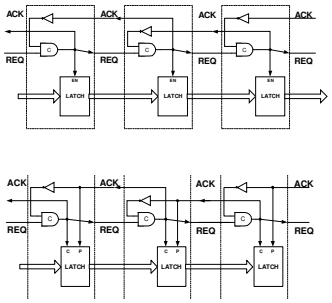
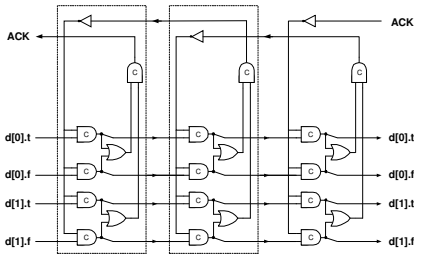
HY 590.20

Χρονισμός Ψηφιακών Συστημάτων
Χρήστος Σωτηρίου
6

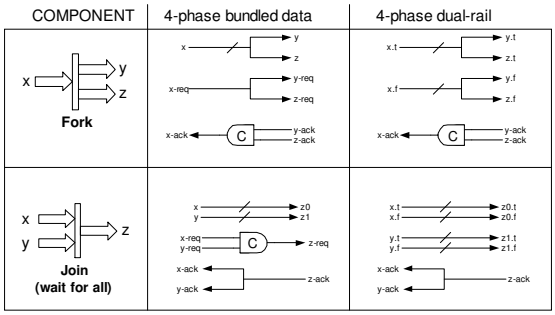
4- & 2-phase bundled data latches



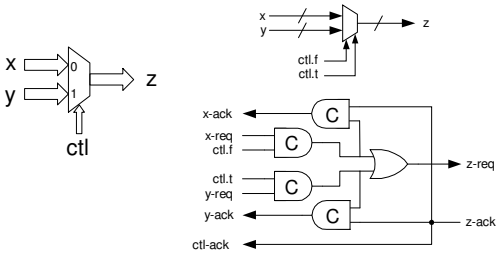
4-phase dual rail – many bits



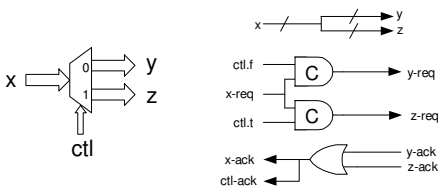
4-phase Fork, Join



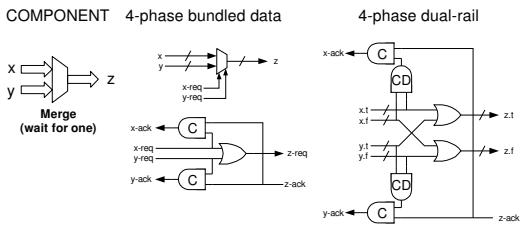
4-phase Bundled-data Mux



4-phase Bundled-data Demux

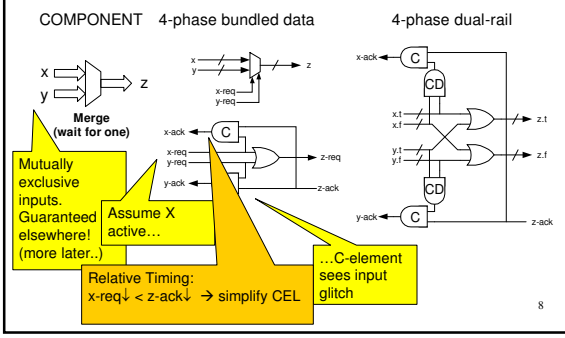


4-phase Merge



7

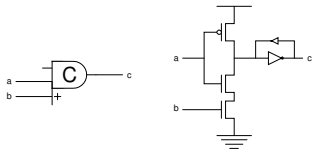
4-phase Merge



8

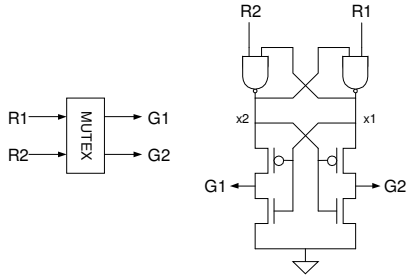
Asymmetric C Element

- Useful when we know the relative timing:
  b↓ < a↓ → only a↓ needed to pull up
- Only one pMOS - faster



