

Uporaba v mikrokrmilnik vgrajenih programirljivih logičnih vezij

Andrej Kenda¹

¹Ime in naslov organizacije prvega avtorja
E-pošta: ak8655@student.uni-lj.si

Povzetek

FPGA integrated circuits are often indispensable in practice, but their implementation is often demanding. In this article, we explore one of the answers to this problem offered by the manufacturer Texas Instruments. In their microcontrollers, there is an additional coprocessor, called "CLB". The composition of said coprocessor is described, which is essential for the use of such system.

1 Uvod

Pri oblikovanju vdelanih sistemov se zaradi preprostega programiranja in cenovne ugodnosti najpogosteje poslužujemo mikrokrmilnikov različnih proizvajalcev. Na začetku (včasih "celo" na sredini) oblikovalnega procesa, včasih preveliki ponudbi, izberemo čip, ki najbolje ustreza našim zahtevam.

Obstajajo pa tudi aplikacije, ki jim konvencionalni mikrokrmilniki niso kos. Tu nastopi iskanje drugačnih rešitev. Če imamo srečo, morda odkrijemo kakšen ASIC (angl. application specific integrated circuit), ki zadošča našim potrebam, vendar je na določenih področjih teh dokaj malo. V določenih aplikacijah nam ne preostane drugega oz. za boljšo rešitev odkrijemo FPGA (angl. field-programable gate array). Ti nam omogočajo postavitve sistema na nizkem nivoju, ki ga lahko prikojimo natanko našim zahtevam. FPGA v veliko aplikacijah namestimo poleg prej omenjenega mikrokrmilnika in ga uporabimo kot dodaten procesor.

Kljub temu, da so FPGA čipi precej zmogljivi, pa nastopi težava pri sami izvedbi. Kadra, ki ima globoko znanje takšnih sistemov je zaradi zahtevnosti precej malo. Poleg tega pa uporaba dveh čipov za pri številnih aplikacijah nemalokrat ni idealna, saj ti čipi dostikrat zahtevajo uporabo zunanjega RAM in Flash spomina, za komunikacijo s katerima smo odgovorni sami. Razne komunikacije med posameznimi moduli običajno terjajo največ razvojnega časa, hkrati pa je poleg FPGA ponavadi prisoten tudi mikrokrmilnik, ki je hkrati dodatna finančna obremenitev in še ena linija komunikacije za implementacijo.

Na našete težave je proizvajalec Texas Instruments odgovoril s koprocesorjem CLB (angl. configurable logic block), ki je vdelan v nekaj njihovih mikrokrmilnikov. Ta naj bi po proizvajalčevih trditvah tako nadomestil dodaten FPGA (in s tem razvijalcu prihranil marsikatero

uro), kot tudi omogočil programiranje s preprostim vmesnikom [1].

2 Predstavitev CLB

Slika 1 prikazuje shematski prikaz komponent v CLB koprocesorju, ki je del mikrokrmilnika F28379D, proizvajalca Texas Instruments. Na sliki je razvidno, da je omenjeni CLB sestavljen iz štirih med seboj enakih podsklopov (angl. Tile). Vsak podsklop je sestavljen iz procesorja (angl. CELL) ter vmesnika za "priklop" signalov iz matične naprave (angl. CPU I/F).

