Fachprojekt: Advanced Processor Design (DT/ RF Acceleration)

Immanuel Dick

September 2022

1 Orientierung und Kurswechsel

Ursprünglich war für diese Themengruppe der Schwerpunkt gewählt, auf Basis der Arbeit vom letzten Fachprojekt die DecisionTree- und RandomForest-Implementierung auf systolischen Arrays auszutesten. Dazu sah der Projektplan unter anderem ein Code-Review zum Einstieg vor, gefolgt von einem minimalen Proof-of-Concept anhand eines einzelnen Decision Trees. Auf den Erkenntnissen dieser Vorstufen sollten systolic Arrays dann schließlich auf dem gesamten Random Forest zum Einsatz kommen. Der Effekt dieser Umsetzungsweise sollte dann anhand von Laufzeittests gemessen und verglichen werden.

Im Verlauf des Fachprojektes wurde die Leistung jedoch einseitig erbracht, ohne dass sich eine Änderung der Situation abzeichnete. Daher wurde der Schwerpunkt auf einen Themenbereich gelenkt, der für die Bearbeitung durch eine einzelne Person besser geeignet schien.

Der neue Plan sah vor, ein ausführliches Code-Review des von der Vorgänger-Gruppe erarbeiteten VHDL Codes durchzuführen und insbesondere die Nachvollziehbarkeit auf Basis logischer Grundbausteine zu erhöhen.

2 Ergebnisse

2.1 Buildscript

Es wurde zunächst ein relativ einfaches Buildscript erstellt, um im weiteren Verlauf sowohl die Evaluation als auch Simulation des VHDL-Codes zu vereinfachen. Das Buildscript eignet sich im Allgemeinen auch für weitere Projekte und kann Kursteilnehmer/innen des nächsten Fachprojekts zur Verfügung gestellt werden.

Die Bedienung besteht aus drei konfigurierbaren Feldern innerhalb der Skriptdatei und einem optionalen Argument `--verbose` (kurz `-v`). Dieses Argument aktiviert erweiterte Ausgabe von zusätzlichen Fehlerinformationen.

Die in der Skriptdatei konfigurierbaren Felder seien im Folgenden aufgelistet und kurz erklärt:

`SRCS`

Array aus case-sensitiven Strings, separiert durch Leerstellen.
Das Skript ließt das Array von rechts nach links und konstruiert die vollständigen Pfade der Dateien einer VHDL-Einheit `file` wie folgt;

- Quelldatei: `SRCSDIR/file.vhdlSRCSDIR/file.vhdl`
- Testbench: `SRCSDIR/testbenches/file_tb.vhdl` (optional)
- VCD-Datei: `WORKDIR/file.vcd` (generiert)

`SRCSDIR`

Ordnerpfad in dem die VHDL-Quell- und Testbench-Dateien liegen.
Relativ zum Skript.

`WORKDIR`

Ordnerpfad in den kompilierte (VCD-)Dateien gespeichert werden sollen.
Relativ zum Skript.

2.2 Atomare Logik und generische Parameter

Jede Komponente der ursprünglichen Implementierung von Marji et. al wurde in seiner Komplexität bis auf die Logikgatter herunter vereinfacht. So wurde z.B. `node.vhdl` neu hinzugefügt um die Komplexität von `decisionTree.vhdl` zu reduzieren. Es wurden neue Komponenten (wie `counter.vhdl`) implementiert, die zuvor durch die high-level Implementierung verdeckt wurden.

Ferner wurden alle Komponenten an sinnvollen Stellen mit generischen Parametern versehen. Dies erlaubt die dynamische Synthetisierung abhängig vom Datensatz ohne größeren Mehraufwand.

2.3 Dokumentation und Komplexitätsberechnung

Alle Dateien wurden hinsichtlich ihrer Dokumentation grundlegend neu aufgearbeitet. Ein- und Ausgabeparameter sowie generische Argumente wurden beschrieben; Funktion und Handhabe von VHDL-Einheiten wurde in einheitlich formattierten Kommentaren veranschaulicht.

Exemplarisch wurde die Berechnung der Gatter-Komplexität der `majorityVote`-Komponente in der Dokumentation ergänzt. Analog zum Beispiel wurde die Komplexität für viele weitere Komponenten bestimmt.

2.4 Testbenches

Mit Ausnahme der Package-Datei `randomForestTypes.vhdl` wurden für alle Komponenten eigene Testbenches deklariert. Hierbei wurde bei den grundlegenden Komponenten angesetzt und auf komplexere Komponenten hochgearbeitet. Bedauerlicherweise konnte die Testbench für `randomForest.vhdl` aufgrund Zeitmangels nicht vervollständigt werden.

3 Ausblick

Zwar sind die Ergebnisse dieses Fachprojekts doch recht weit von den Vorschlägen der letzten Gruppe entfernt. Dennoch konnte der Weg für folgende Arbeiten geebnet werden und die Funktionsweise der Decision Trees bzw. Random Forests auf ihrer logischen Ebene durch eine atomare Implementierung näher erleuchtet werden.

In Zukunft kann auf Grundlage dieses und des vorigen Fachprojekts insbesondere der Blick auf die Skalierbarkeit von Random Forests gelegt werden. Die Verwendung atomarer Komponenten könnte die Implementierung auf einem FPGA erlauben bzw. vereinfachen. Ebenso bleibt offen, in welcher Weise ein Random Forest von der Nutzung von systolischen Arrays profitiert.

Diese Ansätze möchte ich um die Anregung ergänzen, die Einbindung in eine MIPS-Architektur zu betrachten und mögliche Use-Cases zu erforschen.