Optimizing Sequential Cycles through Shannon Decomposition and Retiming

Cristian Soviani

Olivier Tardieu

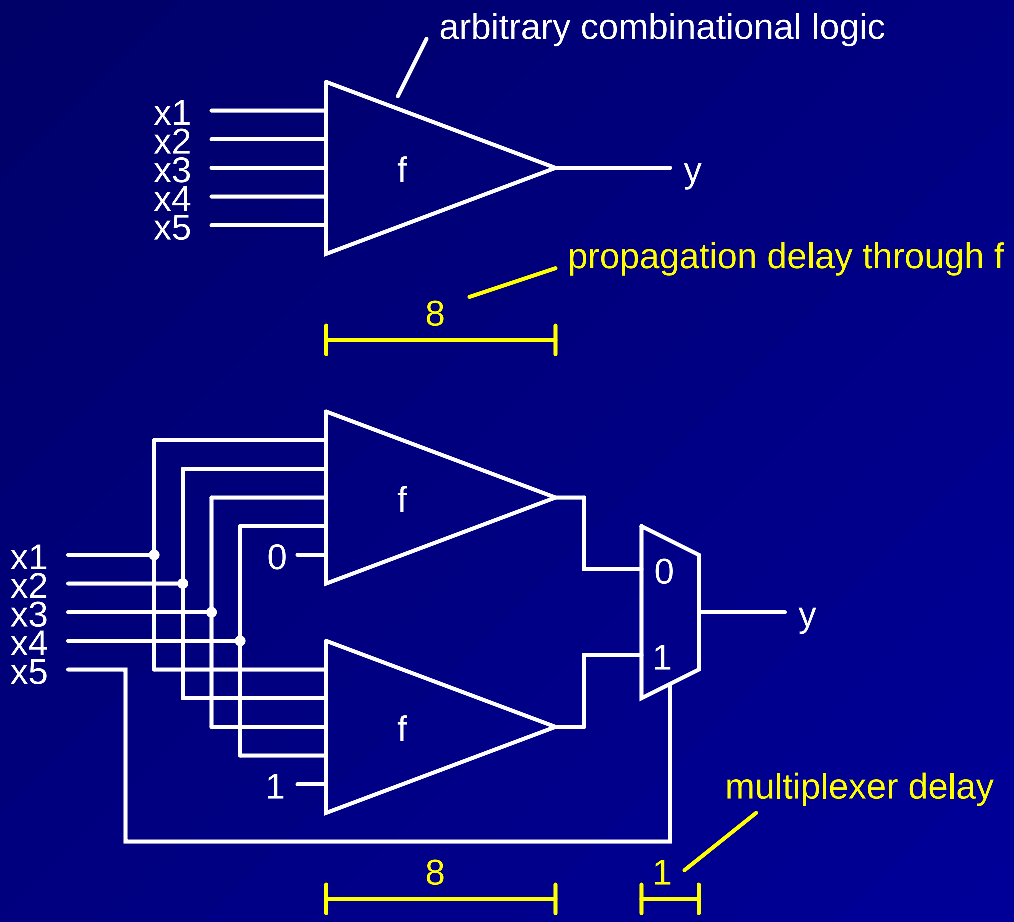
Stephen A. Edwards

Department of Computer Science,

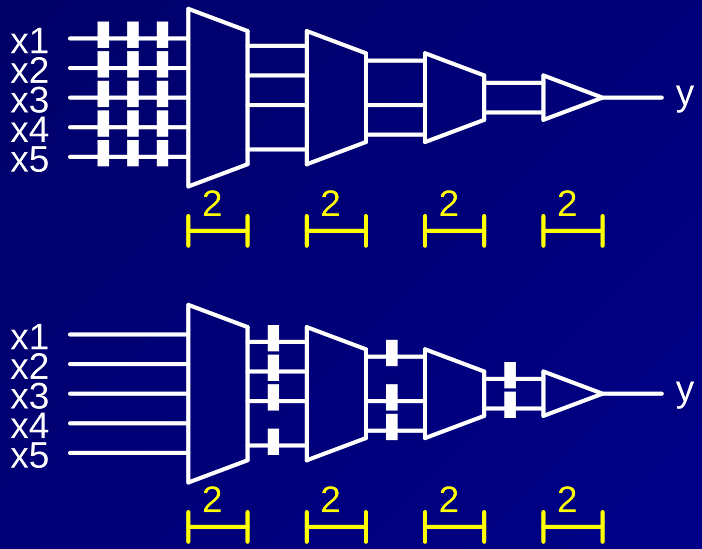
Columbia University

{soviani,tardieu,sedwards}@cs.columbia.edu

Shannon transform - review

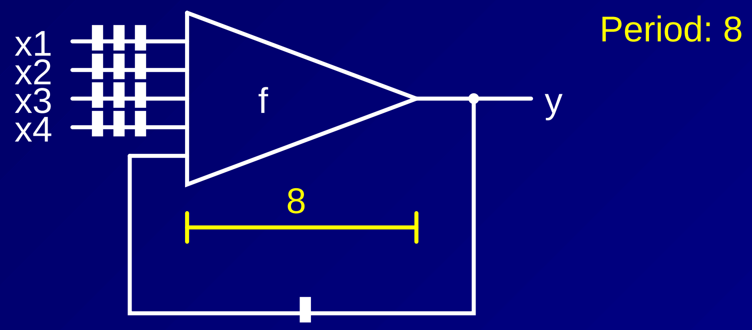


Retiming - review



Improves performance by re-distributing the registers

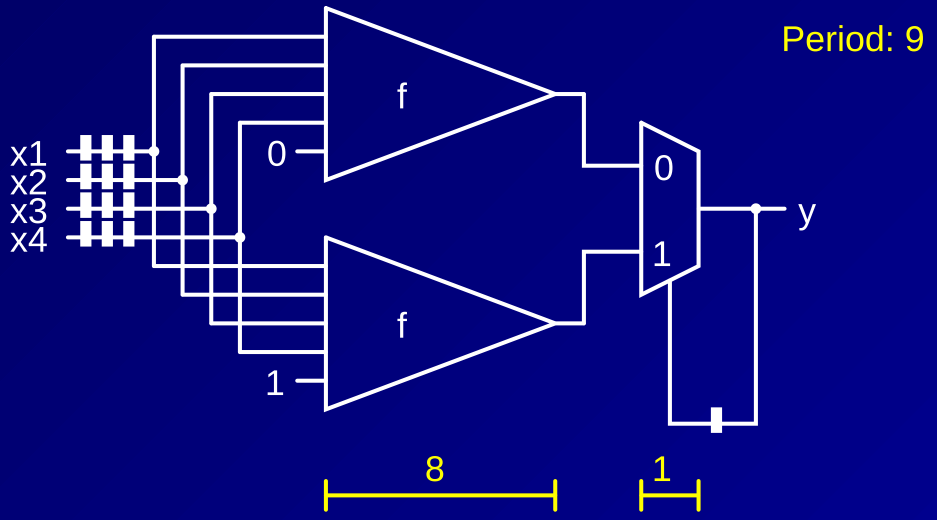
Motivation: speed up this sample



Observations:

- Retiming can not improve performance because of the loop
- Shannon seems useless, as all inputs arrive at the same time (0)

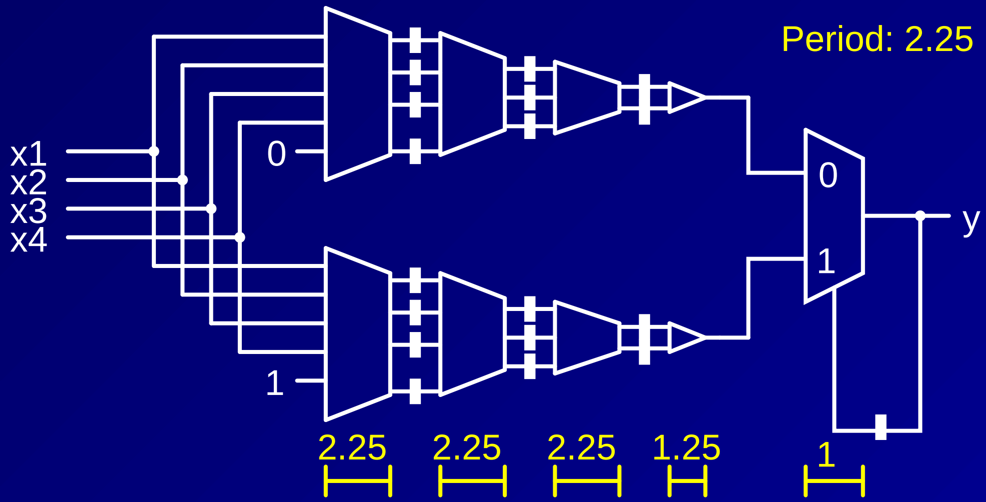
Sample after Shannon transform



Observations:

- the performance is worse
- the loop is much smaller

Sample after Shannon and retiming



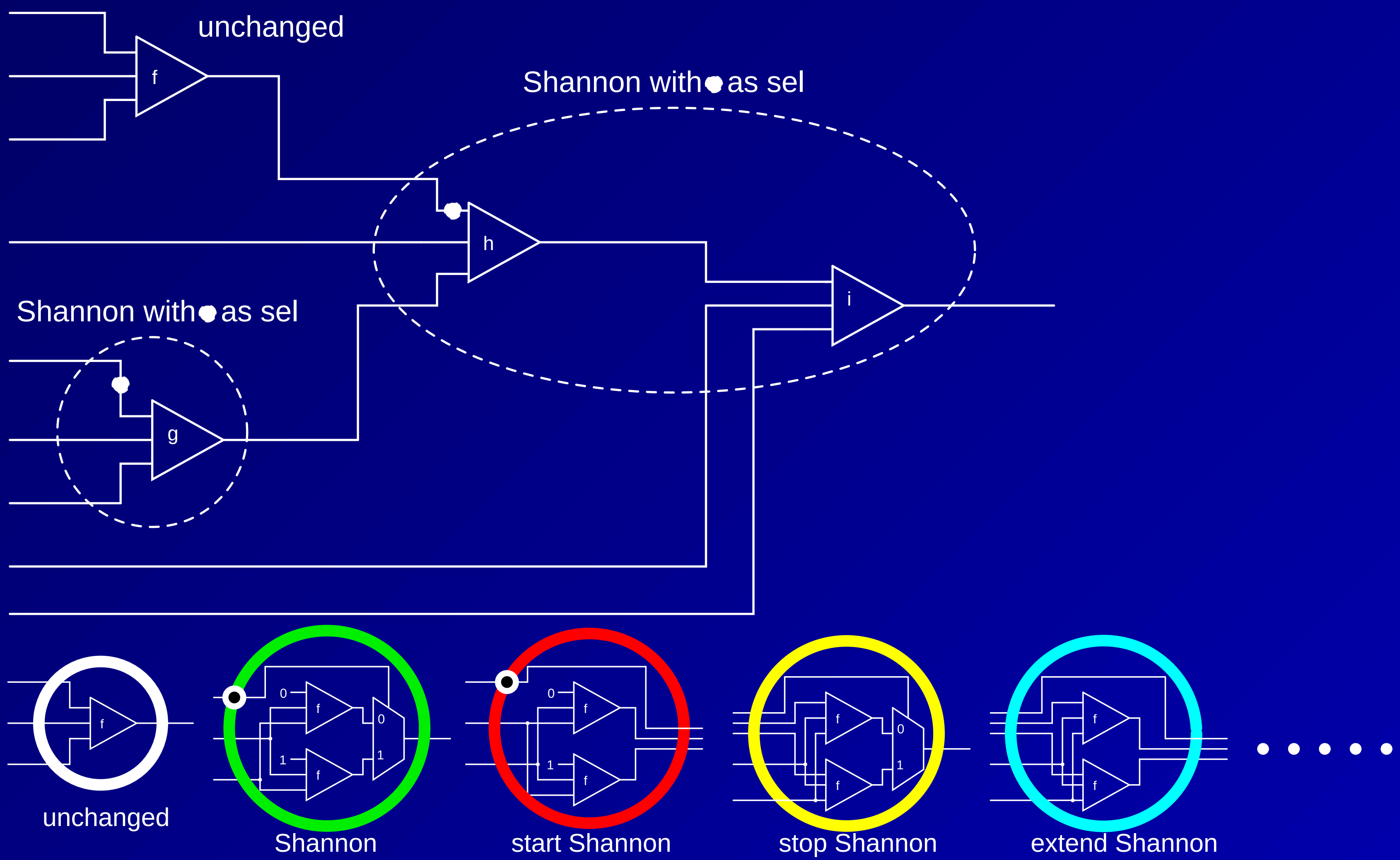
Shannon transform(s) and retiming: a huge design space

- finding the best combination is not trivial
- we want a systematic way to explore it

