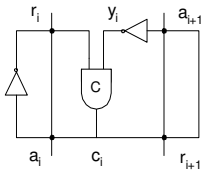
HY 590.20

Χρονισμός Ψηφιακών Συστημάτων
Χρήστος Σωτηρίου
4

Classification of Async Circuits

- Self-timed (ST)
  - Requires some timing assumptions
- Speed-independent (SI)
  - Zero (ideal) wire delay, arbitrary gate delay
- Delay-insensitive (DI)
  - Arbitrary delays
- Quasi-delay-insensitive (QDI)
  - DI with the Isochronic Fork assumption
  - Theoretically equivalent to SI
- SI and DI are mathematically provable

Speed Independence (SI)



Concurrent Boolean Functions

$r_i' = \text{not}(c_i)$.
$c_i' = r_i y_i + (r_i + y_i) c_i$.
$y_i' = \text{not}(a_{i+1})$.
$a_{i+1}' = c_i$.

- Circuit is modeled as a **closed** network of elements (Muller).
- A gate with output **z** is either *stable* **(zi'=zi)** or *excited* **(zi'!=zi)** depending on its inputs.
- An excited gate eventually fires and become stable.
- **SI**: Firing of one gate must never cause another excited gate to become stable without firing.

Data Flow Structures

- Abstraction similar to synchronous RTL
  - Likewise, described by either schematics or HDL
- Applies to all (3) handshake protocols, but we ***assume 4-phase***:
  - Alternating data tokens and bubbles.
- Assume handshake latches and handshake-ignorant function blocks
  - Recall token flow rules.

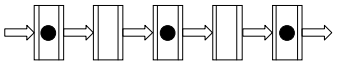
Token Flow

- Transfer of one token ≡ one handshake cycle
- Register *k* is FULL when it has data
- When register *k+1* gets the data from *k*,
  - Register *k+1* becomes FULL
  - Register *k* now has BUBBLE (≡ data that has already been copied)
- FULL register cannot receive data.
  Only BUBBLE register may receive data.

Comments on the Tokens Game

- Abstract all communications (handshake) and computations
  - Hide implementation details
- CL is transparent
  - Special type of CL required: "Function Blocks"
- Local "clocks" spread over time
  - Lower power
  - Lower emissions
  - No need to synchronize events
- Before playing more token games, let's consider some implementation details

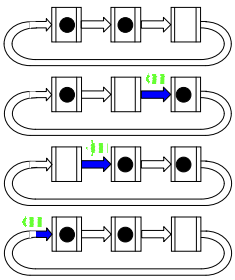
Abstract Pipeline



- Bubbles.
- Data Tokens
- Transparent function blocks (don't change token flow, only introduce delay)

7

Abstract Ring



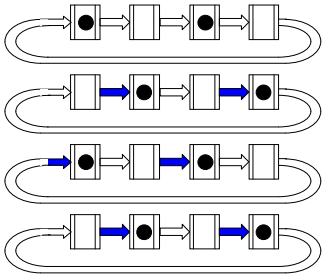
token

bubble

- 3 stages, 1 bubble:
  - 3 steps for token round
  - 6 steps to cycle

8

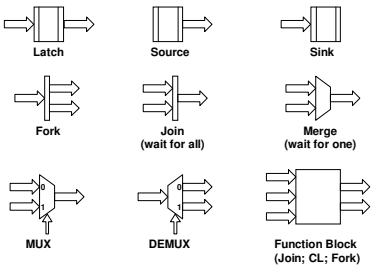
Abstract Ring



- 4 stages, 2 bubbles:
  - How many steps to cycle ?
- An added latch did not change the function (*unlike sync. pipeline*)