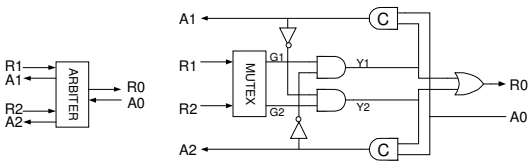
9

Mutual Exclusion: MUTEX



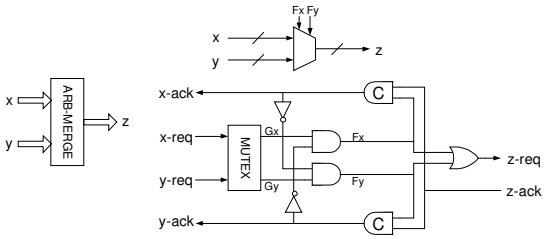
10

Arbiter



11

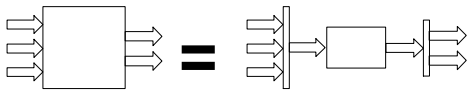
Arbitrating Merge



12

Function Blocks

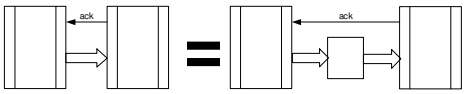
- We said "transparent" but…
  - Need a matched delay for bundled-data
  - Need to generate completion for dual-rail
  - Need to join inputs, fork outputs:



13

Transparency Revisited

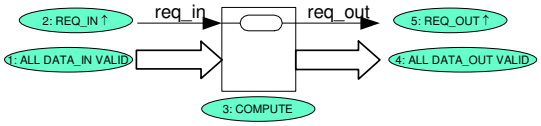
- Function blocks must not affect how the latches "shake hands" (except for timing)



14

Indication Revisited

- FB(req_out↑) means
  - FB(req_in↑)
  - Computation finished, data out ready
- Simple "strong indication" for bundled data:



15

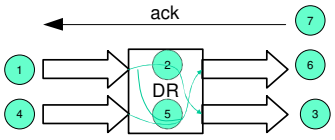
Strong vs. Weak Indication

- Strong Indication: All inputs must arrive before any output is allowed ("indicated").
  - Even if some outputs are ready earlier, there is no REQ_OUT, so they cannot be used.
  - Implies worst-case latency
- Weak Indication: Some outputs are allowed even before all inputs arrived
  - Only makes sense in dual-rail:

16

Weak Indication

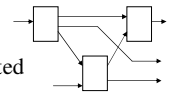
- No REQ on dual-rail – each bit is "self-indicating"
- May lead to faster circuits
- Example chain of events:



17

Composition of FBs

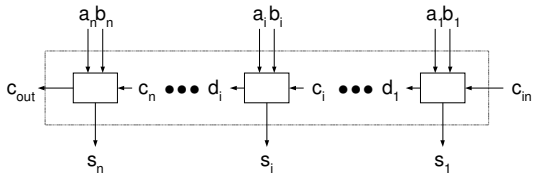
- Legal composition:
  - All inputs and outputs are connected
  - No cycles



- Legal composition of weekly indicating FBs is weakly indicating
- Legal composition of strongly indicating FBs is strongly indicating

18

Example: Ripple-carry



Example: Ripple-carry

- Full adder (a,b,c) = (s,d)
  - $s = a \oplus b \oplus c$
  - $d = ab + ac + bc$
- Shortcuts for look-ahead (prop, gen, kill):
  - $p = a \oplus b$ $\quad s = p \oplus c$
  - $g = ab$ $\quad d = g + pc$, OR $d' = k + pc'$
  - $k = a' b'$
- Sometimes d can be made valid without waiting for c

Ripple Carry Adder- SR operands, DR carry bits

- SR Implementation:
  - $s_i = c_{i-1} (a_i b_i + a_i' b_i') + c_{i-1}' (a_i' b_i + a_i b_i')$
  - $c_i = a_i b_i + c_{i-1} (a_i + b_i)$
- Carry DR only Implementation:
  - $s_{i\_T} = c_{i-1\_T} (a_i b_i + a_i' b_i') + c_{i-1\_F} (a_i' b_i + a_i b_i')$
  - $s_{i\_F} = c_{i-1\_T} (a_i' b_i + a_i b_i') c_{i-1\_F} (a_i b_i + a_i' b_i')$
  - $c_{i\_T} = a_i b_i + c_{i-1\_T} (a_i + b_i)$
  - $c_{i\_F} = (a_i' + b_i') c_{i-1\_F} + (a_i' b_i')$

Telescopic Units

- 16 bit ripple-carry adder, bundled-data
- Longest carry is 16 stages
- But if $p_8=0$ then longest carry is 8 stages
- And if $p_{12}p_8p_4=0$, then longest carry is 4 stages
- If willing to trade area and power for speed:

Telescopic Units – Multiple Delays