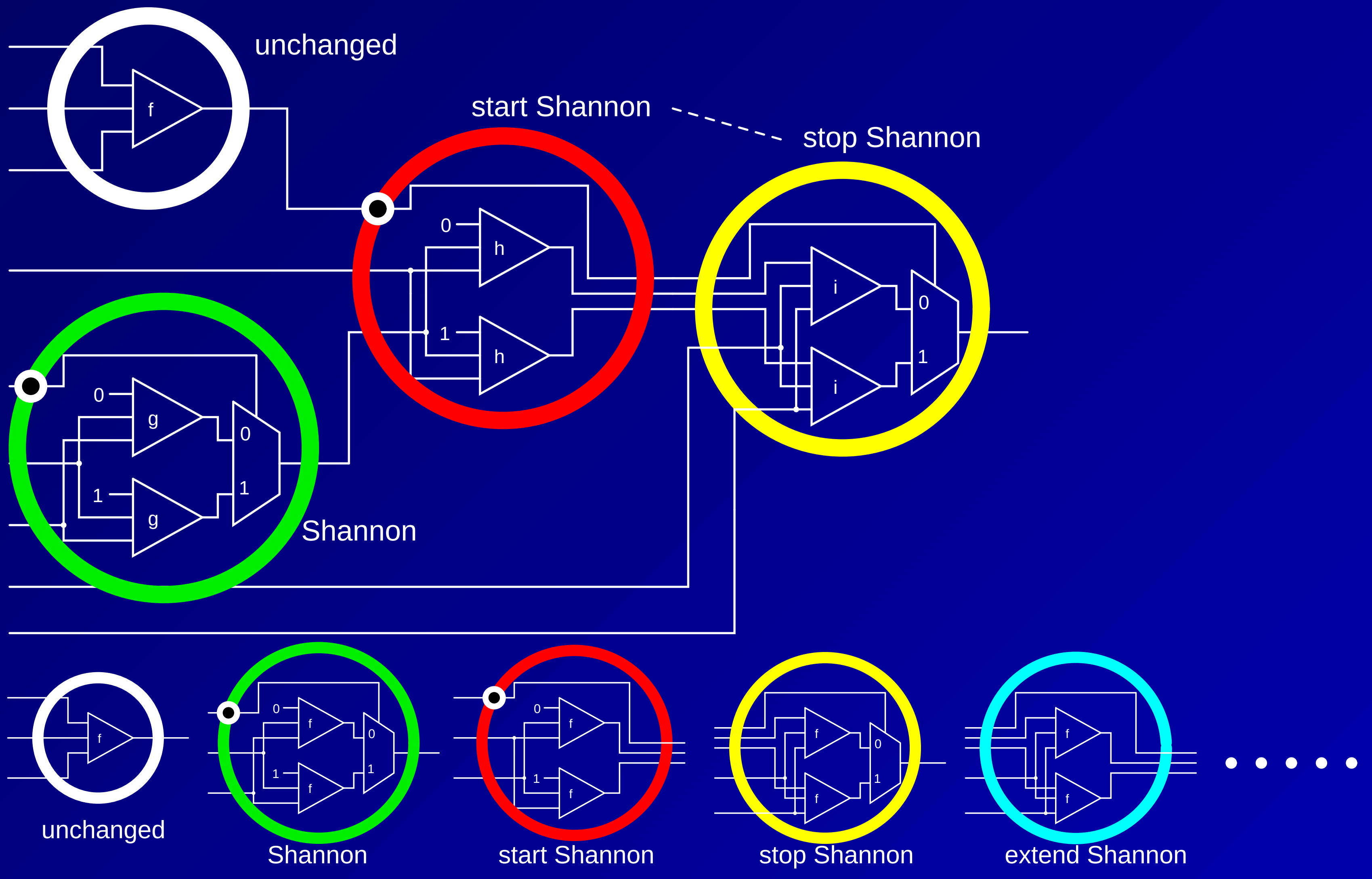
Presentation outline. Contributions

- new model for describing complex combinations of Shannon decompositions
- feasible arrival time (*fat*) sets : generalization of arrival times for Shannon-encoded signals
- algorithm: systematic exploration of the Shannon / retiming design space

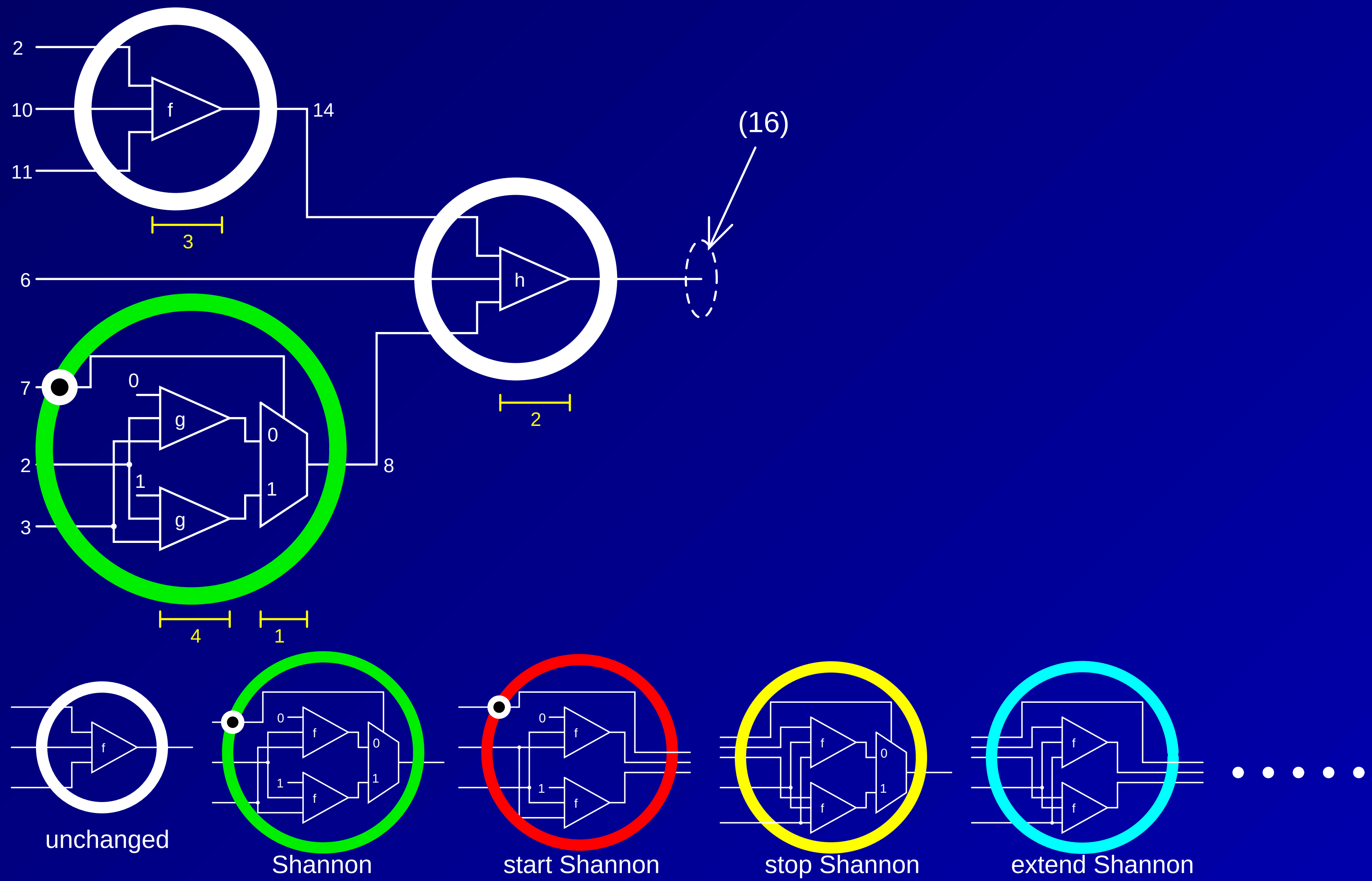
Modeling Shannon decompositions



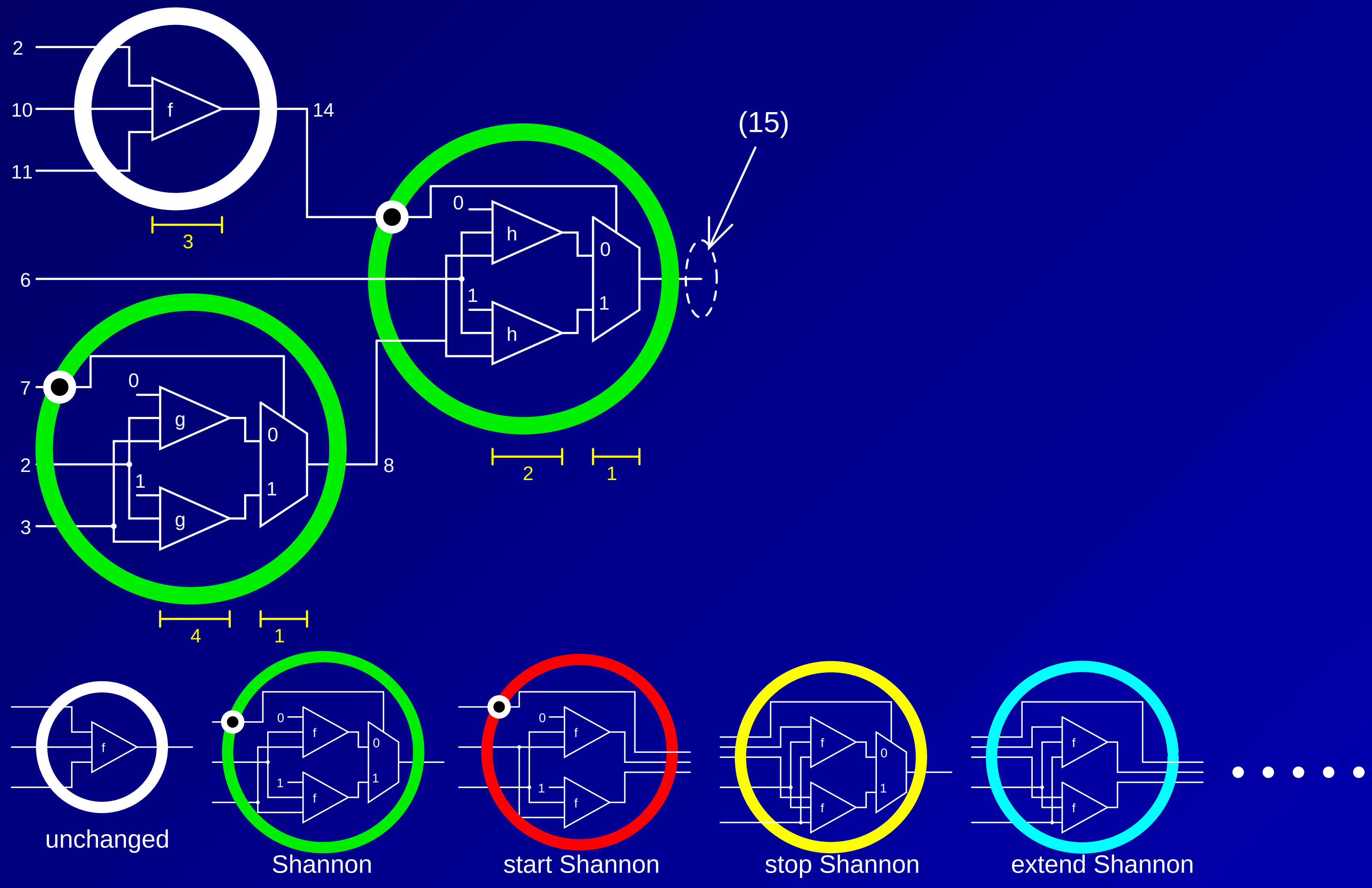
Modeling Shannon decompositions



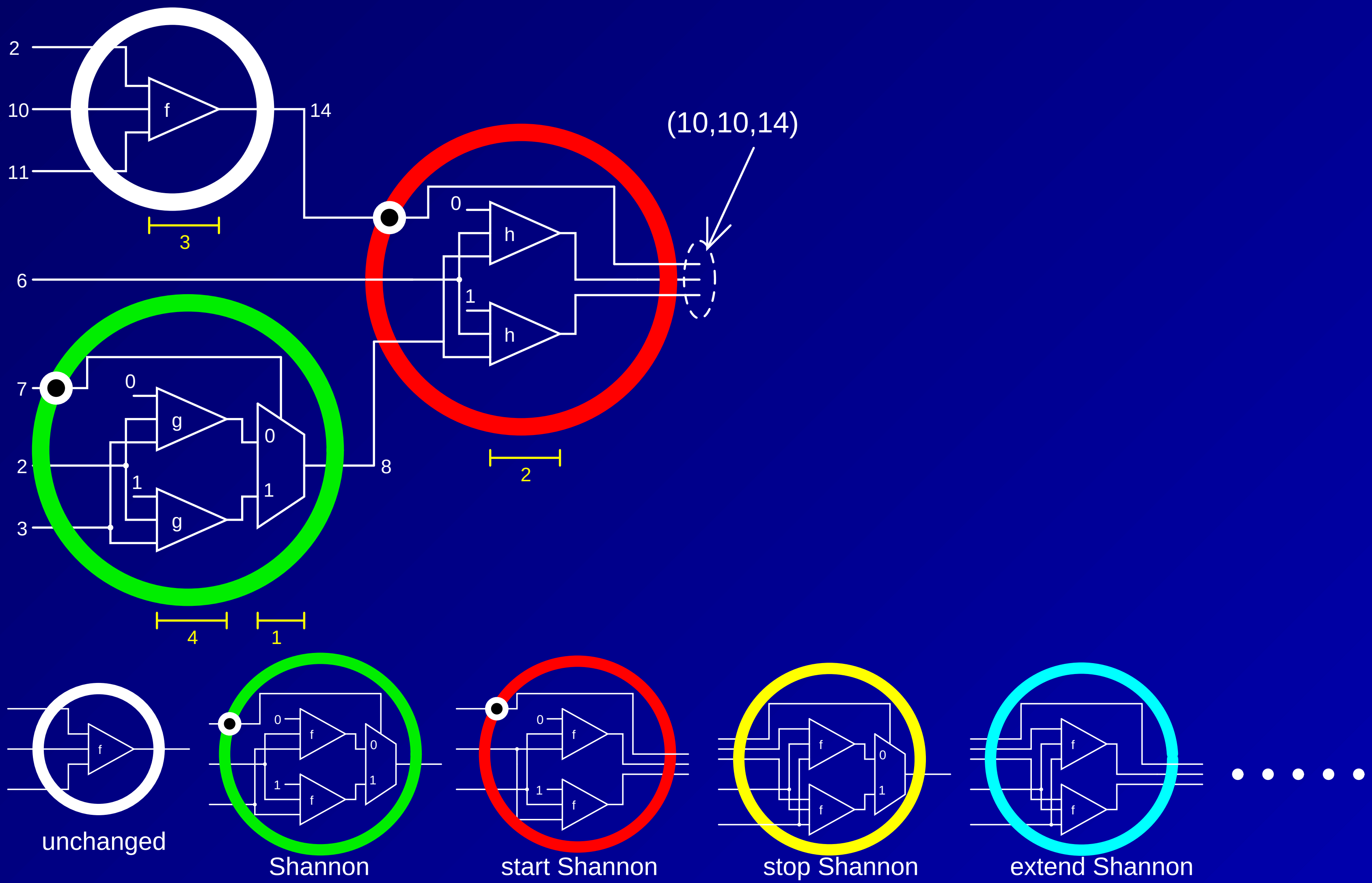
Exploring variants for h ...