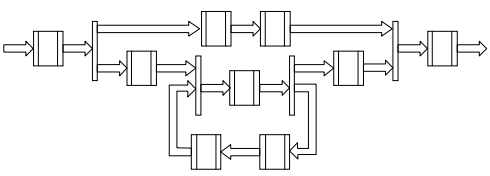
9

Building Blocks



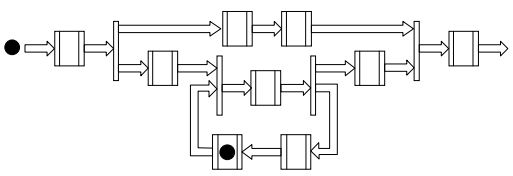
10

Example



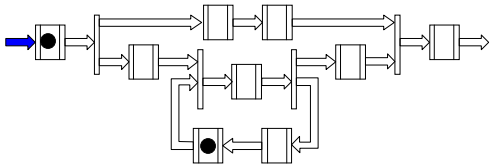
11

Example: time = t0



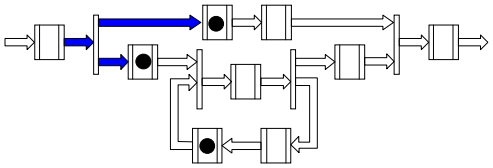
12

Example: time = t1



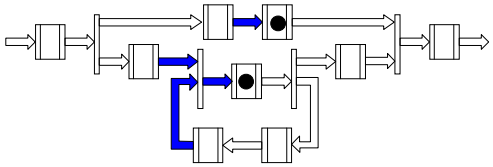
13

Example: time = t2



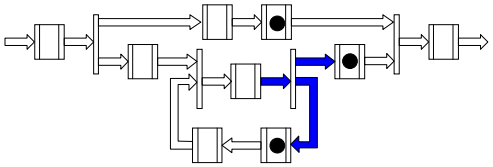
14

Example: time = t4



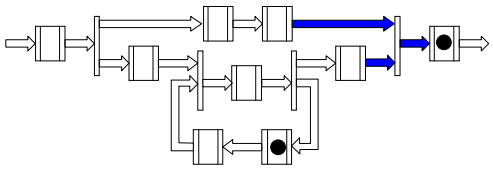
15

Example: time = t5



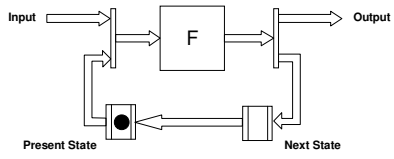
16

Example: time = t6



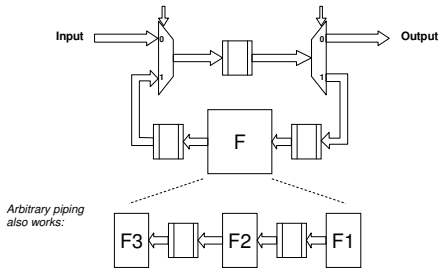
17

Another Ring: Simple AFSM



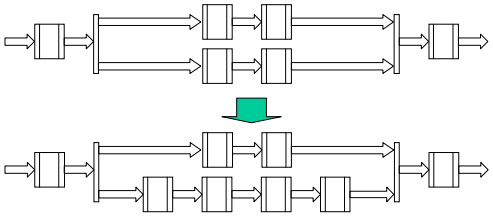
18

Another Ring: Iterative Computation



19

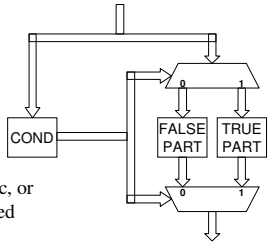
Latches **don't foul** the pipeline!