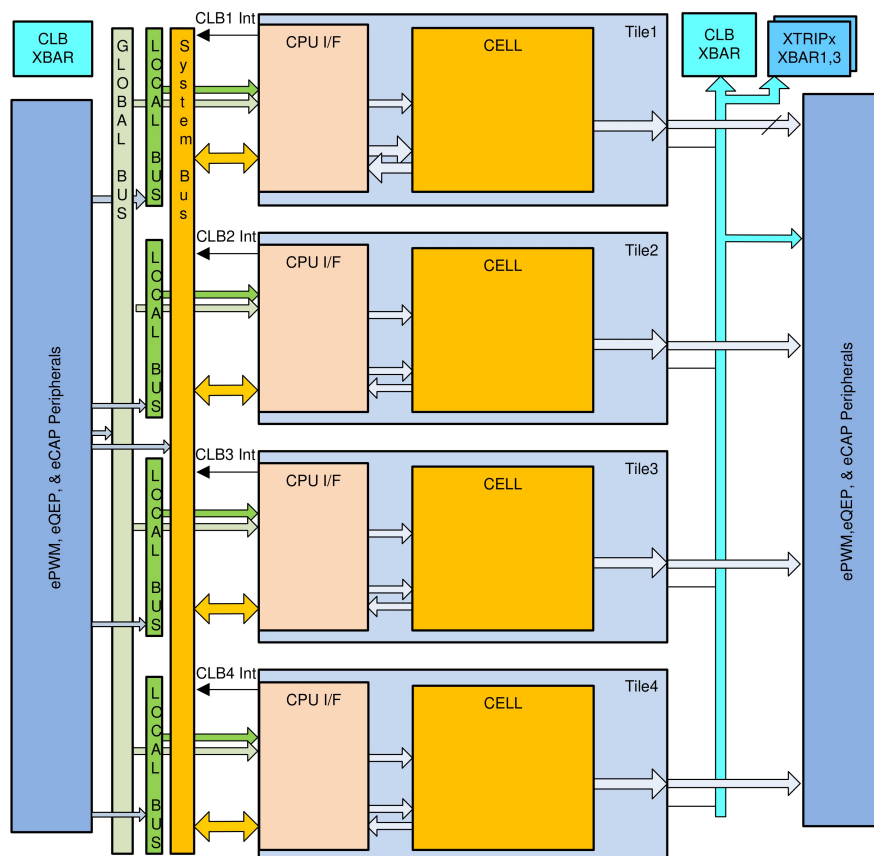
V CLB lahko vhodne signale pripeljemo preko treh vodil, lokalnega, globalnega in sistemskega vodila. Teh se za vhode v CLB poslužujemo glede na samo postavitev ostalih dodatkov, ki so poleg CLB-ja prisotni na čipu [5, Pogl. 26.3]. Posebnost omenjenega mikrokrmilnika je tudi prisotnost t.i. "crossbar"-jev, ki omogočajo dovod vhodnih signalov preko lokalnega in globalnega vodila, ter odvod signalov iz CLB-ja. Ti signali lahko izvirajo iz različnih dodatkov, kot so eCAP, ePWM, GPIO... iz CLB-ja pa lahko v prav te dodatke tudi "pripeljemo" izhodni signal iz CLB-ja, kot vhod.

Interakcija pa ni omejena le z dodatnimi napravami in matičnim procesorjem, temveč je mogoča tudi med različnimi podsklopi CLB-ja. Možno pa je tudi proženje prekinitve, na katere lahko procesor ustrezno reagira.

3 Zgradba CLB modulov

V poglavju 2 je omeneno, da je sam CLB sestavljen iz štirih podsklopov. Na sliki 2 je videti, da vsak podsklop vsebuje [5, Pogl. 26.4]:

- 3 štirivhodne LUT4 (angl. 4-input lookup table),
- 8 trivhodnih izhodnih LUT3 (angl. 3-input lookup table),
- 3 števec,
- 3 FSM (angl. finite state machine),
- 1 HLC (angl. high level controller) in
- 1 nastavljen preklopni blok.



Slika 1: Shematski prikaz komponent CLB koprocesorja [5, Pogl. 26.2]

3.1 LUT moduli

LUT4 modul omogoča modulacijo signala preko logičnih operacij. Vhodne signale (slika 3 - "IN0"- "IN3") lahko preko enačbe z "AND", "OR", "NOT" in "XOR" operacijami odvedemo v izhodni signal [2, Pogl. 3.3].

Razlika med "LUT4" in izhodnim "LUT3" modulom je le, da ima slednji tri namesto štirih vhodov, ter je njegov izhod vezan direktno na izhod CLB-ja, kar je tudi ena izmed njegovih funkcij [5, Pogl. 26.4.4-26.4.5].