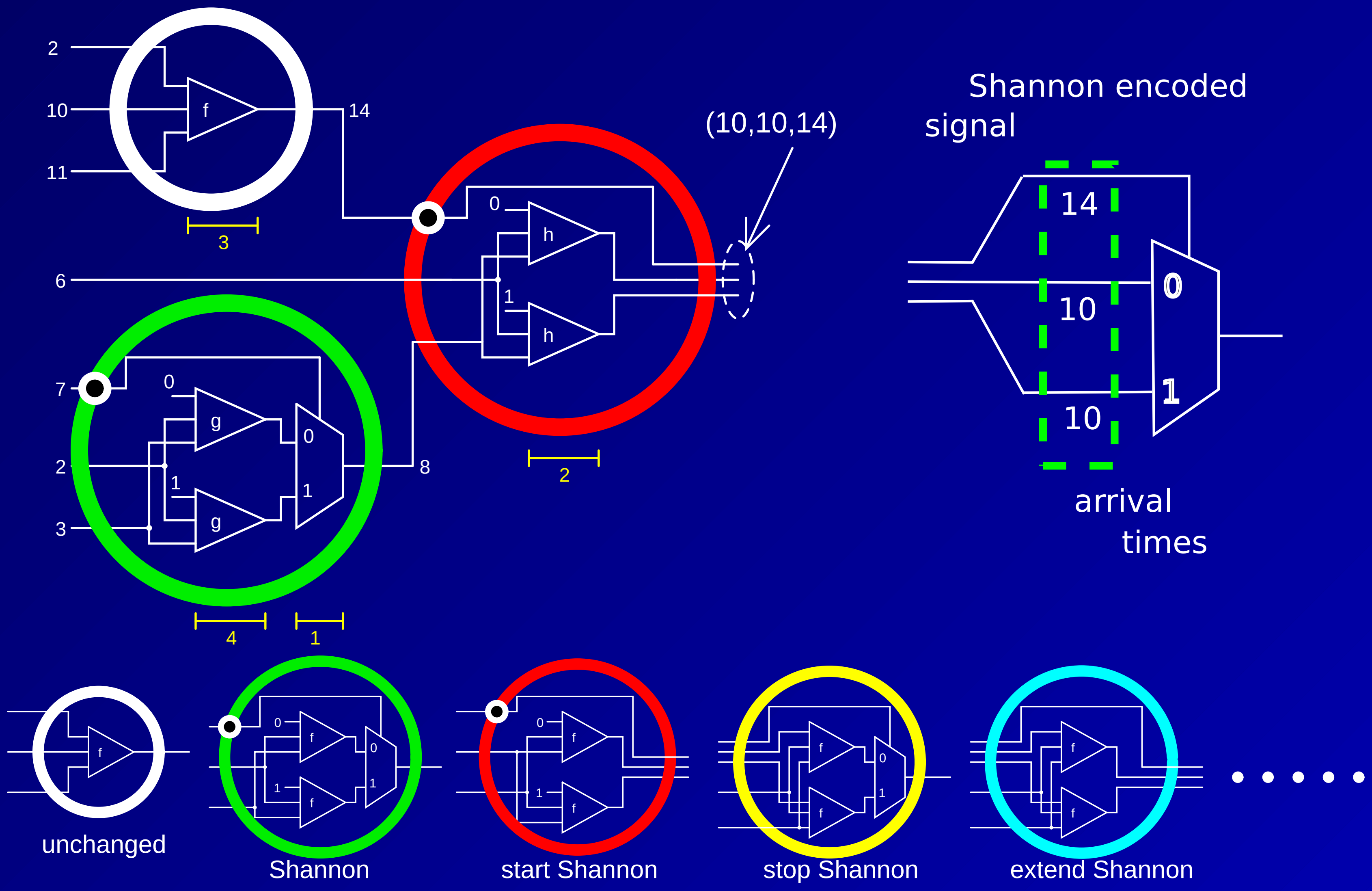
Exploring variants for h ...



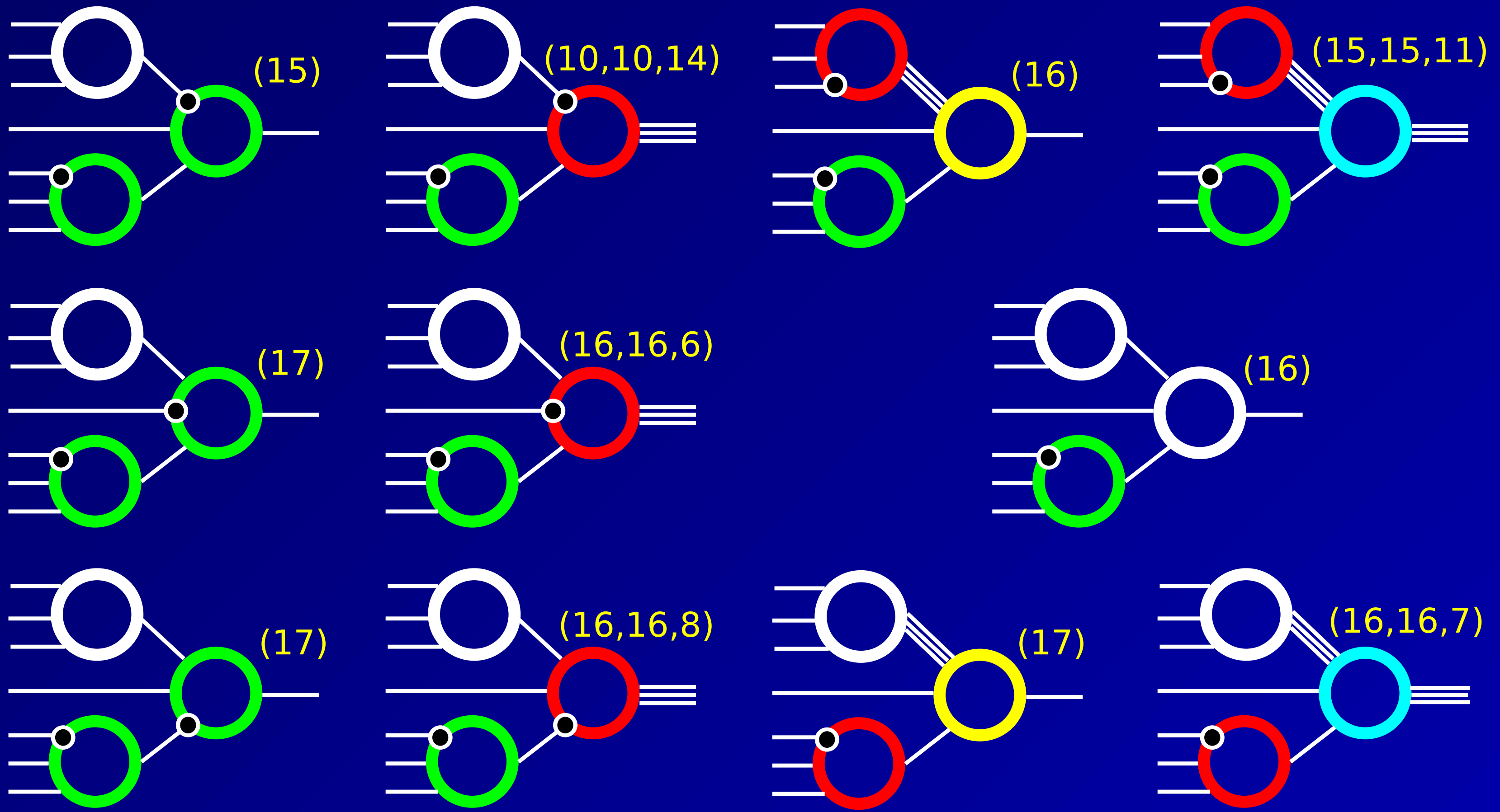
Exploring variants for h ...



Exploring variants for h ...



What combination is faster ?



Problem : several combinations have “triple” outputs

All feasible arrival times for h

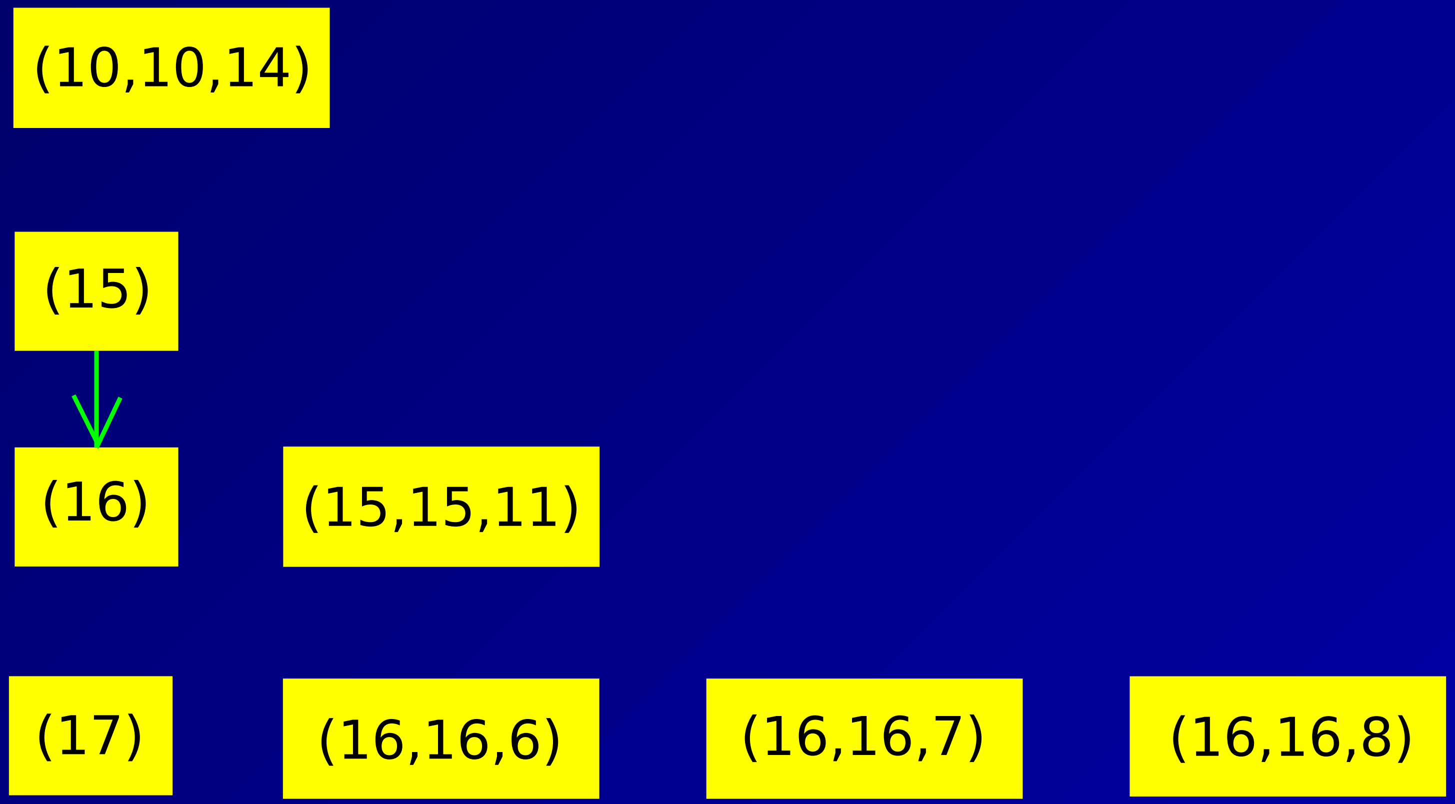
(10,10,14)

(15)

(16) (15,15,11)

