# Posudek Vedoucího Bakalářské Práce

Matematicko-fyzikální fakulta Univerzity Karlovy

**Autor práce:** Maximilian Kulikov
**Název práce:** Emulátor zvukových syntezátorů
**Rok odevzdání:** 2022
**Studijní program:** Informatika
**Studijní obor:** IOI
**Autor posudku:** David Klusáček
**Pracoviště:** ÚFAL/MFF UK

## K celé práci

Cílem práce byl systém pro popis syntezátorů zvuku a jejich následnou emulaci. Měl obsahovat ovládací GUI, které by interagovalo s uživatelem, posílalo data na zvukovou kartu přes ASIO driver a případně četlo MIDI vstup. Syntezátor sám měl být popsán svým blokovým zapojením, které by představovalo další vstup programu.

Aby byl program využitelný profesionálními hudebníky, byl kladen důraz na nízkou latenci. Pro její dosažení bylo navrženo řešení, kdy by blokové zapojení syntezátoru bylo převedeno do jednoduchého programu v jazyce C, který by po zkompilování a dynamickém přilikovnání k hlavnímu programu poskytl funkci, která by vždy přečetla proměnné reprezentující vstupy, odsimulovala jeden časový krok a vrátila další syntezovaný vzorek signálu.

Odtud vyvstala potřeba jednoduchého jazyka, ve kterém by bylo možné popsat zapojení bloků syntezátoru pomocí výrazů operujících nad signály v jednotlivých uzlech obvodu, čímž by byl umožněn relativně přímý přepis schematu zapojení. Signály by byly udržovány v cyklických bufferech, aby operátory mohly kombinovat aktuální vzorek signálu s minulými.

Hlavní problém původní odevzdané verze byl, že návrh jazyka studenta zcela pohltil a díky tomu již nestihl implementovat ostatní části systému, dokonce ani srozumitelně popsat samotný jazyk. Opravená verze již demonstruje napojení na MIDI, jazyk je též popsán dostatečně, dokonce je uveden i strohý pokus o měření latence.

Největším nedostatkem práce je nyní přílišná složitost jazyka. To je ovšem očekávané, protože po dohodě s oponentem měla oprava spočívat v dopsání dokumentace a napojení na MIDI. Nikoliv v přepracování navrženého jazyka.

Stejně se ale nelze ubránit dojmu, že místo jednoduchého konfiguračního jazyka pro popis signálového obvodu, který by mohl mít náročnost implementace jen o třídu vyšší než překladač aritmetických výrazů, vznikl značně abstraktní jazyk, který bude pro svou složitost jen stěží přijat předpokládanou cílovou skupinou uživatelů.

Že míra abstrakce neznamená automaticky větší expresivitu je vidět z toho, že současná implementace neumožňuje popis IIR filtru, nebo obecně, jakýhkoliv zpětných vazeb v zapojení. Autor si je však toho vědom, a v kapitole 6.4.4 navrhuje řešení jak tuto vlastnost snadno přidat. Celkově lze návrh jazyka shrnout výrokem, který je připisován N. Wirthovi: *"The hardest thing in designing a programming language is to decide what to leave out."*. Přes tyto nedostatky práce splňuje zadání a doporučuji ji k obhajobě.

**Obtížnost zadání:**[1]: 2
**Splnění zadání:** 3
**Rozsah práce:** 1

## K textové části:

Práce je psána česky, vkusně vysázena v TEXu a rozdělena na 6 částí: Úvod, Návrh, Demonstrace, Specifikace jazyka Cynth, Implementace a Evaluace.

[1] 1=lepší, 2=OK, 3=horší, 4=nevyhovuje

Po čtivém úvodu do problematiky syntézy zvuku následuje návrh jazyka a pak návod jak systém spustit, s několika příklady. Potom následuje neformální specifikace jazyka, následovaná implementačními poznámkami a popisem měření latence.

Bohužel, podobně jako minule, mi student poskytl text k připomínkám až těsně před odevzdáním, takže v práci zbytečně zůstalo množství překlepů. Některé jsou lexikální (*"konfuguračí jazyk"*, dokonce i ve vzorcích: na straně 7 jsou IIR i FIR koeficienty nazývány kolidujícími symboly $a_i$ ale ve vzorci (1.4) už je správně $b_i$ pro FIR a $a_i$ pro IIR). Jiné překlepy jsou sémantické, např. na straně 24: *"Celočíselné dělení na rozdíl od C vždy provádí zaokrouhlení směrem k nule. Taková sémantika je vhodnější hlavně kvůli potřebě indexování cyklických bufferů."*. Jenže dělení v C zaokrouhluje k nule. Autor měl zjevně na mysli zaokrouhlování směrem dolů, jak se můžeme přesvedčit v implementaci v souboru `internal/compiler/inc/sem/operations.hpp`, kde je to správně.

Často podobné překlepy znesnadňují pochopení jazyka, např. na straně 21 je příklad

```
buffer [16] c = fn (Int t) b[0.] + b[1.] + b[2.] / 3.;
```

Přitom ale není definováno co znamená indexování float číslem. Kromě toho je v odstavci před tím napsáno, že *"Aktuální vzorek bude reprezentován indexem 0. Starší vzorky pak budou na menších indexech."*. To by naznačovalo, že buffery lze indexovat pouze nekladnými indexy a tedy místo `b[1.]` mělo být `b[-1]`. Podobně na straně 24 je příklad

```
sum(for (Int e in a, Int f in b[n - 1 to -1]) a[i] * b[n - i])
```

kde není vysvětleno k čemu slouží nepoužité proměnné `e` a `f` a co je proměnná `i`.

**Formální úprava:** 2
**Struktura textu:** 2
**Analýza:** 3
**Vývojová dokumentace:** 2
**Uživatelská dokumentace:** 2

## Implementační část

Práce je implementována v jazyce C++ s využitím generátorů `flex` a `bison`. Trochu na škodu je použití moderních vlastností C++20, které vylučují kompilaci projektu staršími verzemi C++. Celkový rozsah zdrojového kódu je enormní. Program sestává z 27570 řádků (955 KiB) ve 173 souborech.

Přestože se autor snažil o formální styl, který dokonce definoval v dokumentaci (jak se mají pojmenovávat různé entity, jak odsazovat, apod.), nestačilo to k tomu, aby bylo snadné se v programu zorientovat. Krom velké délky je příčinou i extensivní využívání šablon a netriviální objektová struktura programu.

**Kvalita návrhu:** 3
**Kvalita zpracování:** 2
**Stabilita implementace:** 2

**Celkové hodnocení:** Velmi dobře
**Práci navrhuji na zvláštní ocenění:** Ne

28.8.2022 David Klusáček