3.2 Števci

Števec je precej kompleksen modul v CLB-ju, ki ga lahko nastavimo za delovanje kot števec, seštevalnik, komparator ali operator zamika. S slike 4 je moč razbrati štiri funkcijske vhode; "RESET", "MODE 0", "MODE 1" in "EVENT". Prvi od teh vhodnih signalov ob visoki vrednosti števec postavi na vrednost 0, "MODE 0" ob visoki vrednosti omogoči štetje, "MODE 1" nastavi smer štetja (visoko - navzgor, nizko - navzdol), signal "EVENT" pa ob pozitivnem robu sproži dogodek, na katerega ta odreagira z nastavljen računsko operacijo, katere parametre nastavimo z drugima signaloma (slika 4 - "Static controls" in "LOAD VALUE"). Do vsebine števca lahko dostopamo preko HLC modula (pogl. 3.4), kjer je ta označena z zaporedno številko števca (npr. števec 1 - C1).

Kot izhod iz števca lahko spremljamo signale "ZERO", "MATCH1" in "MATCH2". Prvi prevzame vi-

soko vrednost kadar je vrednost števca 0, ostala dva pa se vklopita, kadar je vrednost števca enaka nastavljeni pripadajoči vrednosti. Ko števec doseže svojo maksimalno vrednost (2^{32}) ta "prelije" in začne šteti od 0. Temu obnašanju se ponavadi želimo izogniti [5, Pogl. 26.4.2].

3.3 FSM moduli

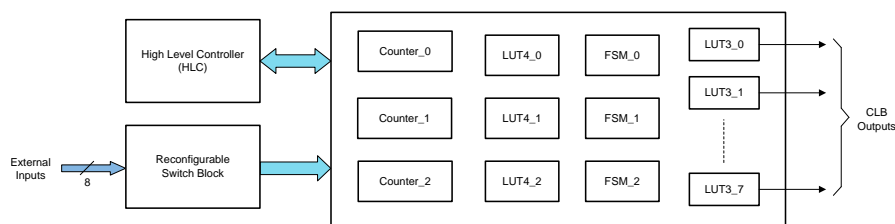
FSM modul omogoča implementacijo avtomata stanj. Le to se izvede preko dveh internih LUT blokov; "Output LUT", "S0 Next State LUT" in "S1 Next State LUT" (slika 5), te so po zgradbi povsem enake opisanim v poglavju 3.1. Za implementacijo avtomata stanj uporabljamo le slednji LUT tabeli, ki služita interni modulaciji signalov in operirata z zunanjima signaloma "EXT_IN0", "EXT_IN1" in vhodoma "S0" in "S1", ki prevzameta prejšnjo vrednost izhoda pripadajoče tabele.

Če nimamo potrebe po implemetaciji avtomata stanj, se lahko poslužimo tudi "Output LUT", in modul uporabimo kot običajno LUT tabelo (izhod "FSM_LUT_OUT").

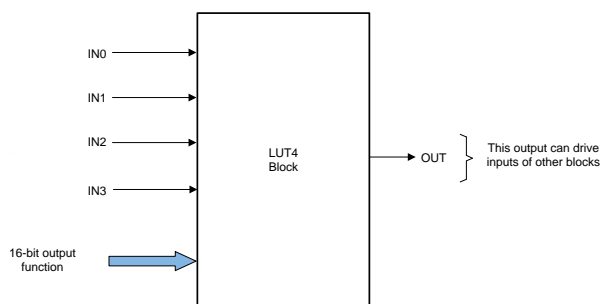
Če aplikacija ne potrebuje uporabe FSM modula, se lahko poslužimo še vhodov "EXTRA_EXT_IN0" in "EXTRA_EXT_IN1". V tem primeru celoten modul deluje kot štirivhodni LUT modul.

