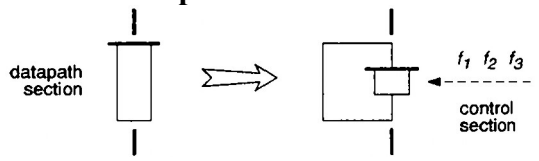
VLSI I – Short Summary

by Stephan Senn, D-ITET

Universal Transformations

Iterative Decomposition



d: level of decomposition

$$\frac{A_f}{d}+A_{reg}\le A(d)\le A_f+A_{reg} \quad , \quad L(d)=d$$

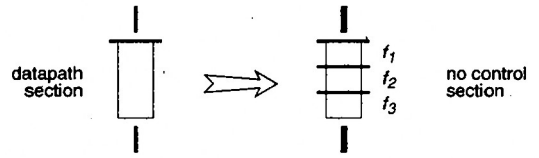
$$cpd(d)=d \quad , \quad t_{lp}(d)=\frac{t_f}{d}+t_{reg} \quad , \quad T(d)=d\cdot t_{lp}(d)$$

$$E_f+d\,E_{reg}<E(d)$$

$$d\,A_{reg}t_{reg}+A_{reg}t_f+A_f t_{reg}+\frac{1}{d}A_f t_f\le AT(s)$$

$$AT(d)\le d(A_f+A_{reg})t_{reg}+(A_f+A_{reg})t_f$$

Pipelining



p: pipeline stages

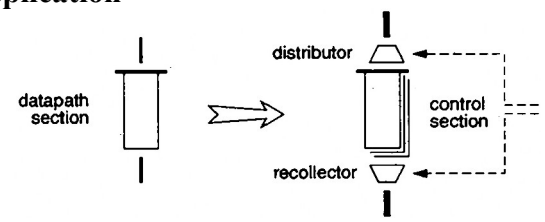
$$A(p)=A_f+p\,A_{reg} \quad , \quad L(p)=p \quad , \quad cpd(p)=1$$