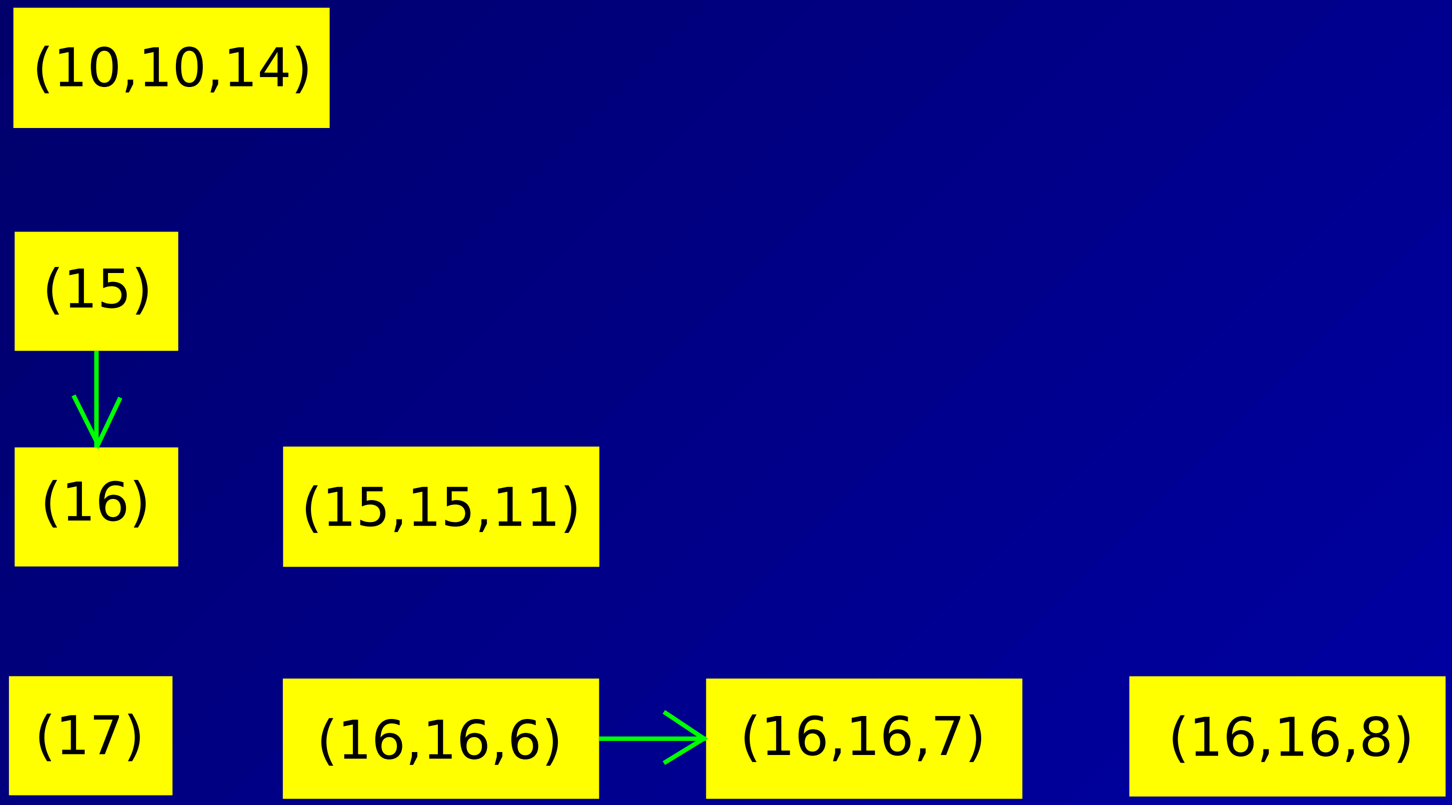
(17) (16,16,6) (16,16,7) (16,16,8)

Pruning *fat*(h) ...



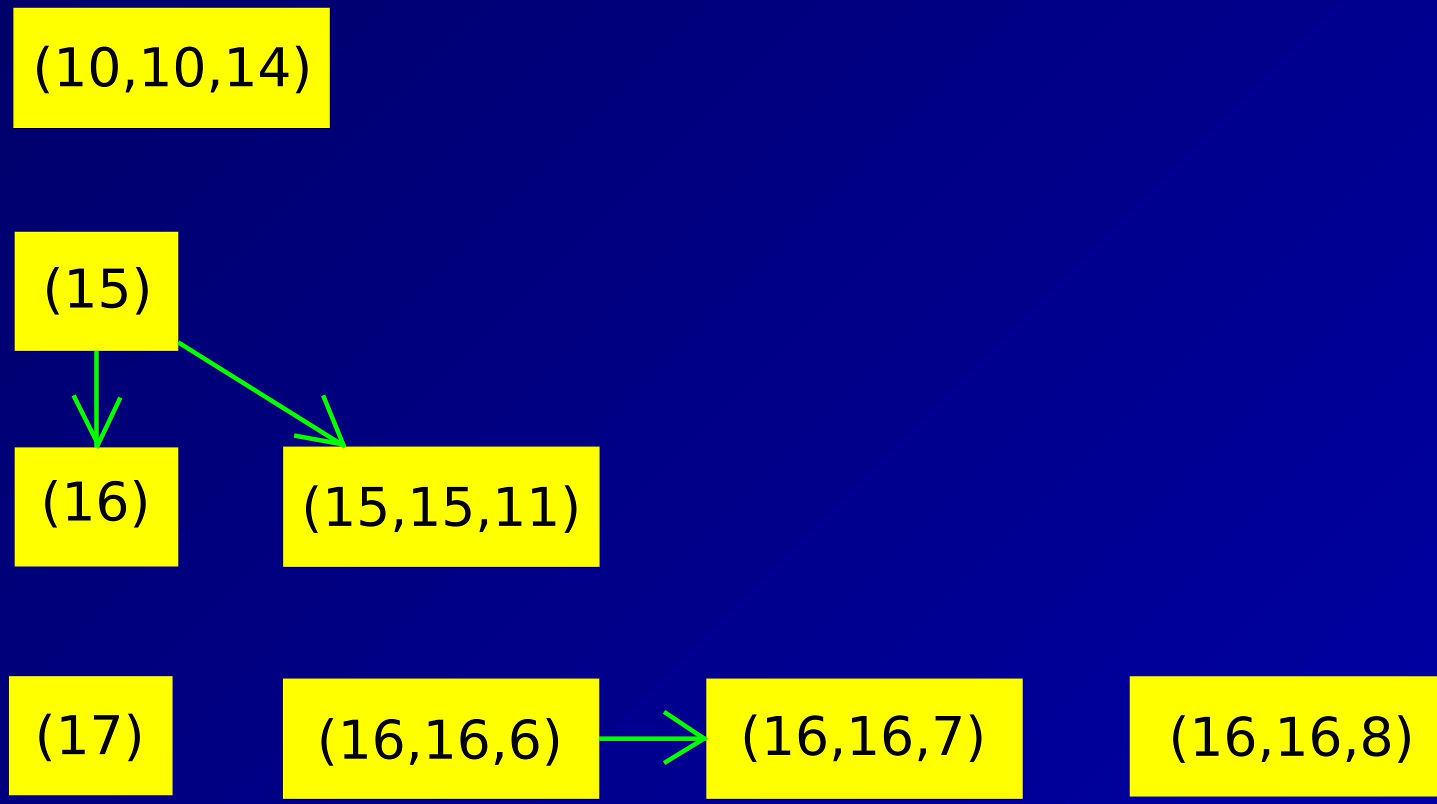
$(15) \preceq (16)$: (16) can be safely ignored

Pruning *fat*(h) ...



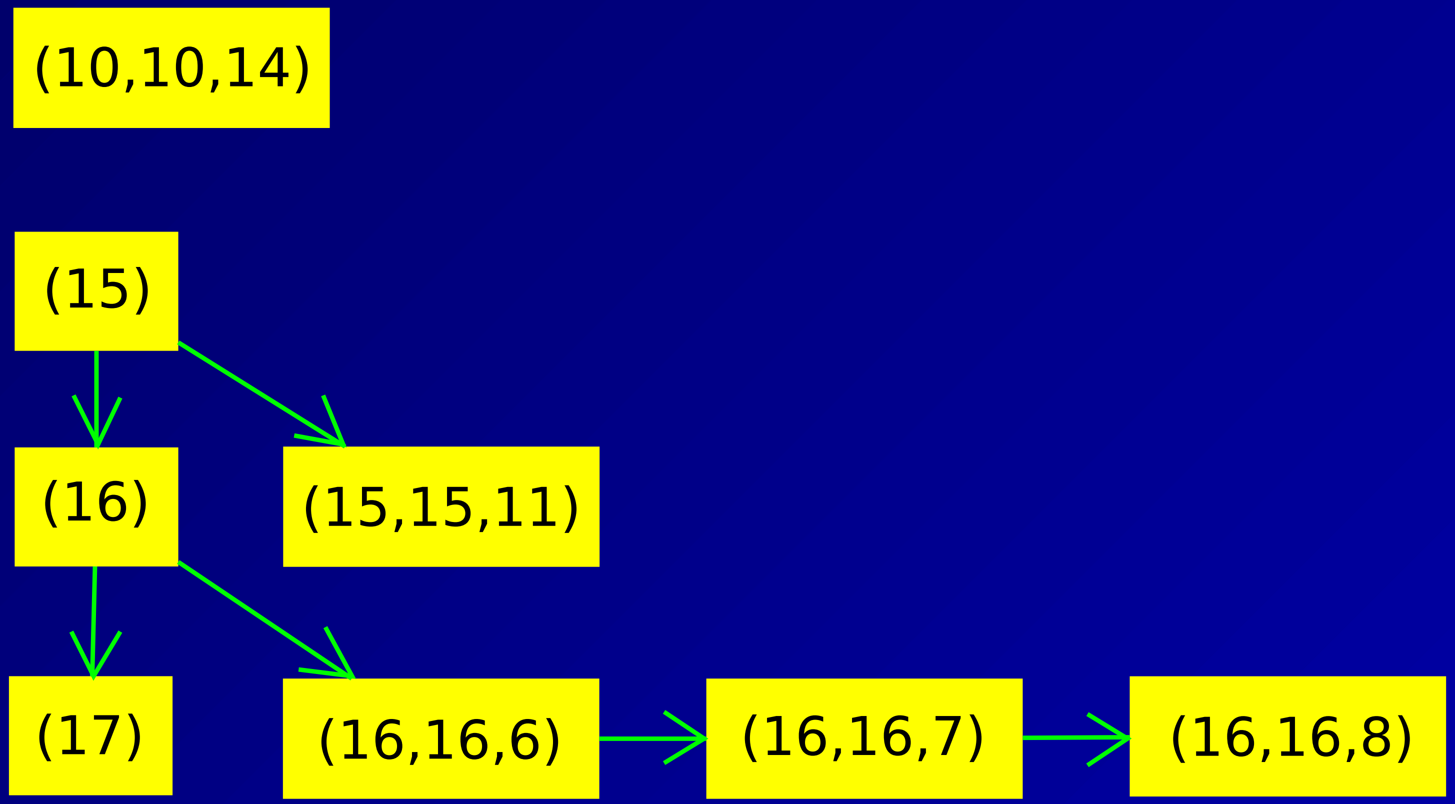
$$(16, 16, 6) \preceq (16, 16, 7)$$

Pruning ***fat***(h) ...



