CSEE 6861
Prof. Nowick

Kernels, Co-Kernels + Extraction Examples

This handout covers kernels/co-kernels, Brayton/McMullen's theorem, + multi-cube + single-cube extraction.
(these examples were presented in class in a previous lecture.)

---

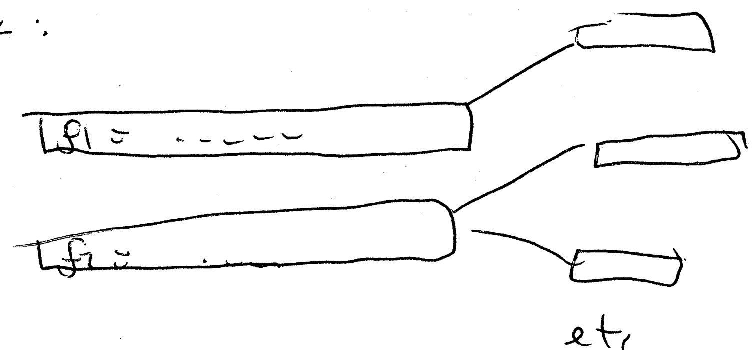
we'll start with multi-cube extraction, then later do single-cube extraction.

multi-cube extraction

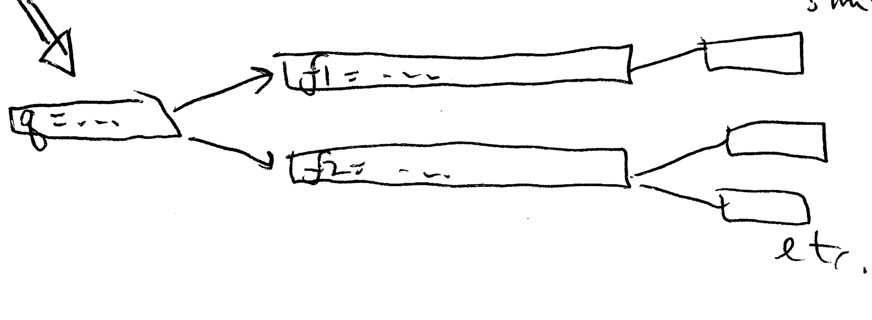
Ex. #1 we are given 2 local functions:

$$f1 = abcd + abce + abf + ag + hi$$
$$f2 = cdhk + cehk + hkji + an$$

In our multi-level representation, these are 2 separate nodes in a given logic network:



Our goal: extract a multi-cube divisor g, allowing us to simplify nodes f1 + f2



CSEE 6861
Prof. Nowick

Kernels, Co-Kernels + extraction (cont.)

multi-cube extraction (cont.)

Compute kernels/co-kernels of f1 + f2:

$f1 = abcd + abce + abf + ag + hi$

Co-kernels/Kernels(f1):

- f1-1) $1 \cdot (abcd + abce + abf + ag + hi)$ — 1 = trivial co-kernel; (abcd+abce+abf+ag+hi) = trivial kernel = f → level-3
- f1-2) $a \cdot (bcd + bce + bf + g)$ — a = co-kernel; (bcd+bce+bf+g) = kernel → level-2
- f1-3) $ab \cdot (cd + ce + f)$ — ab = co-kernel; (cd+ce+f) = kernel → level-1
- f1-4) $abc \cdot (d+e)$ — abc = co-kernel; (d+e) = kernel → level-0

$f2 = cdhk + cehk + hkji + an$

Co-kernels/Kernels(f2):

- f2-1) $1 \cdot (cdhk + cehk + hkji + an)$ — 1 = trivial co-kernel; (cdhk+cehk+hkji+an) = trivial kernel = f → level-2
- f2-2) $hk \cdot (cd + ce + ji)$ — hk = co-kernel; (cd+ce+ji) = kernel → level-1
- f3-3) $chk \cdot (d+e)$ — chk = co-kernel; (d+e) = kernel → level-0

Now, use Brayton/McMullen's theorem to find all candidate multi-cube extracted divisors.

