
DLIBCXXABI_ENABLE_ASSERTIONS=OFF -DLIBCXXABI_ENABLE_STATIC_UNWINDER=ON -
DLIBCXXABI_USE_COMPILER_RT=ON -DLIBCXXABI_ENABLE_THREADS=OFF -
DLIBCXXABI_ENABLE_SHARED=OFF -DLIBCXXABI_BAREMETAL=ON -
DLIBCXXABI_USE_LLVM_UNWINDER=ON -DLIBCXXABI_SILENT_TERMINATE=ON -
DLIBCXXABI_INCLUDE_TESTS=OFF -DLIBCXXABI_LIBCXX_SRC_DIRS=/home/esi/workspace/llvm-project/libcxx -
DLIBCXXABI_LIBUNWIND_LINK_FLAGS="-L/home/esi/arm-none-eabi/armv6m-none-eabi/lib" -
DLIBCXXABI_LIBCXX_PATH=/home/esi/workspace/llvm-project/libcxx -
DLIBCXXABI_LIBCXX_INCLUDES=/home/esi/arm-none-eabi/armv6m-none-eabi/include/c++/v1 -DUNIX=1 -
DLIBCXXABI_SYSROOT=/home/esi/arm-none-eabi/armv6m-none-eabi -DLIBCXXABI_TARGET_TRIPLE=armv6m-none-
eabi -DLIBCXXABI_LIBUNWIND_PATH=/home/esi/arm-none-eabi/armv6m-none-eabi/lib
```

We finally compile fft.c:

```
$ /home/esi/arm-none-eabi/bin/clang fft.c --sysroot=/home/esi/arm-none-eabi/armv6m-none-eabi --target=armv6m-none-
eabi -mcpu=cortex-m0 -mthumb -mabi=aapcs -fshort-enums -mfloat-abi=soft -g -Os -ffunction-sections -fdata-sections -fno-
stack-protector -fvisibility=hidden -Wno-unused-command-line-argument -L/home/esi/arm-none-eabi/armv6m-none-eabi/lib
```