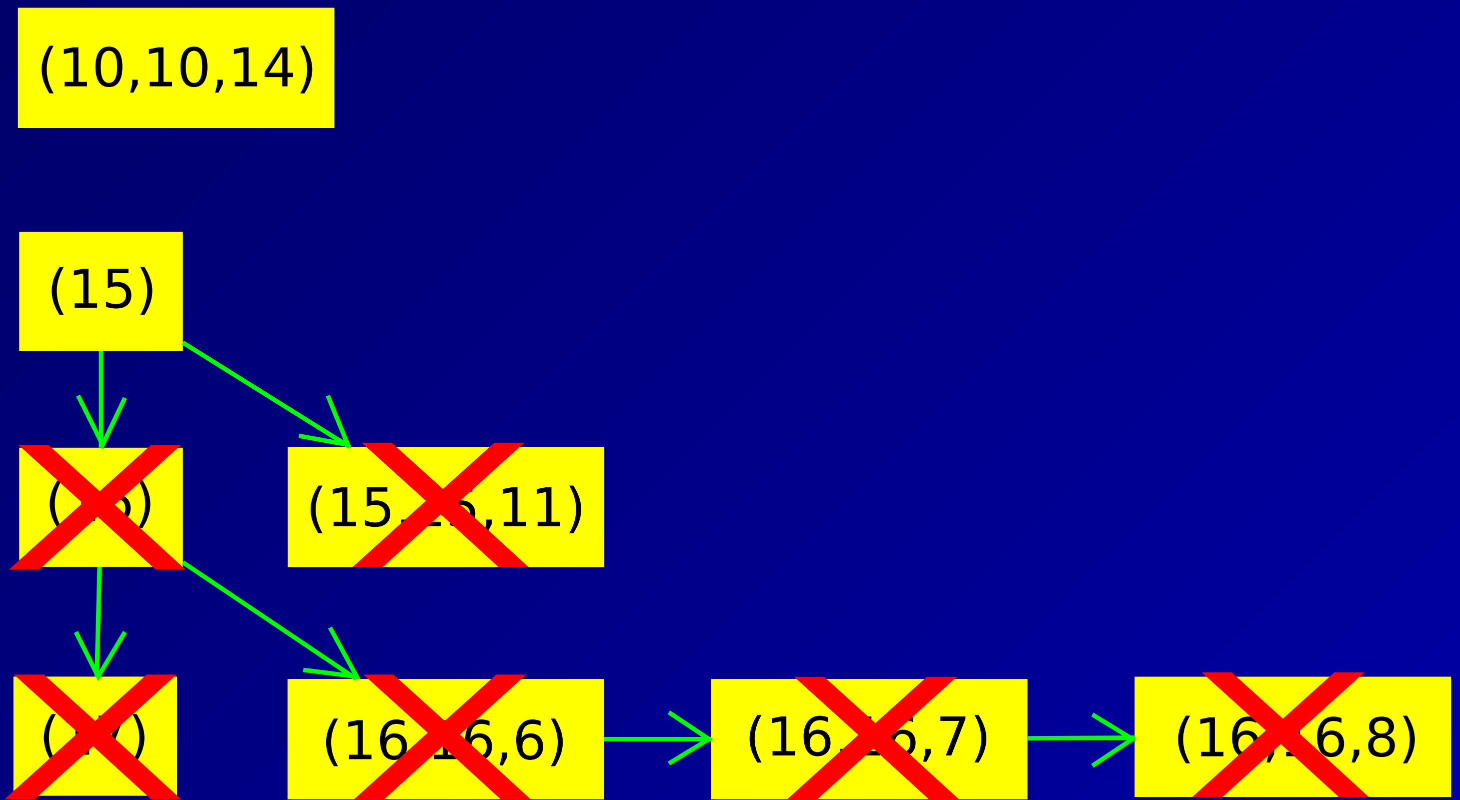
$$(15) \preceq (15, 15, 11)$$

fat(h) poset

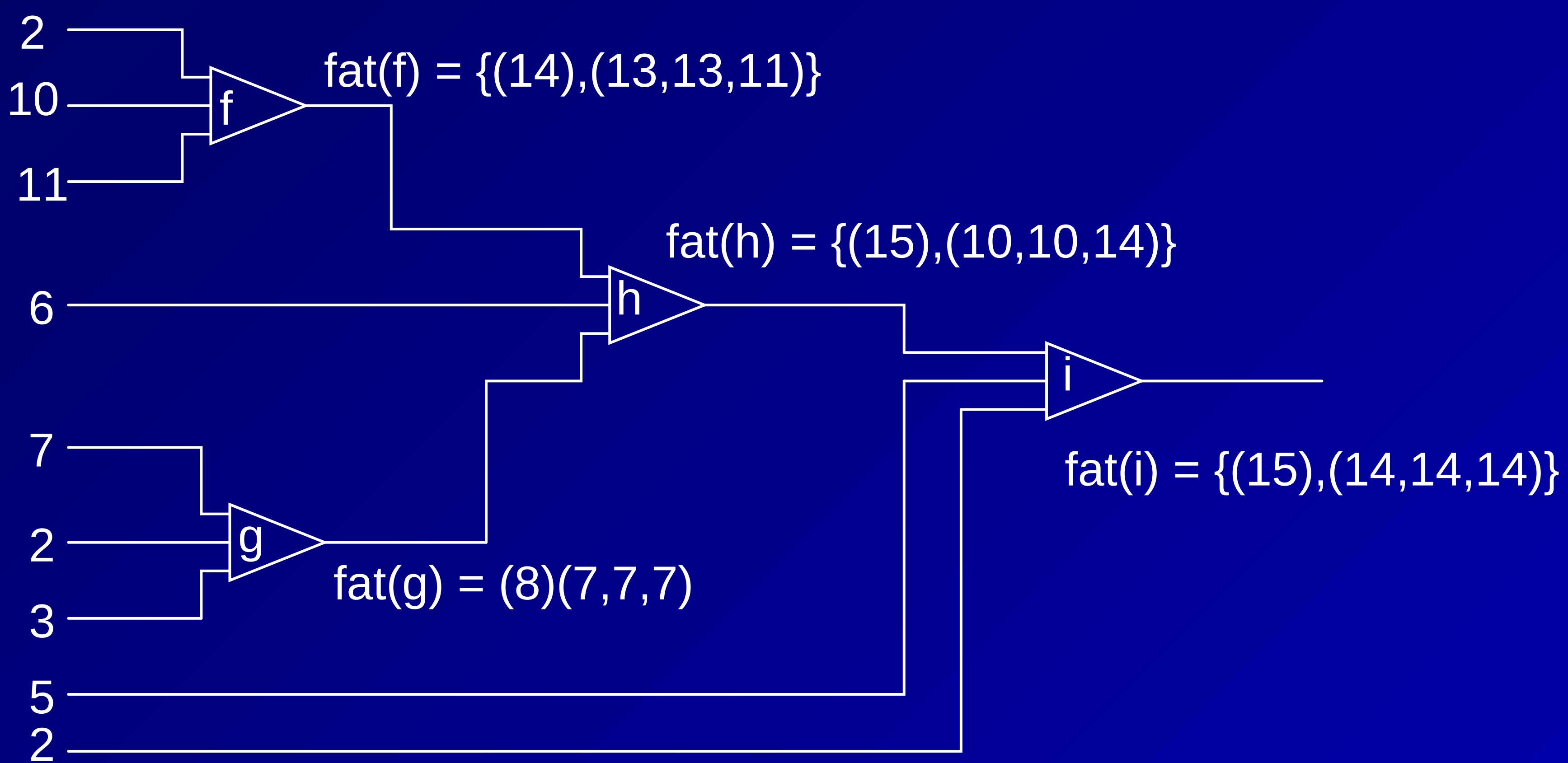


We want to keep only the maximal elements

The pruned *fat*(h) set

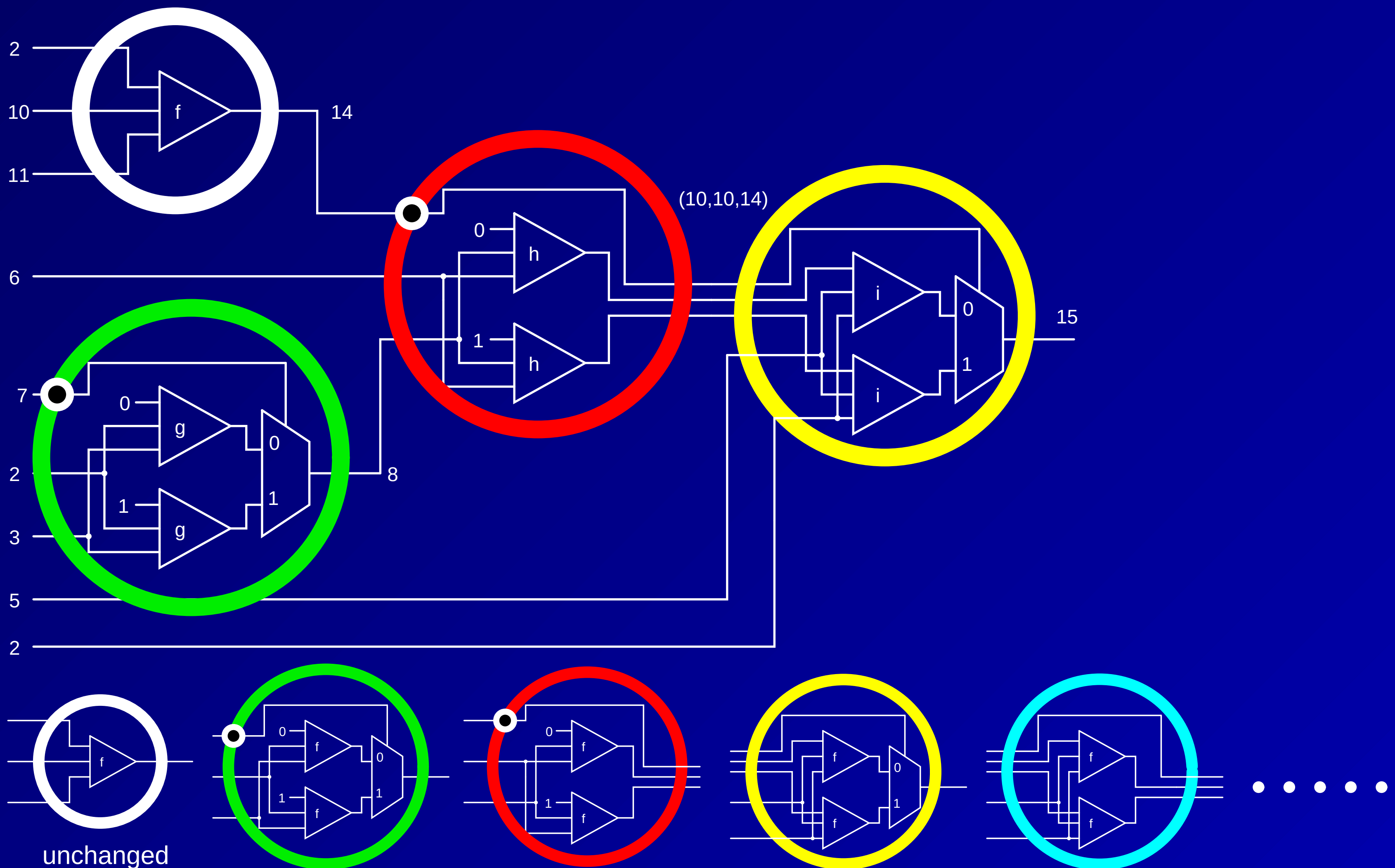


The fat sets



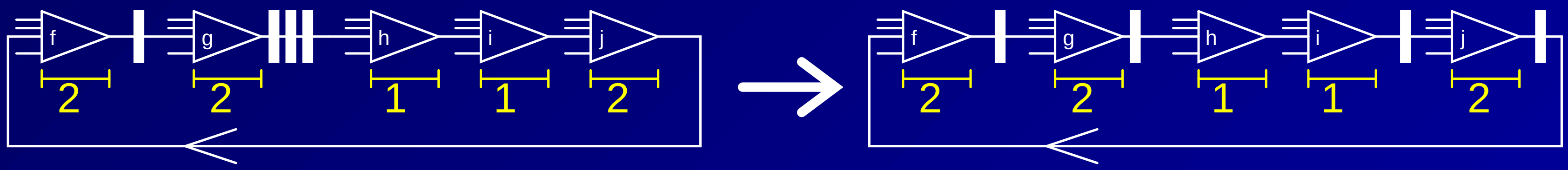
$$\textit{fat}(h) \quad = \quad \text{combine}(d(h), \{\textit{fat}(f), \{(6)\}, \textit{fat}(g)\})$$

Best delay for combinational logic



unchanged Shannon start Shannon stop Shannon extend Shannon

Retiming limitation for one cycle



D : total loop combinational delay $(2+2+1+1+2=8)$

R : number of registers on the loop $(1+3=4)$

$$\begin{aligned} \frac{D}{R} &\leq c \\ D &\leq c \cdot R \\ D + (-c) \cdot R &\leq 0 \end{aligned}$$

Assign weight $(-c)$ to registers:

Period c is feasible $\Leftrightarrow$ the cycle has negative weight

The fundamental limit of retiming

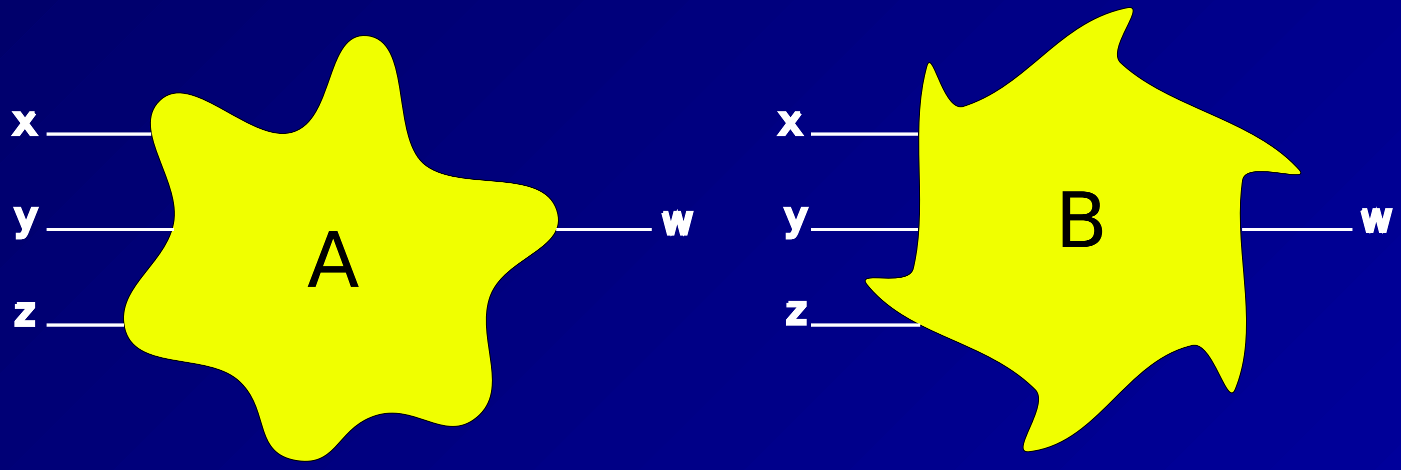
For a given sequential network,

Period c is feasible if ALL^* cycles are negative.

f.l.ret$(S) = \min c$ such that all cycles in S are negative

(*) including an artificial external arc between POs and PIs

Retiming and restructuring



Is A faster than B ?

max_comb_delay$(A) <$ *max_comb_delay(B)* : WRONG

f.l.ret$(A) <$ *f.l.ret*(B)

The Bellman-Ford algorithm

Bellman-Ford detects positive cycles in polynomial time

$$\left\{ \begin{array}{l} \text{Period c is feasible} \\ \text{ALL cycles are negative} \\ \text{Bellman-Ford converges to a FIX POINT} \end{array} \right\}$$

Generalization: Instead of arrival times (real numbers, e.g. 3) we use *fat* sets (e.g. $\{(15),(10,10,14)\}$).