20

IF statement

`if` <COND> `then` <TRUE PART> `else` <FALSE PART>

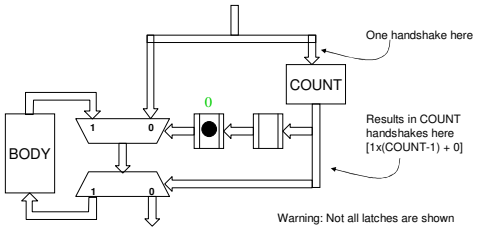


Combinational logic, or latches may be added

21

FOR statement

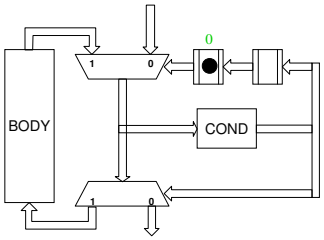
`for` <COUNT> `do` <BODY>



22

WHILE statement

`while` <COND> `do` <BODY>



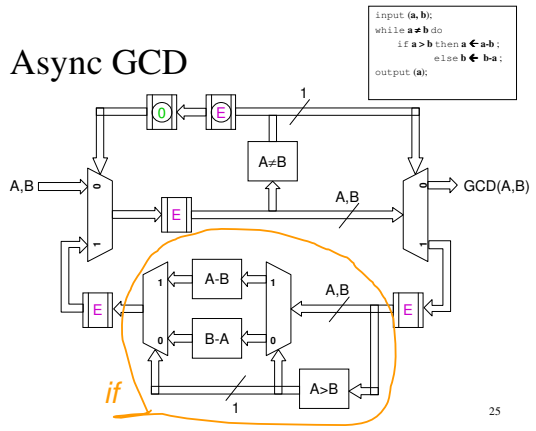
23

Async GCD

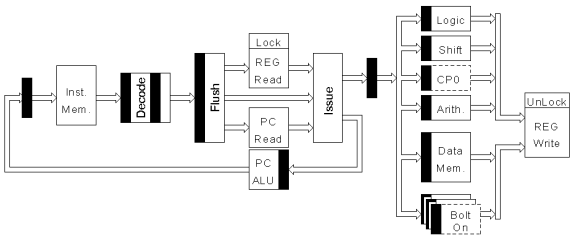
```
input (a, b);
while a ≠ b do
    if a > b  then a ← a-b ;
              else b ← b-a ;
output (a);
```

24

Async GCD



Async Processor (DTU TinyRISC)



Performance Analysis

- Synchronous performance analysis is ***simple***:
  - Check all register-to-register paths.
  - **Critical-path Based**
  - Static Timing Analysis (HY-523).
  - Dynamic simulations only check correctness, not performance.
- Asynchronous performance analysis is ***complex***:
  - Many internal cycles.
  - **Critical-cycle Based**
  - Data dependent delays (completion detection).
  - Dependency on environment and initialization (speed/response).
  - Harder to compute the solution

Performance Measures: **Latency**

- **Latency** (**L**): Delay from input to output.
- **Forward Latency** ($\mathbf{L_f}$): Latency, provided ACK has been waiting.
- $\mathbf{L_f}$: Forward latency of data token.
- **Reverse Latency** ($\mathbf{L_r}$): Delay from ACK at output to ACK at input, provided REQ has been waiting.
- $L_r\uparrow$, $L_r\downarrow$: Reverse latencies for ACK↑, ACK↓.

Performance Measures: **Period**

- **Period** (P): Delay from input of Data token to input of next Data token
- **4-phase period**: $P \geq L_f + L_r\uparrow + L_f + L_r\downarrow$
- Symmetric circuit:
  $P \geq 2\,L_f + 2\,L_r$
- ACK signals are overhead

Performance Measures:**Throughput**

- **Throughput** (T): Number of Data tokens that flow through circuit per unit time:

$$T = 1/P$$

Performance Measures: **Dynamic Wavelength**

- **Dynamic wavelength** (W_d): In a pipeline, number of pipeline stages that a (forward-going) token passes during P:

$$W_d = P/L_f$$

- **Lower Bound of $W_d = 1$**
- Also: Number of pipeline stages between successive Data tokens when tokens flow.

31

Performance Measures: **Static Spread**

- **Static spread** (S): In a pipeline, number of pipeline stages between successive Data tokens when the pipe is full (no bubbles).
- **Occupancy**: Number of Data tokens per pipeline stage: 1/S

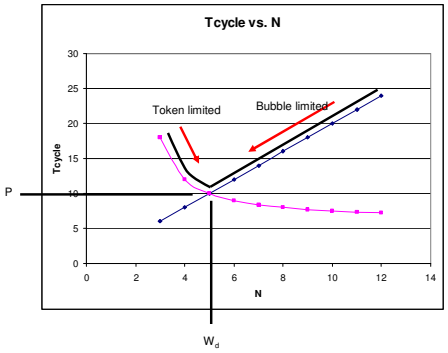
32

Ring Performance: Ring Cycle

- **Ring Cycle Time** (T_{cycle}): Roundtrip time of one token in the ring.
- Best performance (=minimum T_{cycle}):
  - Number of stages per token = Dynamic wavelength
  - $T_{cycle} = P$
- Best performance a.k.a. "matched slack"
- If fewer stages, T_{cycle} limited by lack of bubbles
- If more stages, T_{cycle} limited by forward latency

33

Ring Performance: Ring Cycle Time vs. Number of Bubbles



34