Method: kernel intersection = find all non-trivial multi-cube kernel intersections (between f1 and f2):

Examples:

a) kernel f1-3 ∩ kernel f2-2 = {cd, ce} = **cd+ce**

b) kernel f1-4 ∩ kernel f3-3 = {d, e} = **d+e**

CSEE 6861
Prof. Nowick

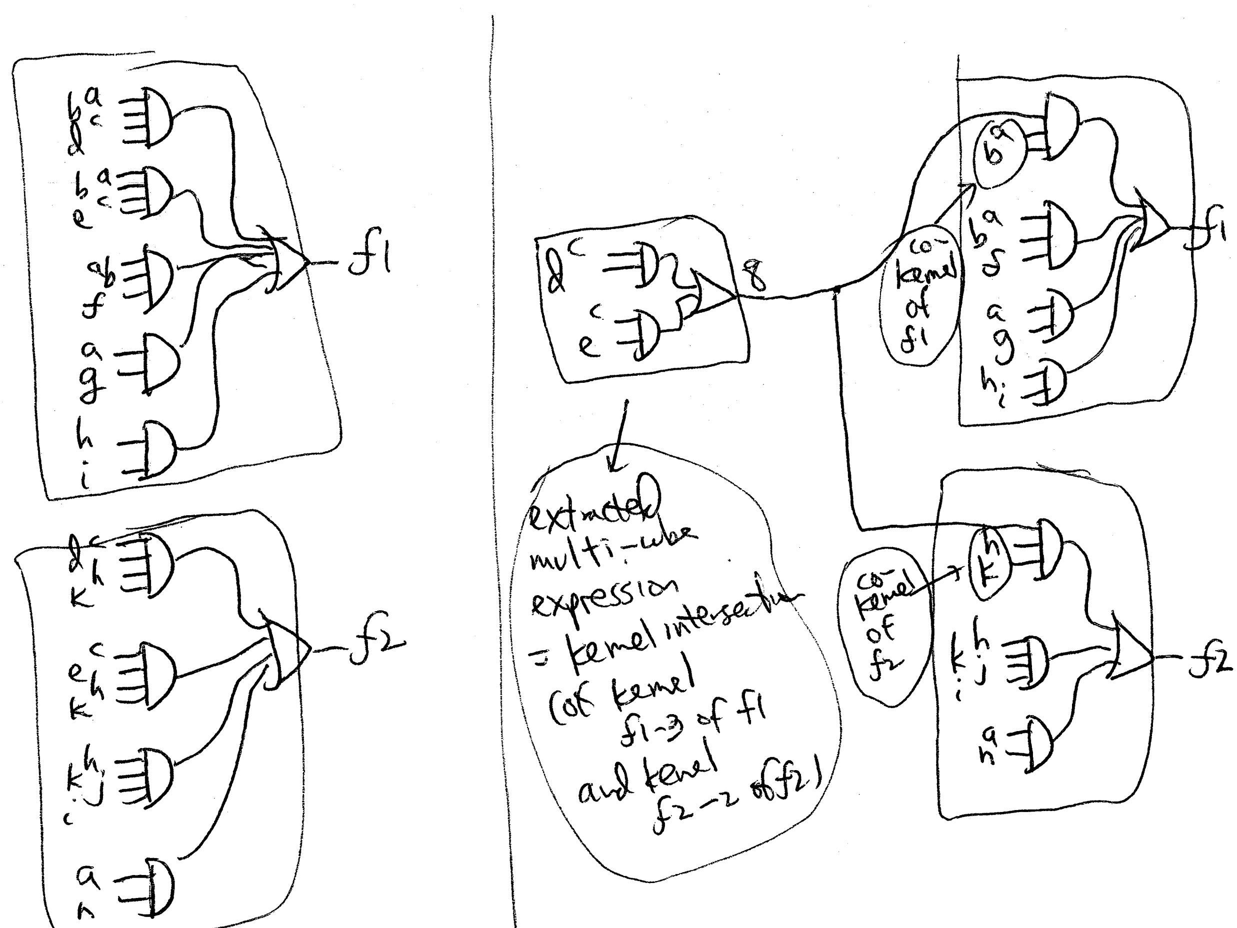
Kernels, Co-kernels & extraction (cont.)

multi-cube extraction (cont.)