3.4 HLC modul

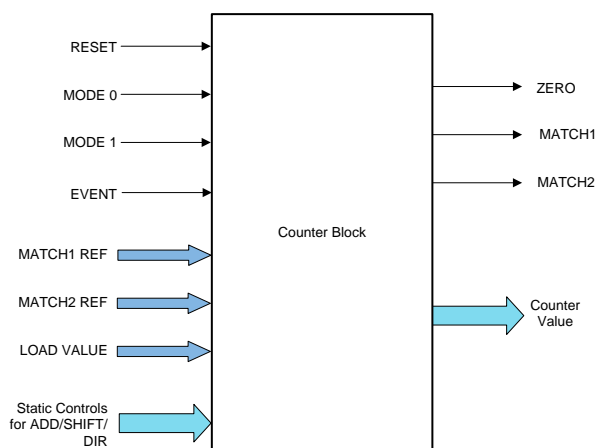
HLC modul je za razliko od ostale naprave veliko bolj kompleksen. S slike 6 je iz bloka za procesiranje dogod-



Slika 2: Shematski prikaz komponent CLB koprocetorja [5, Pogl. 26.2]



Slika 3: Shematski prikaz štirivhodnega LUT4 modula CLB koprocetorja [5, Pogl. 26.4.4]



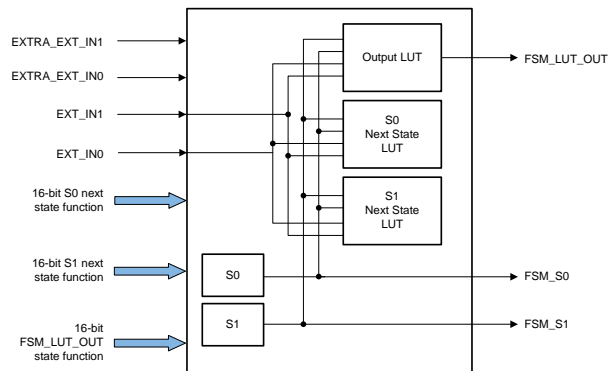
Slika 4: Shematski prikaz števca CLB koprocetorja [5, Pogl. 26.4.2.1]

kov in štirih registrov, ki služijo za shranjevanje spremenljivk "R0"- "R3"[5, Pogl. 26.4.6].

Modul lahko direktno upravljamo tudi z matičnim procesorjem, s katerim modul lahko komunicira tudi preko "push-pull" metode zapisovanja v skupni spomin.

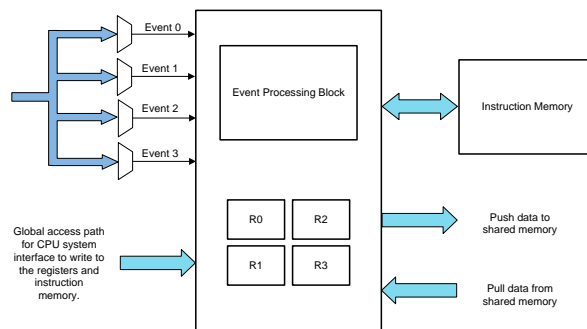
Operiramo lahko s prej omenjenimi spremenljivkami "R0"- "R3", z vrednostmi števecv "C1"- "C2" in z obema "MATCH" vrednostmi v števcih. Za proženje dogodkov ima modul štiri vhodne signale. Vsak od teh sproži sebi pripadajoč program, zapisan v instrukcijskem spominu. Ta lahko na omenjenih spremenljivkah izvede naslednje operacije [5, Pogl. 26.4.6.2]:

- ADD/SUB - seštevanje in odštevanje,
- MOV/MOV_T1/MOV_T2 - premikanje,



Slika 5: Shematski prikaz FSM modula CLB koprocetorja [5, Pogl. 26.4.3]

- PUSH/PULL - pisanje in branje v skupnem spominu,
- INTR - proženje prekinitev.



Slika 6: Shematski prikaz HLC modula CLB koprocetorja [5, Pogl. 26.4.6]