$$t_{lp}(p)=\frac{t_f}{p}+t_{reg} \quad , \quad T(p)=1\cdot t_{lp}(p)$$

$$AT(p)=p\,A_{reg}t_{reg}+A_{reg}t_f+A_f t_{reg}+\frac{1}{p}A_f t_f$$

$$E(p)=E_f+p\,E_{reg}$$

Replication



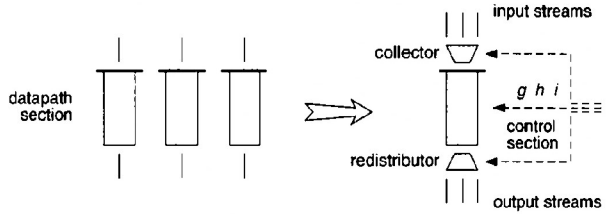
q: number of replicated items

$$A(q)=q(A_f+A_{reg}) \quad , \quad L(q)=1 \quad , \quad cpd(q)=\frac{1}{q}$$

$$t_{lp}(q)=t_f+t_{reg} \quad , \quad T(q)=\frac{1}{q}\cdot t_{lp}(q)$$

$$AT(q)=(A_f+A_{reg})(t_f+t_{reg}) \quad , \quad E(q)=E_f+E_{reg}$$

Time Sharing



s: number of shared resources

$$\max(A_f)=(A_g,A_h,A_i) \quad , \quad \max(t_f)=(t_g,t_h,t_i)$$

$$\sum A_f=(A_g+A_h+A_i) \quad , \quad \sum E_f=(E_g+E_h+E_i)$$

$$\max(A_f)+A_{reg}\le\sum A_f+A_{reg}$$

$$L(s)=s \quad , \quad cpd(s)=s \quad , \quad t_{lp}(s)=\max(t_f)+t_{reg}$$

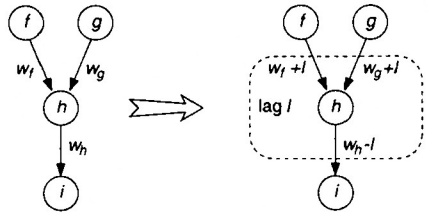
$$T(s)=s\cdot t_{lp}(s)$$

$$s[\max(A_f)+A_{reg}][\max(t_f)+t_{reg}]\le AT(s)$$

$$AT(s)\le s[\sum A_f+A_{reg}][\max(t_f)+t_{reg}]$$

$$s\max(E_f)+E_{reg}\le E(s)$$

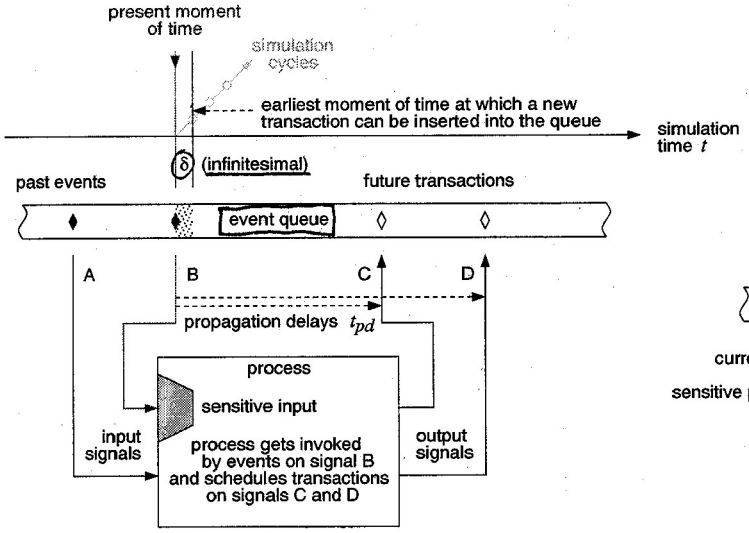
Retiming

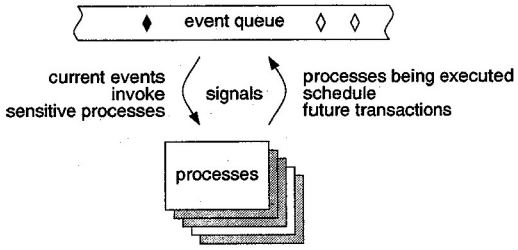


The parameters A, t, L, cpd and E don't change.

The longest path t_{lp} is shortened!