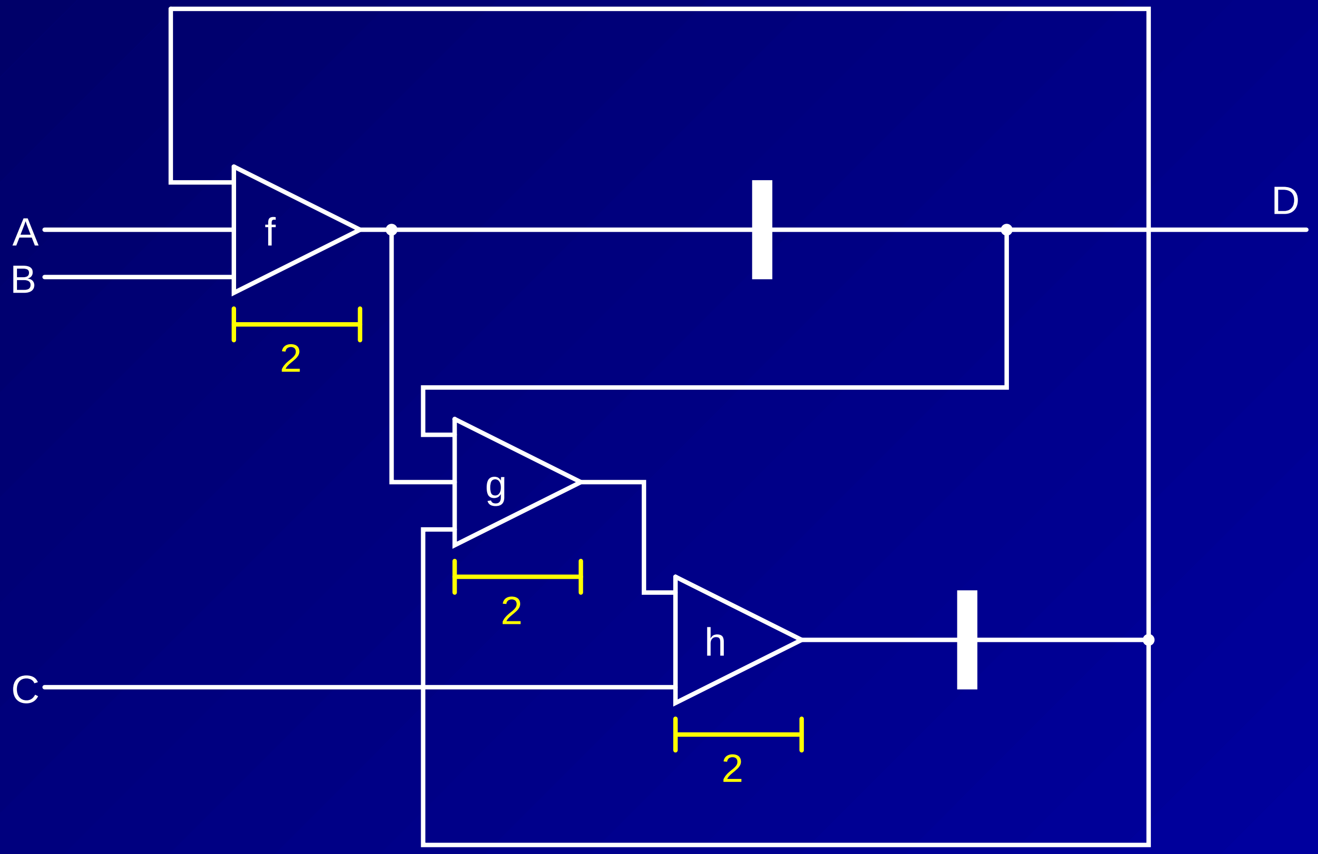
- $\mathit{at}(n) = d(n) + \max_{n' \in \text{fanins}(n)} \mathit{at}(n')$
- $\mathit{fat}(n) = \text{combine}(d(n), \{\mathit{fat}(n')\}_{n' \in \text{fanins}(n)})$

Algorithm outline

procedure SeqShannon(S, c)
    (converges, fix_point_fat) = Bellman-Ford (S, c)
    if not converges **then**
        return NOT_FEASIBLE
    ShannonTransform(S, c, fix_point_fat)
    Retime(S)
    return SUCCESS

- we can approximate the best period c by binary search

Sequential sample - original



period=9

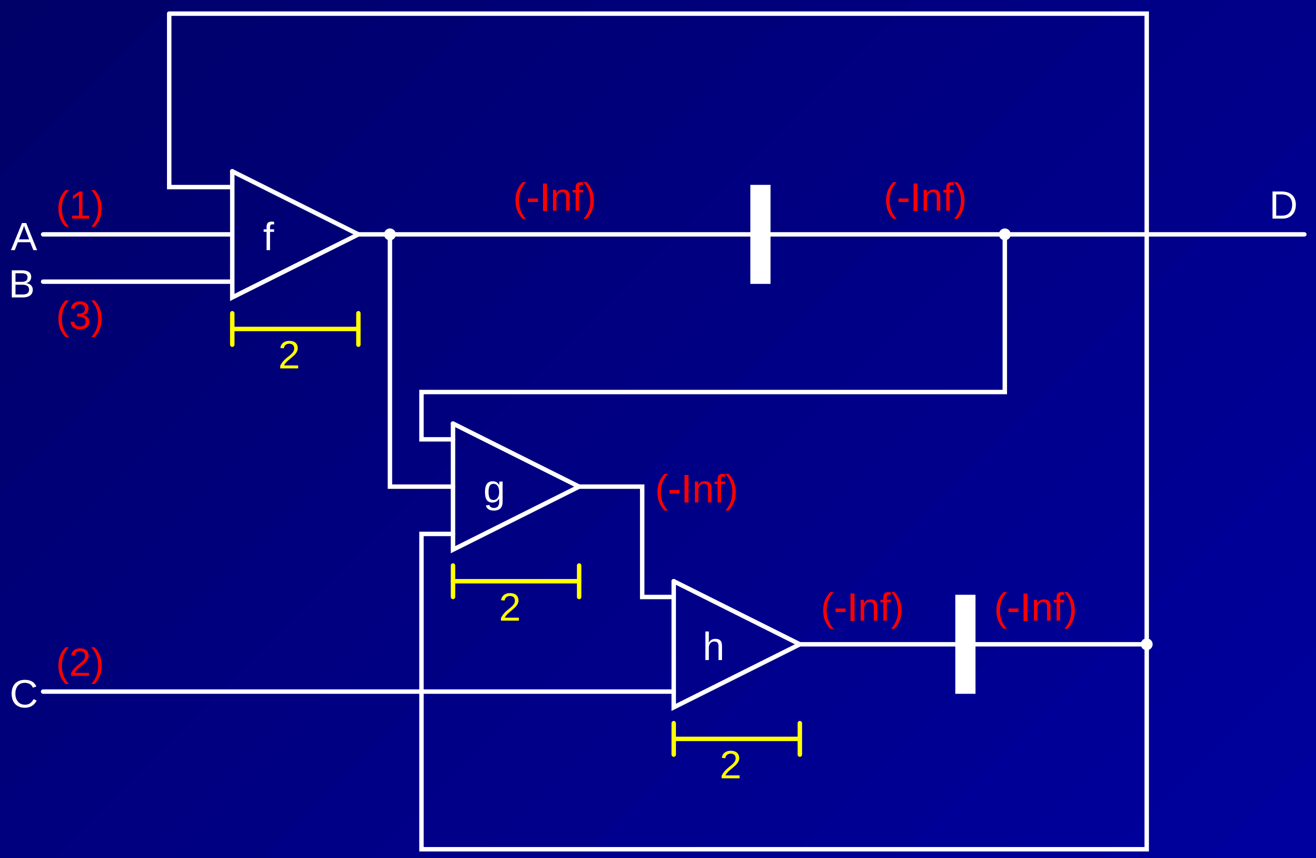
desired period : 3

multiplexer delay : 1

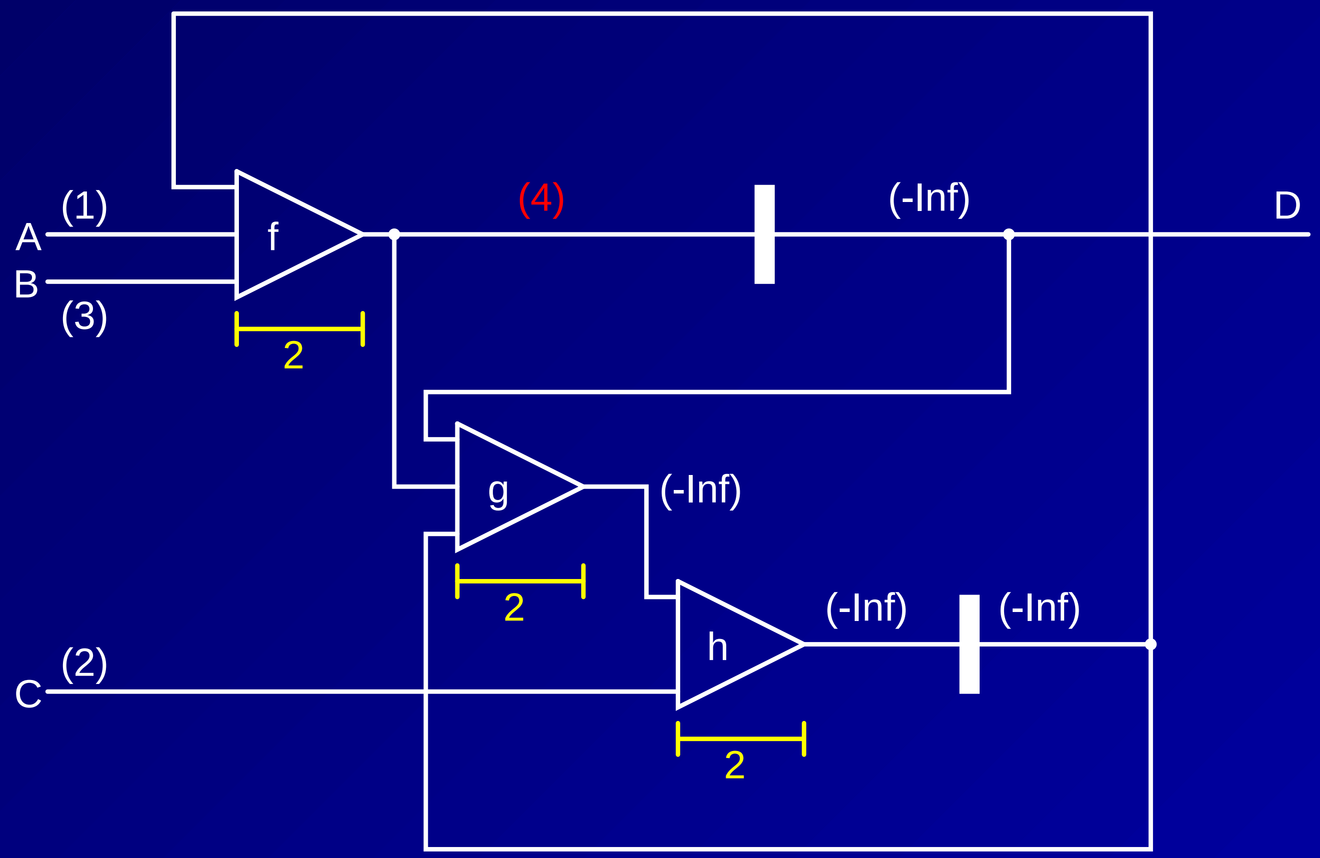
input arrival times: A=1 B=3 C=2

output required time(s): D=3

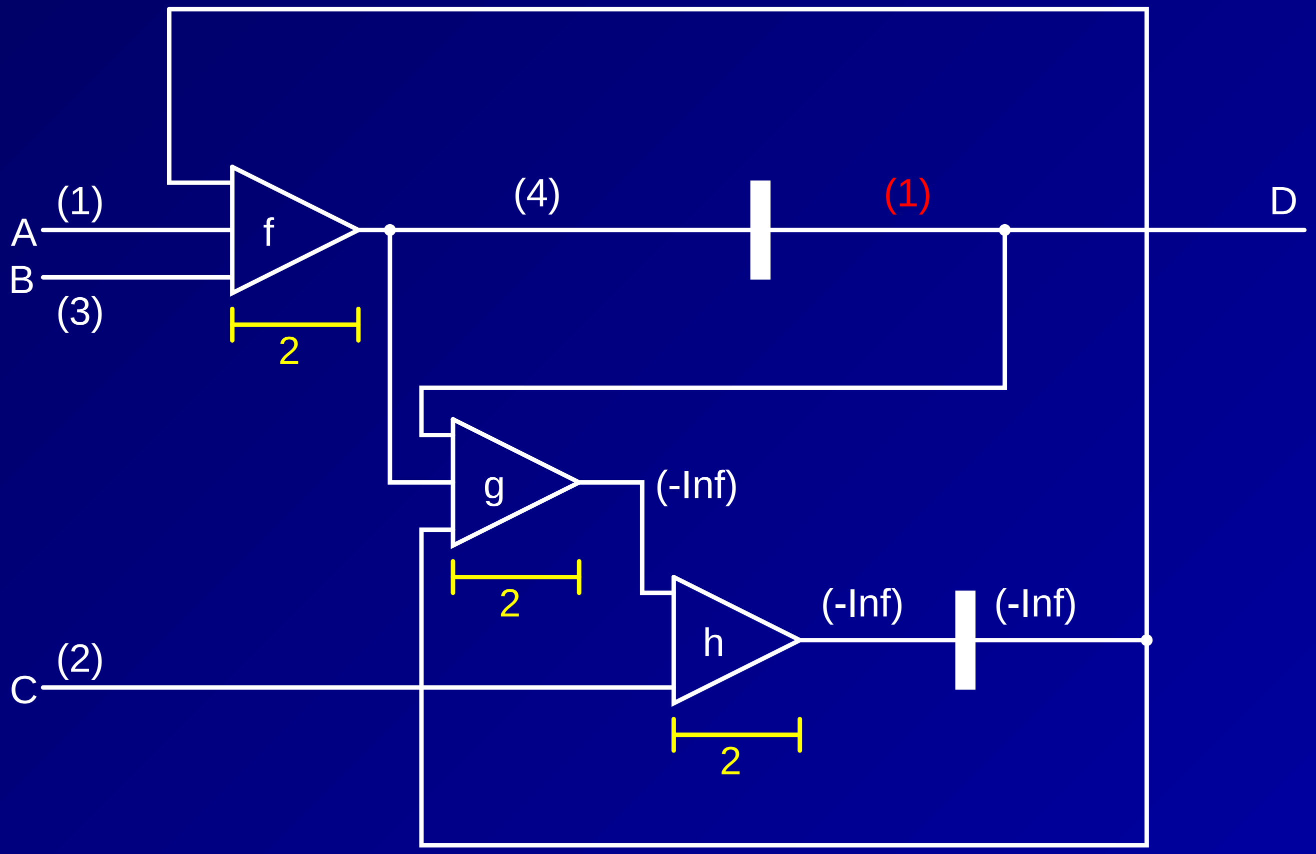
Bellman-Ford : initialization



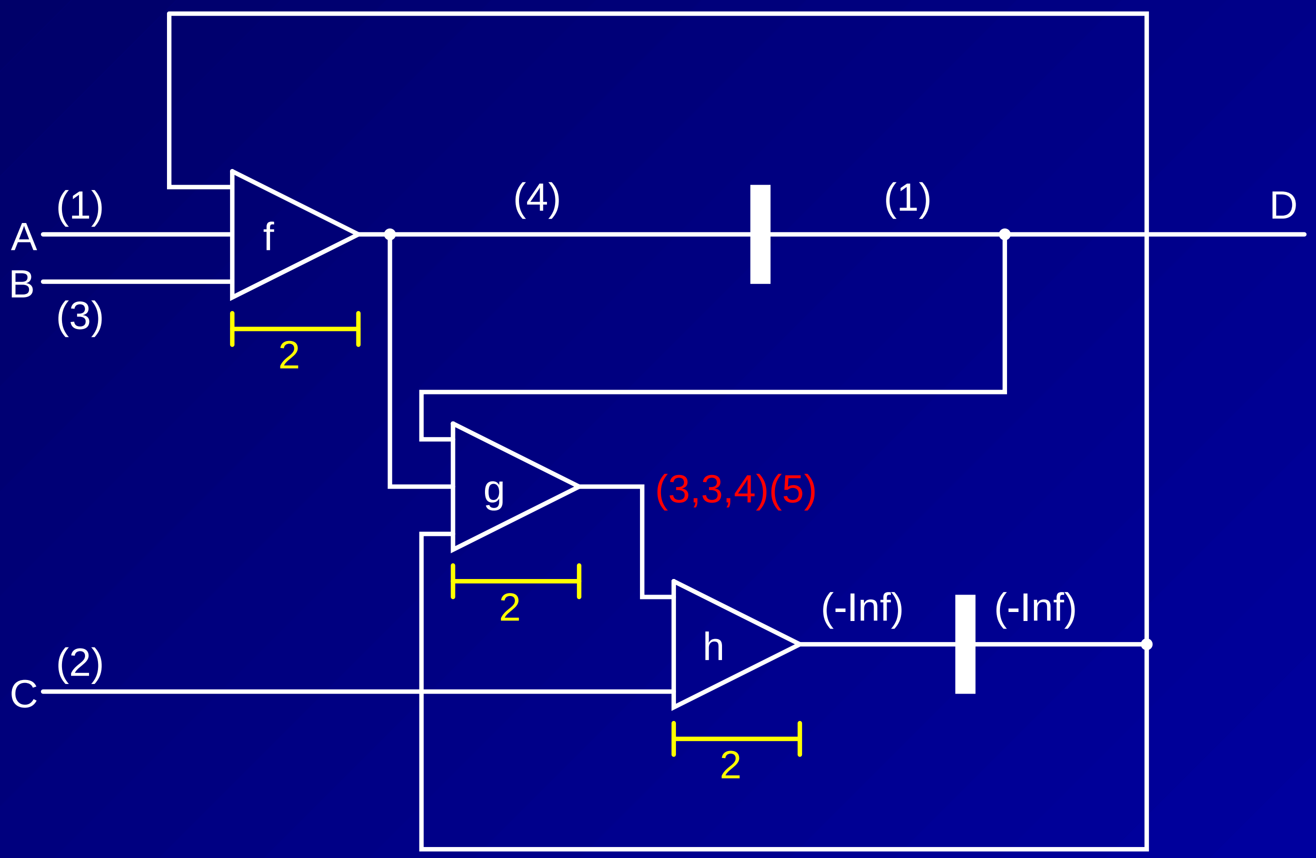
Bellman-Ford : starting relaxation



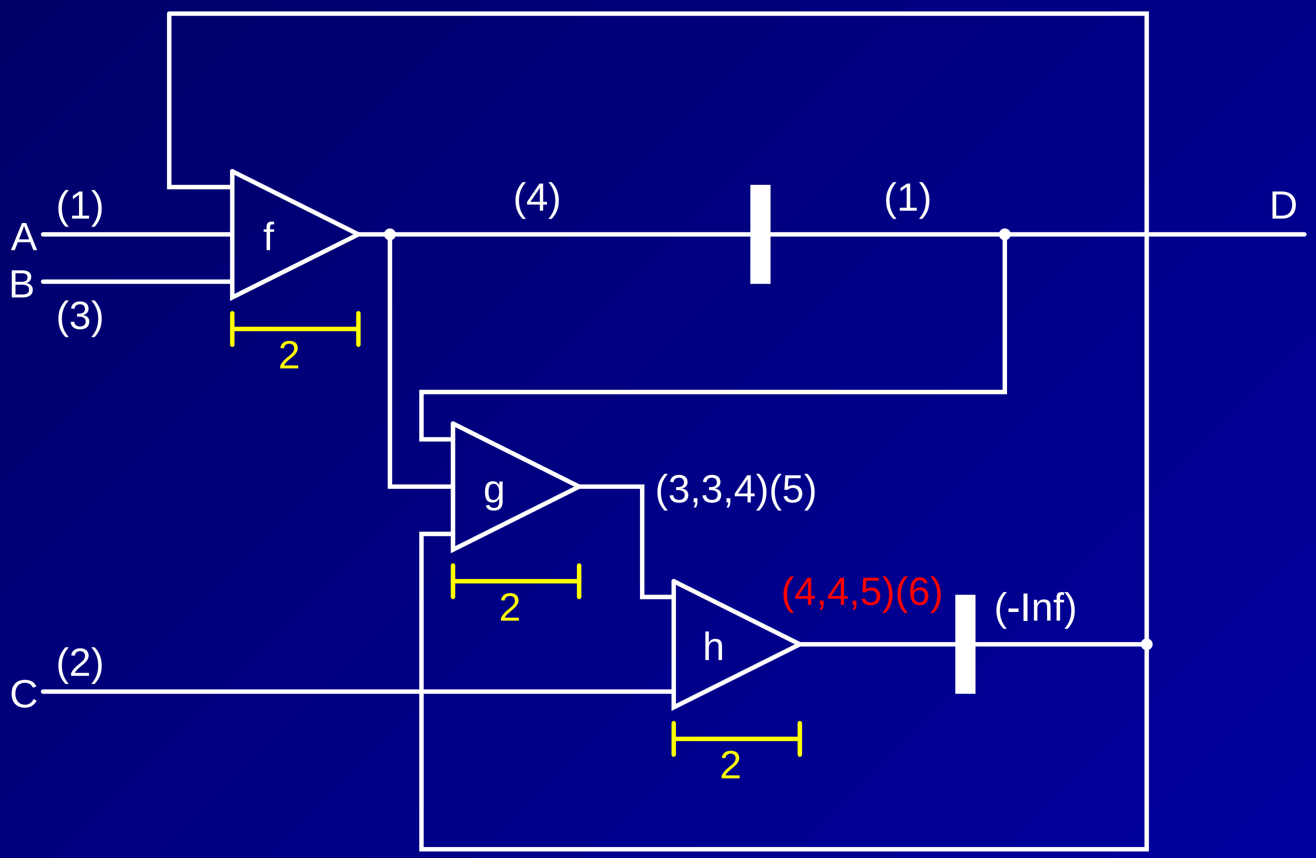
Bellman-Ford : relaxing ...



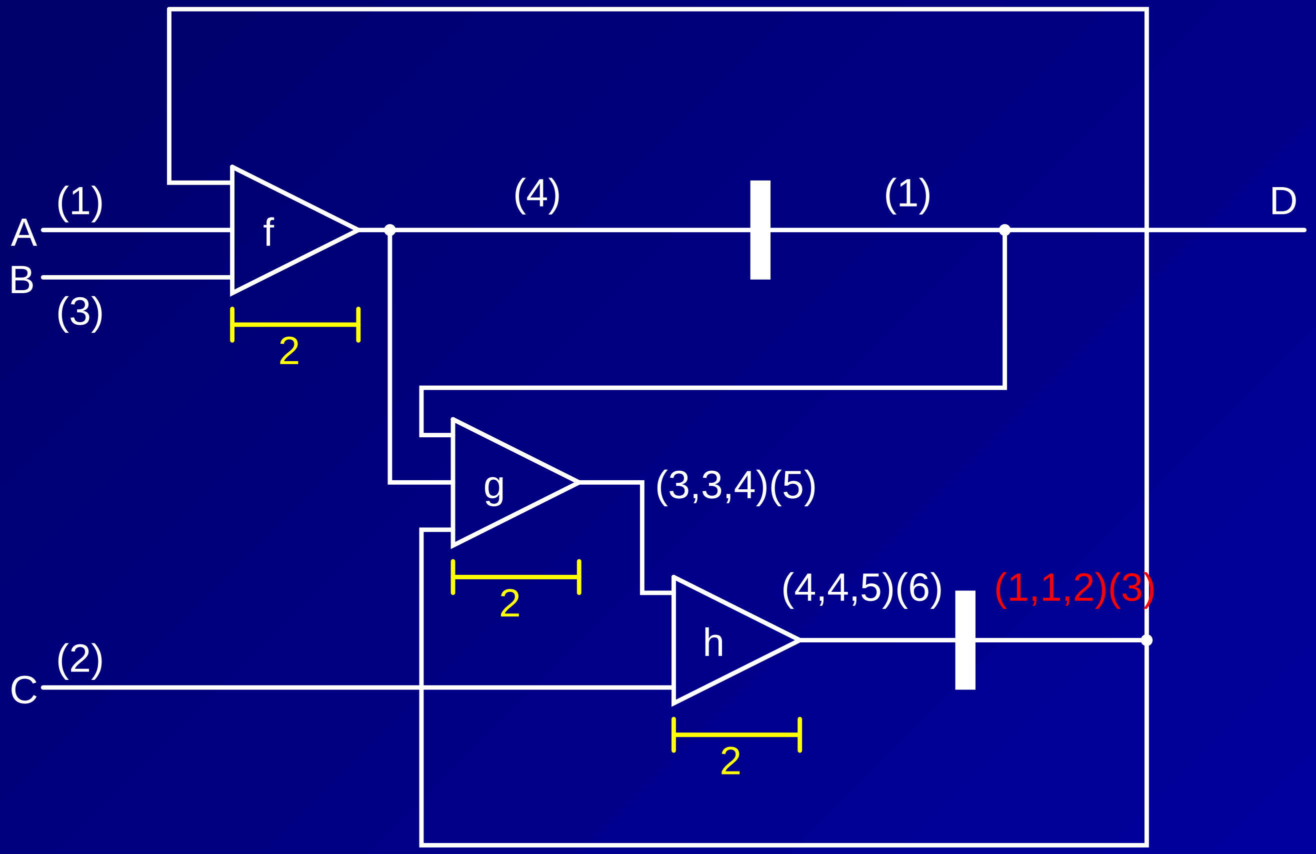
Bellman-Ford : relaxing ...



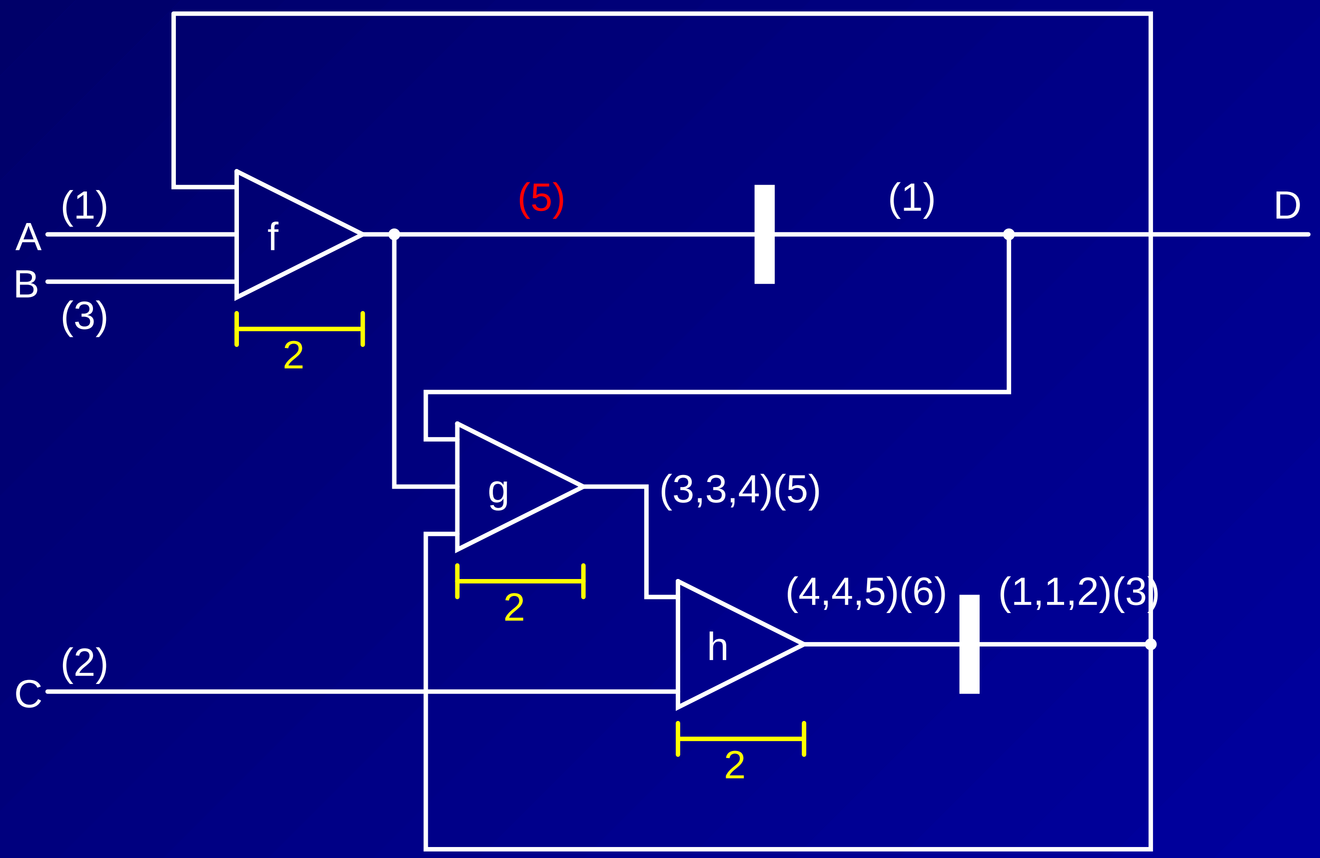
Bellman-Ford : relaxing ...



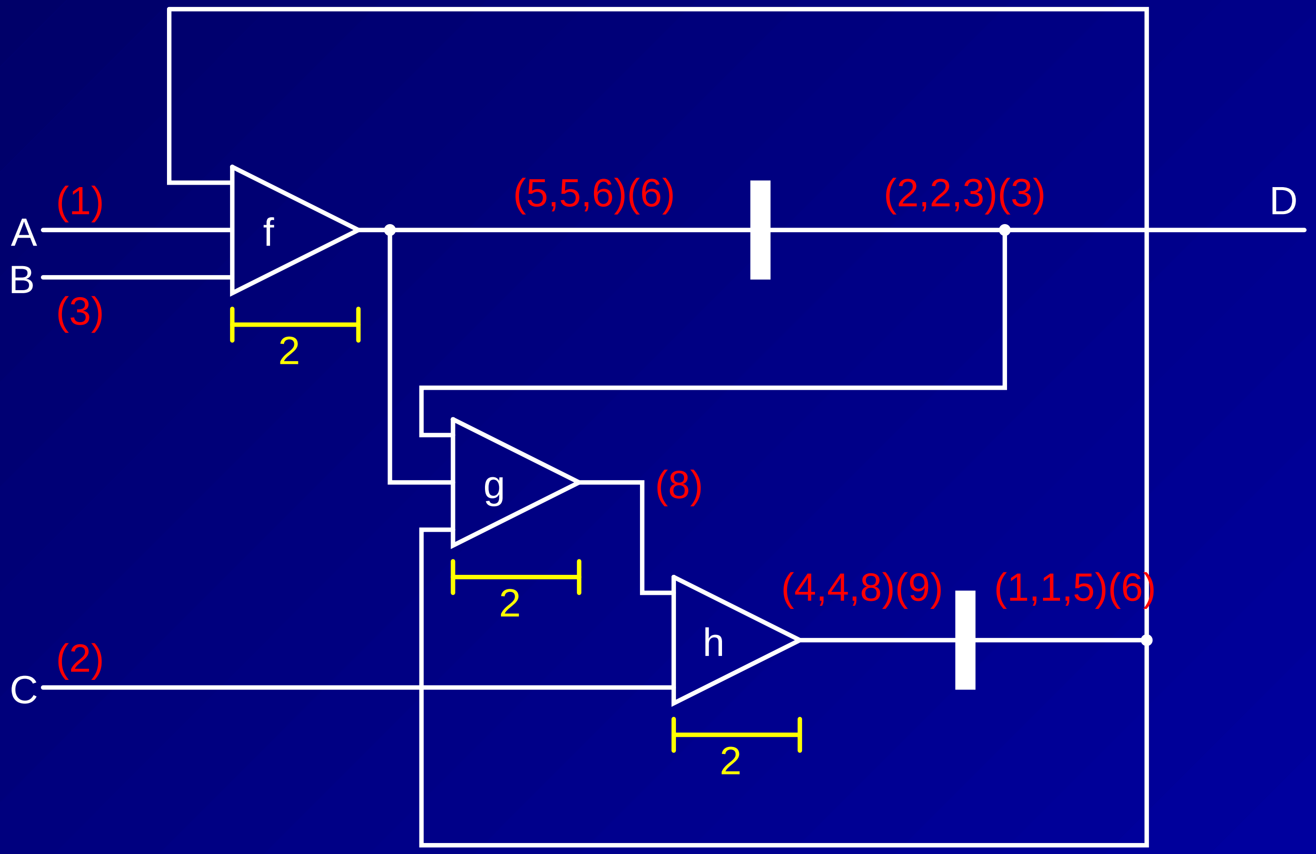
Bellman-Ford : relaxing ...



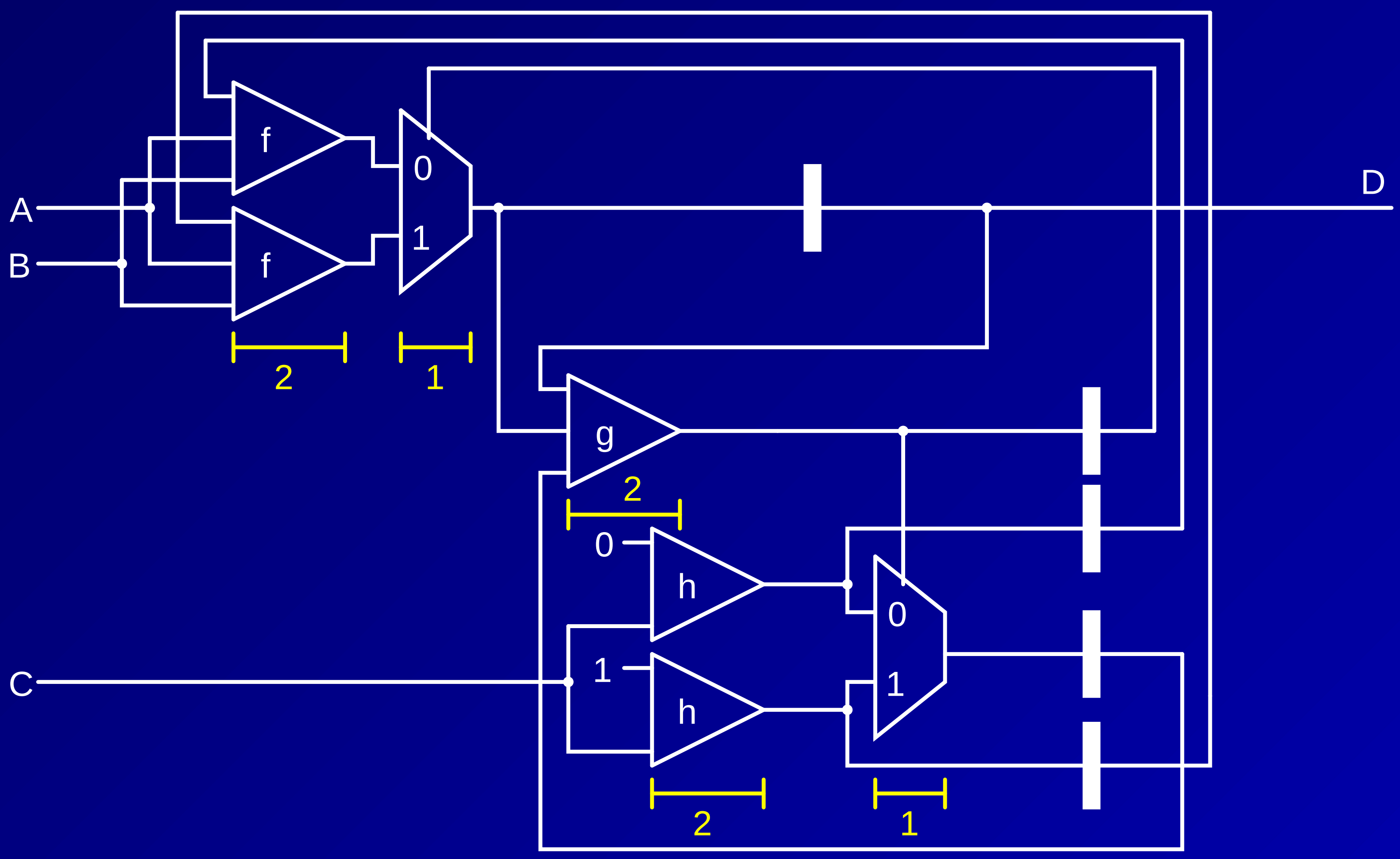
Bellman-Ford : relaxing ...



Bellman-Ford : fix point found

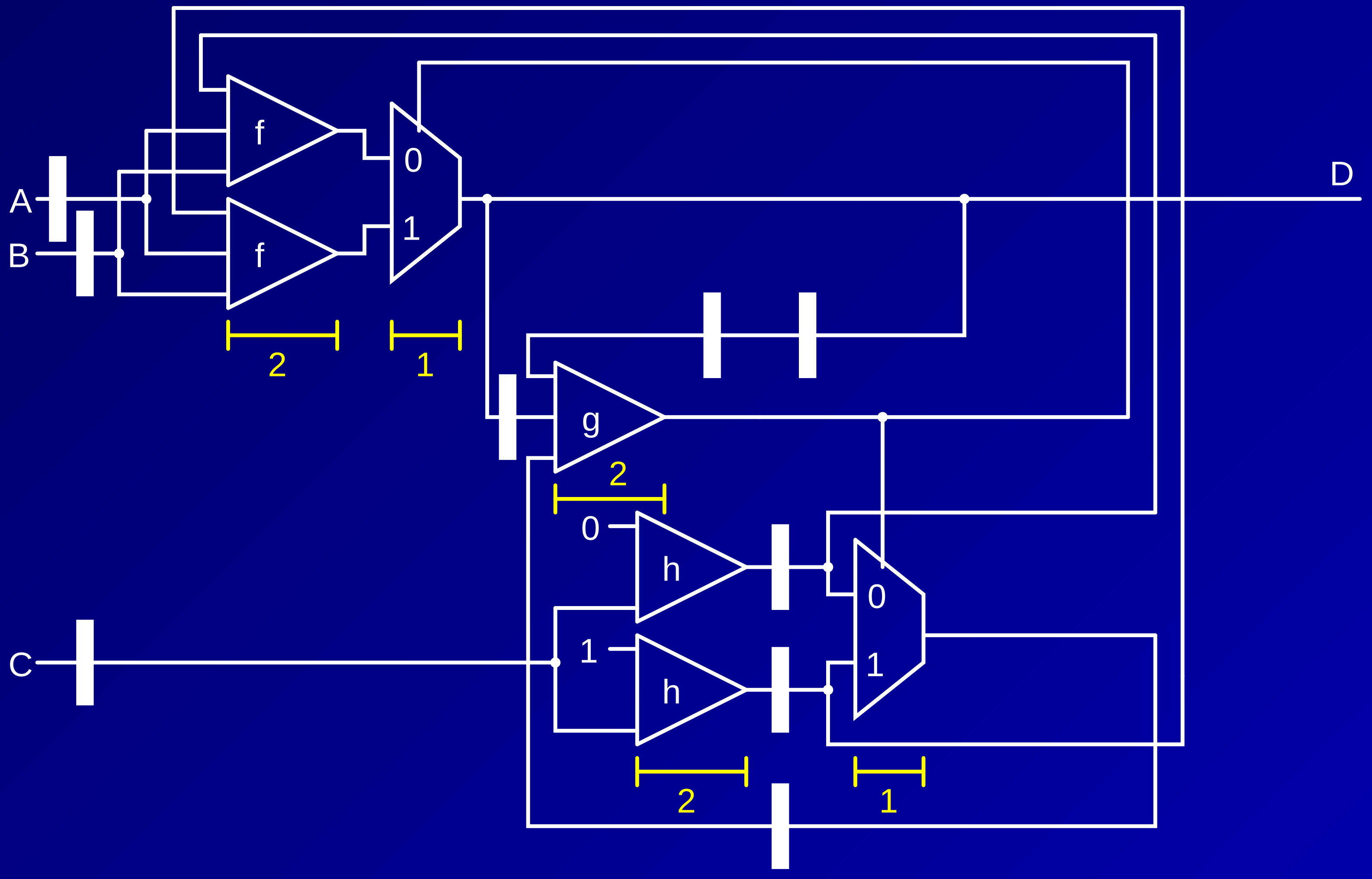


ShannonTransform



period=9

Retiming. Final solution



period=3

ISCAS89 sequential benchmarks

	reference		retimed		Sh. + ret.		time	speed	area
	period	area	period	area	period	area	(s)	up	penalty
s510	8	184	8	184	8	184	0.5		
s641	11	115	11	115	9	122	1.1	22%	6%
s713	11	118	11	118	10	121	0.9	10%	3%
s820	7	206	7	206	7	206	0.5		
s832	7	217	7	217	7	217	0.4		
s838	10	154	10	154	8	162	2.6	25%	5%
s1196	9	365	9	365	9	365	0.6		
s1423	24	408	21	408	13	460	3.8	61%	12%
s1488	6	453	6	453	6	453	0.7		
s1494	6	456	6	456	6	456	0.8		
s9234	11	662	8	656	8	684	6.7		
s13207	14	1382	11	1356	9	1416	18.0	22%	4%
s38417	14	7706	14	7652	13	7871	113	7%	3%

Contributions. Conclusions

- Theoretical model
  - a new way to describe complex combinations of Shannon decompositions
  - *fat* sets : generalized arrival times for “encoded” signals; extension of the Bellman-Ford algorithm
- Algorithm
  - Simultaneously analyses a large design space of Shannon decompositions and retiming.
  - Optimal performance; area heuristics
  - Promising results; short execution time