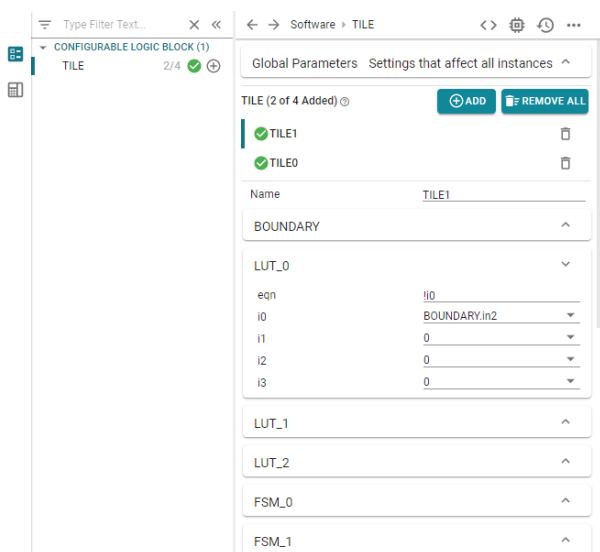
3.5 Preklopni modul

Ta modul je namenjen izbiri vhodnih signalov, ki lahko izvirajo iz zunanjih virov, drugih CLB podsloptov (angl. tile) ali pa signal generira modul sam. Zadnji vir se uporablja le za simulacijske namene. Modul je v orodju "CLB Tool" (Pogl. 4) zaradi postavitve predstavljen pod imenom "BOUNDARY" [2, Pogl. 3.3].

4 Uporaba CLB

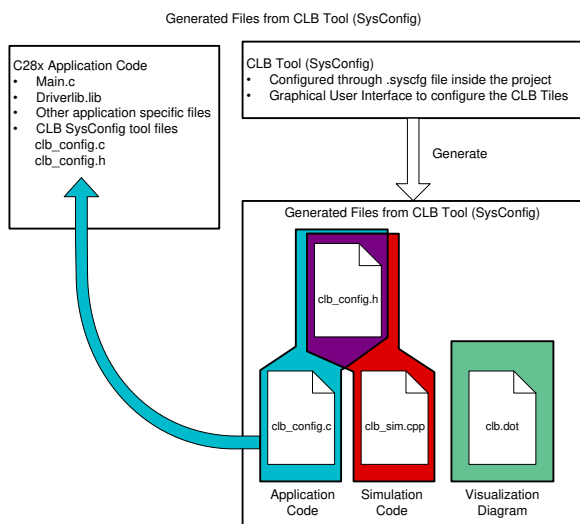
Za samo programiranje CLB koprocetorja v praksi je proizvajalec Texas Instruments postavil grafični vmesnik

“CLB Tool” (slika 7), ki nam preko izbire določenih parametrov zgenerira kodo za uporabo v našem projektu.



Slika 7: Grafično okolje “CLB Tool”

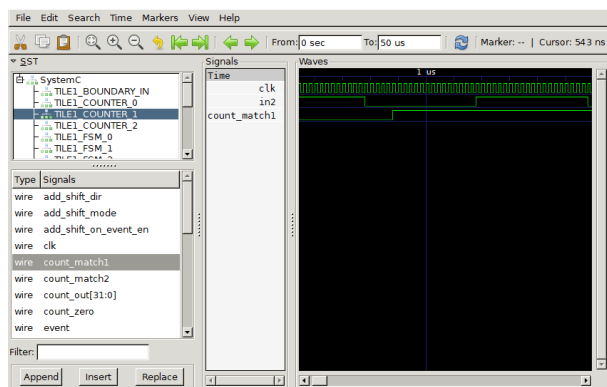
Grafični vmesnik za vsako od komponent, naštetih v poglavju 3, dovoljuje izbiro opisanih parametrov, vhodnih in izhodnih signalov ter splošno konfiguracijo modulov. Kot je prikazano na sliki 8, orodje generira 2 aplikacijski datoteki; “clb_config.h” ter “clb_config.c”. Ti datoteki lahko seveda brez težav vključimo v naš “C” program.



Slika 8: Struktura projekta pri uporabi orodja “CLB Tool” [2, Pogl. 1]

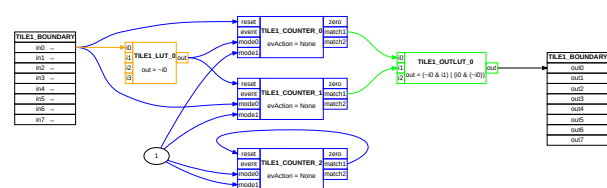
Poleg aplikacijske kode, pa nam orodje dovoljuje tudi generiranje simulacijskih datotek. Te se na koncu prevedejo v datoteko “CLB.vcd”, ki omogoča vpogled v notranje signale CLB preko zunajega programa za prikazovanje grafov (slika 9).