Let's pick $g = cd + ce$ as our multi-cube divisor. (To be legitimate, this must contain 2 or more cubes — it does —, & contribute to 2 or more local functions — it does [f1, f2]).

Note the circuit structure before & after multi-cube extraction:

Before multi-cube extraction: ⇒ After multi-cube extraction:



Kernels, Co-kernels + extraction (cont.)

Next, we'll do a single-cube extraction example, re-using local function f2 but also with a new local function f3.

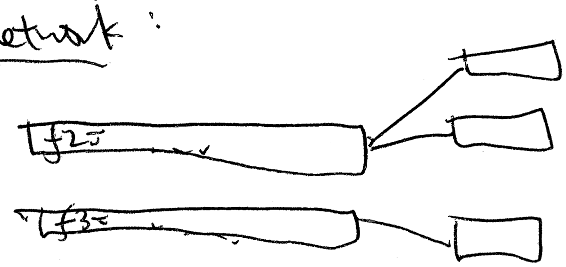
single-cube extraction

Ex. #2 We are given 2 local functions:

(same as before) → $f2 = cdhk + cehk + hkji + an$

$f3 = abchi + cdhi + gchi + j$

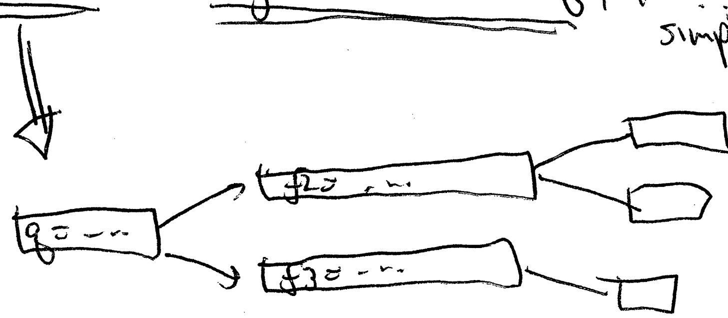
In our multi-level representation, these are 2 separate nodes in a given <u>logic network</u>:



etc.

Our goal: <u>extract</u> a <u>single-cube divisor</u> q, allowing us to simplify nodes f2 + f3

⇓



etc.

CSEE 6861
Prof. Nowick

Kernels, co-kernels, & extraction (cont.)

Single-cube extraction (cont.)

Compute kernels / co-kernels of f2 & f3:

Co-kernels / kernels (f2): <see p. 2>

	co-kernels		kernel
f2-1)	1	·	(cdhk + cehk + hkji + an)
f2-2)	hk	·	(cd + ce + ji)
f2-3)	chk	·	(d + e)

Co-kernels / kernels (f3):

	co-kernel		kernel
f3-1)	1	·	(abchi + cdhi + gchi + j)
f3-2)	chi	·	(ab + d + g)

Now use a variant of Brayton/McMillen's theorem to find all candidate single-cube extracted divisors.

Method: co-kernel intersection = find all non-trivial (single-cube) co-kernel intersections (between f1 and f2):

Examples: only one! =

co-kernel f2-3 ∩ co-kernel f3-2

= chk ∩ chi = **ch**

Kernels, co-kernels, + extraction (cont.)

Single-cube extraction (cont.)

Note the circuit structure before/after single-cube extraction. [In this case, the intersected co-kernel is extracted, while for multi-cube extraction on p. 3 the intersected kernel is extracted!]

Before single-cube extraction: ⇒ After single-cube extraction: