Energy-Aware Computing

Lecture 13: Self-timed systems

Outline

- Description of self-timed circuit characteristics
- Potential advantages and problems
- Handshake protocols
- Data representation and Indication
- Pipelines
- Circuit types

Self-timed systems

- Self-timed or asynchronous systems are systems which do not use a global clock signal to signify when data are ready
- Local, *handshake* signals are used instead
  - A request-acknowledge protocol is used

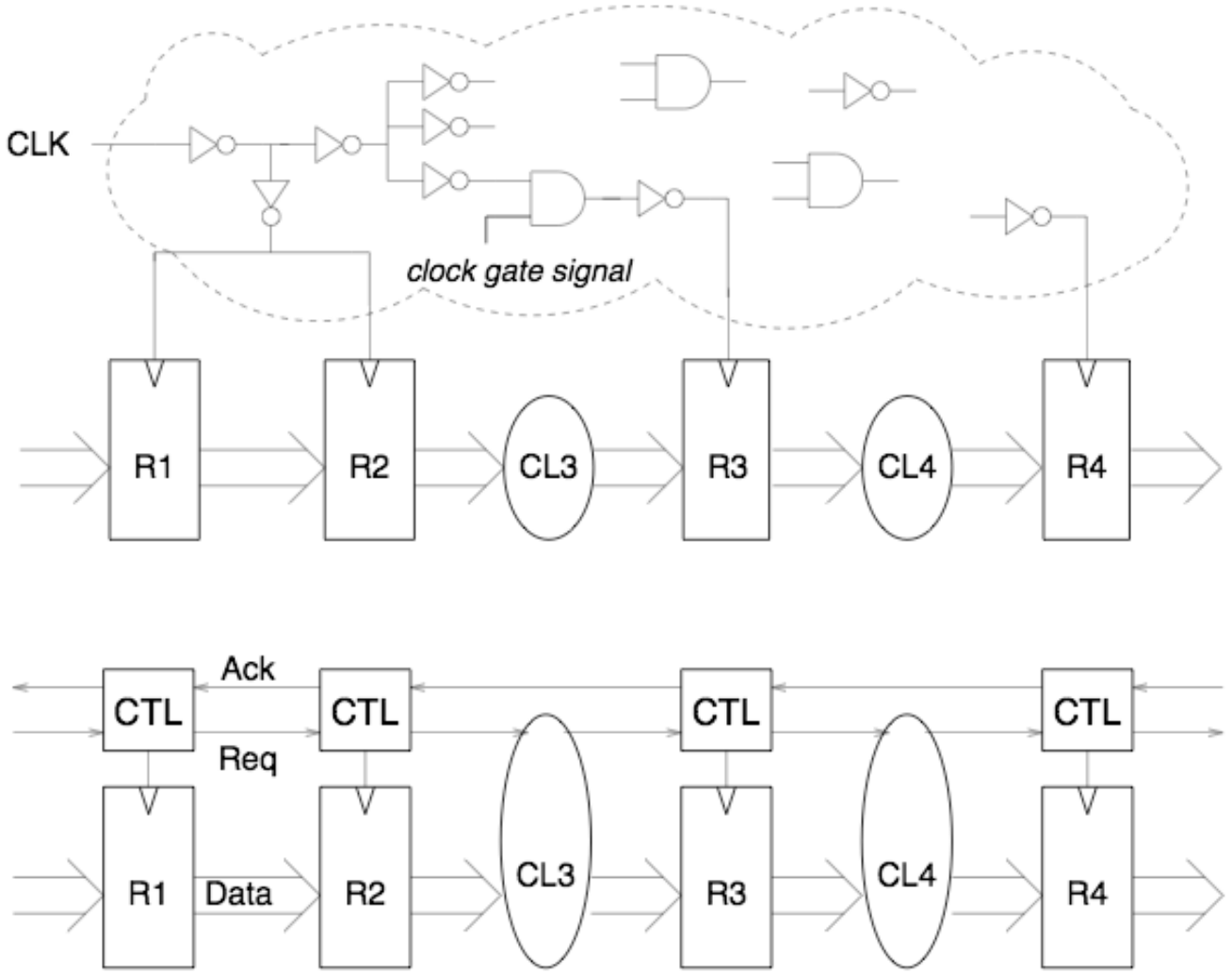
Potential advantages

- Low power consumption
  - No clock tree to drive
    - Up to 40% of total power
  - Automatic input guarding
    - No new inputs, no transitions
- Speed
  - Typical rather than worst case delays
- Modularity
  - previously designed units can be put together as long as they use the same protocol - they don't need to work at the same clock rate
- Low electromagnetic interference
  - Activity more spread out - not everything happens at clock edges

Problems

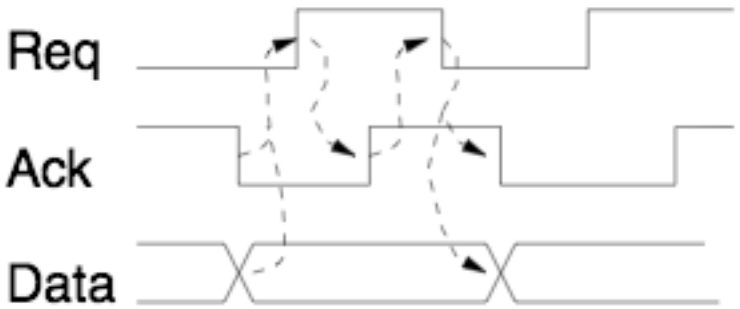
- Little support from EDA tools
- Problems with testing
- Speed/power advantages marginal or proven in niche applications only
- Harder to design
  - Glitches, hazards need to be considered

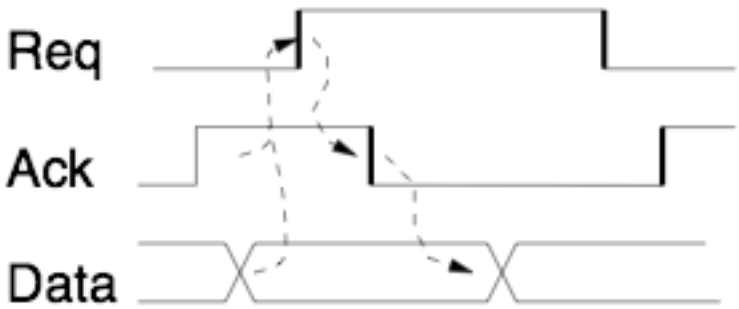
Example: pipeline



To return to zero or not?

- Handshake protocols can be of two types:
  - 2-phase (or non-return to zero)
  - 4-phase (return to zero)





When are data ready?

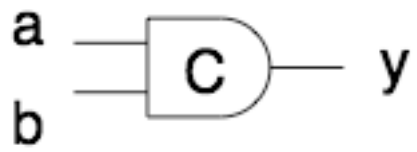
- Match the delay with a tracking circuit
  - Implied in previous figures
  - Called bundled data
  - Very similar to synchronous circuits
- Encode “readiness” in the data using special codes
  - Called delay-insensitive codes
- E.g. dual-rail code: 2 wires per bit At, Af
  - 10 = 1, 01 = 0, 00 = not ready (spacer), 11 = illegal/unused

Indication

- Asynchronous circuits cannot tolerate hazards in most cases:
  - If a signal changes should the next gate act on it or not?
- When an OR gate changes from 1 to 0
  - We know both inputs are 0
- When is changes from 0 to 1
  - We can't determine the values at both inputs
- OR gates *indicates* only when both inputs are 0
- AND gates indicate only when both inputs are 1
- XOR gates indicate all single input changes

Muller C-element

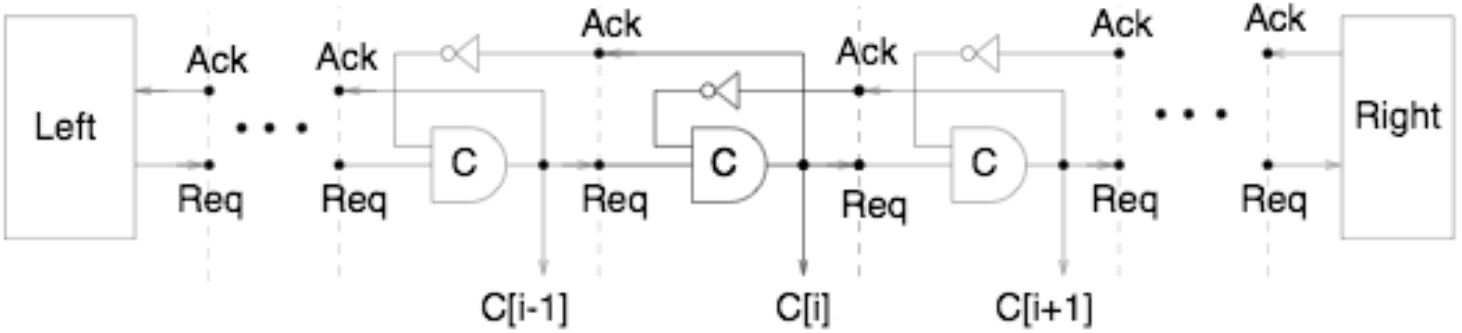
- In many cases we need to know when both inputs are 0 and when both are 1

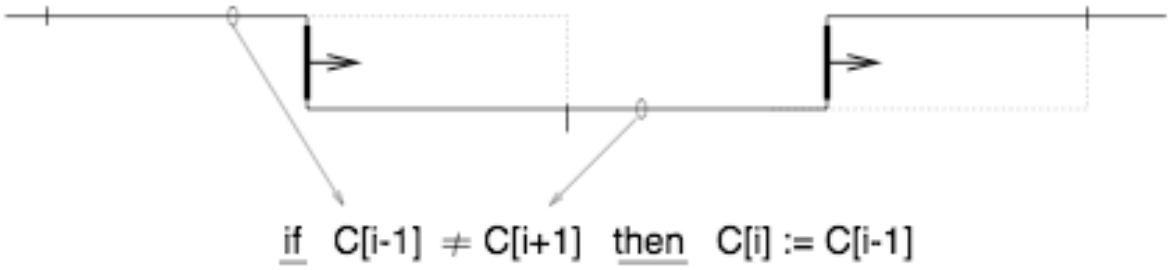


a	b	y
0	0	0
0	1	no change
1	0	no change
1	1	1

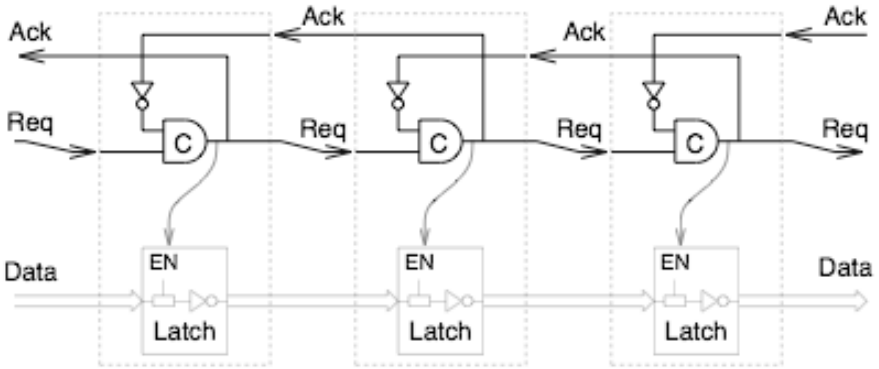
- All handshaking requires cyclic transitions between 1, 0
  - Controllers use C-elements