Orodje avtomatsko generira tudi datoteko za grafični pregled povezav v koprocesorju. Ta se prevede v obliko “.html”, ki jo lahko odpremo v vsakem brskalniku (slika



Slika 9: Pregled notranjih signalov CLB-ja s programom “GTK Wave”

10)[2, Pogl. 1].



Slika 10: Primer generiranega “.html” diagrama

5 Zaključek

Za programerja FPGA vezij so stvari, opisane v tem članku verjetno precej domače. Lahko pa opazimo da sam CLB ni le FPGA vdelan v mikrokrmilnik, vendar prinaša tudi nekatere novosti (npr. HLC), zaradi katerih je marsikatera aplikacija dokaj lažja za implementacijo.

Poleg tega so na voljo tudi precej močna tudi orodja v opisanem ekosistemu. Grafično orodje omogoča uporabo tudi manj večšim programerjem. Dodatne generirane simulacijske datoteke pa so v veliko situacijah nepogrešljive, saj s samim opazovanjem zunanjih signalov težko odkrijemo morebitno napako. Enako velja za generirani diagram, ki je (če že ne drugače) uporaben za odkrivanje napačnih in nepotrebnih povezav med posameznimi moduli v koprocesorju.

Vendar pa tudi CLB, kot vsaka stvar ne pride brez pomanjkljivosti. Skoraj vsaka “izboljšava” nad FPGA s seboj prinese tudi kakšno frustracijo za programerja; grafični vmesnik “CLB Tool” je na prvi pogled res prijazen, a se zaradi velike abstrakcije kaj hitro izgubimo med nešteti nastavitvenimi meniji.

Kljub temu pa lahko rečemo, da je takšna izvedba dodatnega modula zelo smiselna. Nema lokrat se znajdemo v situaciji, kjer bi bilo potrebno modularizirati nek signal, vendar je samo zanj uporaba dodatnega čipa nesmiselna. Poleg tega pa je s stališča delodajalcev velikokrat bolj smiselno uporabiti rešitev, ki jo lahko implementira obstoječ kader, kot pa iskanje znanja izven podjetja.

6 Dodatno branje

Več informacij o sami implementaciji CLB je opisanih v [3]. Za migracijo obstoječih FPGA projektov na CLB pa je precej dober uvod vir [4].

Literatura

- [1] *C2000™ Configurable Logic Block (CLB) introduction*. Texas Instruments Inc. 2021. URL: <https://training.ti.com/c2000-configurable-logic-block-clb-introduction>.
- [2] *CLB Tool*. SPRUIR8A. Rev. A. Texas Instruments Inc. ZDA, sep. 2019. URL: https://www.ti.com/lit/ug/spruir8a/spruir8a.pdf?ts=1608498444631&ref_url=https%253A%252F%252Ftraining.ti.com%252Fclb-training-c2000-mcus.
- [3] Nima Eskandari. *Designing With the C2000™ Configurable Logic Block (CLB)*. SPRACL3. ZDA, avg. 2019. URL: https://www.ti.com/lit/an/spracl3/spracl3.pdf?ts=1608544448925&ref_url=https%253A%252F%252Ftraining.ti.com%252F.
- [4] Peter Galicki. *How to Migrate Custom Logic From an FPGA/CPLD to C2000™ Microcontrollers*. SPRACO2A. Rev. 2020-7. Texas Instruments Inc. ZDA, sep. 2019. URL: https://www.ti.com/lit/an/spraco2a/spraco2a.pdf?ts=1618290495856&ref_url=https%253A%252F%252Ftraining.ti.com%252F.
- [5] *TMS320F2837xD Dual-Core Microcontrollers Technical Reference Manual*. SPRUHM8I. Rev. 2019-9. Texas Instruments Inc. ZDA, dec. 2013. URL: https://www.ti.com/lit/ug/spruhm8i/spruhm8i.pdf?ts=1616431731212&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTMS320F28377D.