Muller pipeline

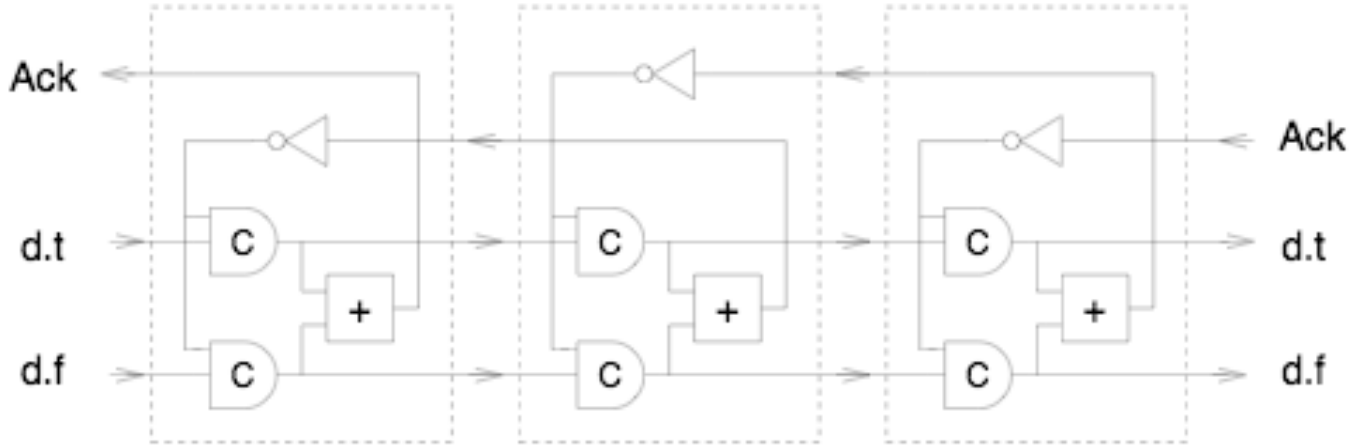




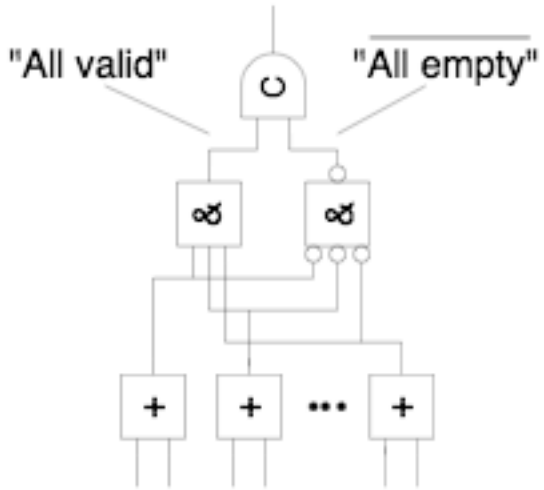
if C[i-1] ≠ C[i+1] then C[i] := C[i-1]



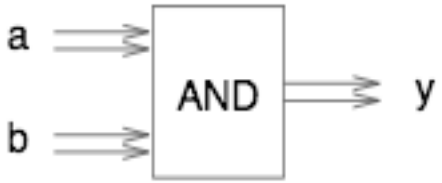
Delay insensitive pipelines



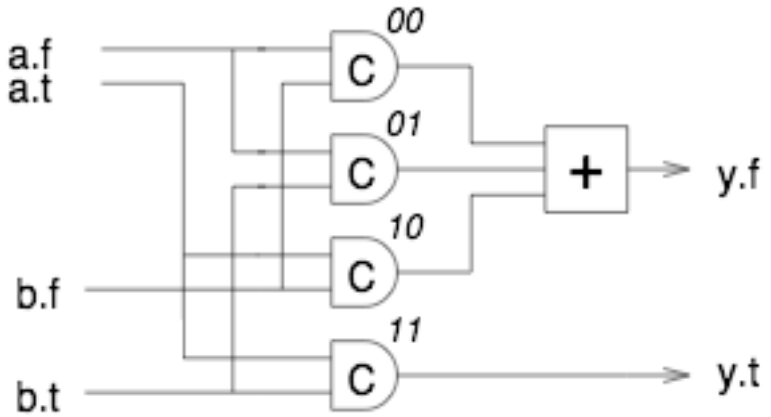
For multiple inputs, completion detection is more complicated:



DI logic operations



a	b	y.f	y.t
E	E	0	0
		NO CHANGE	
F	F	1	0
F	T	1	0
T	F	1	0
T	T	0	1



Timing assumptions

- Circuit implementation depends on timing assumptions
- Delay insensitive circuits
  - Positive, bounded but unknown delays in all gates and wires
  - Only inverters and C elements!
- Quasi-DI circuits
  - As above, but some wire forks are isochronic