Discrete Timed Event-Driven Simulation



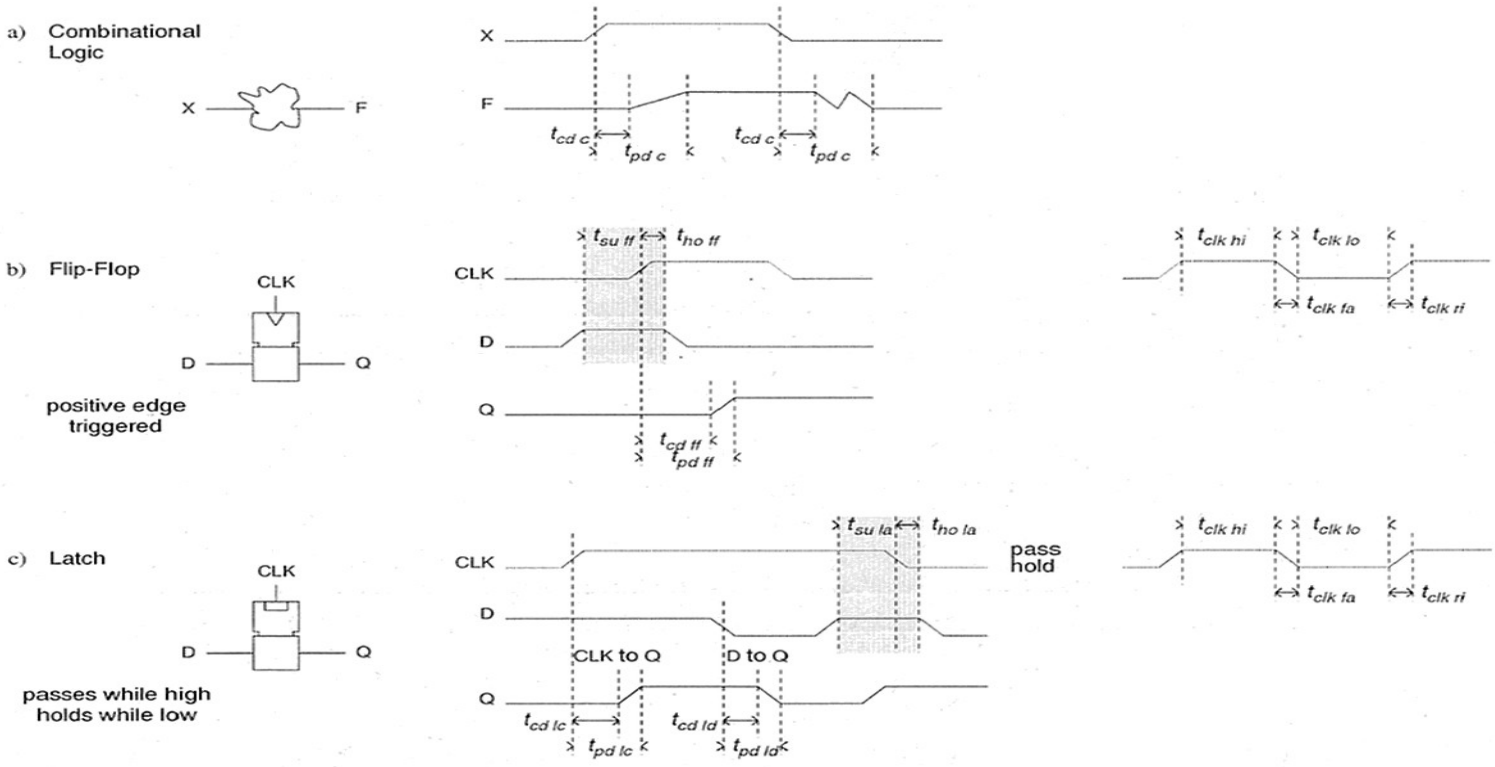


Design Parameters

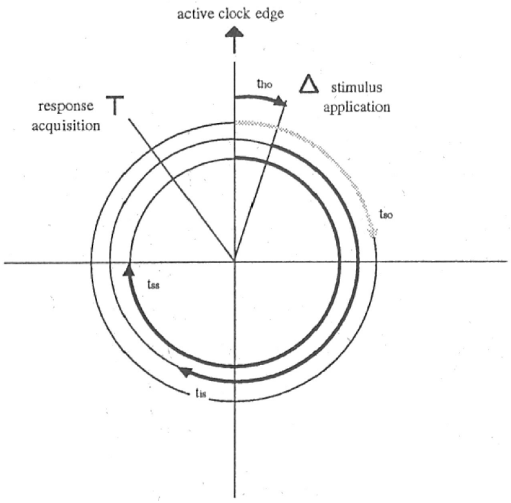
- Circuit Size : A
- Latency : L
- Cycles per Data Item : cpd
- Computation Period : T_{cp}
- Longest Path Length : $t_{lp} \le T_{cp}$
- Computation Rate : $f_{cp} = \frac{1}{T_{cp}}$
- Time per Data Item : $T = cpd \cdot T_{cp} \ge cpd \cdot t_{lp}$
- Data Throughput : $\Theta = \frac{1}{T}$
- Size-Time Product : $AT = \frac{A}{\Theta}$
- Energy per Data Item : $E = PT = \frac{P}{\Theta}$
- Power-Delay Product : $pdp = P \cdot t_{lp}$
- Clock Period : T_{clk} *(=T_{cp} if single-edge triggered)*

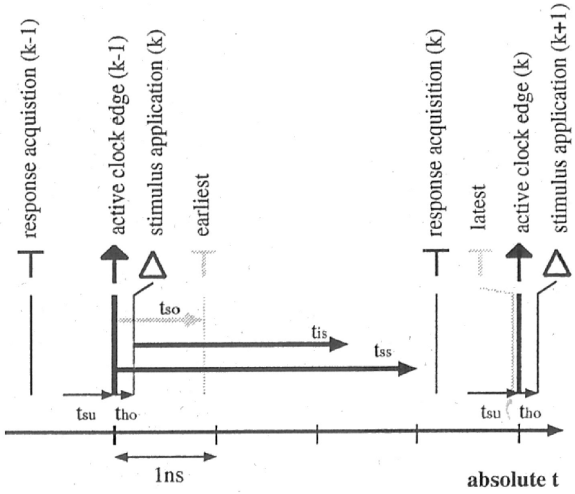
Timing

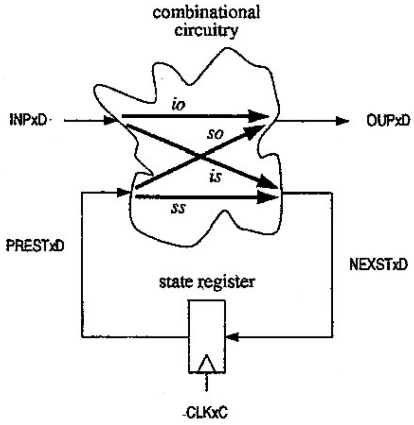
Combinational Logic, Flip-Flop and Latches

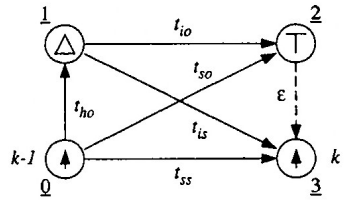


Anceau Diagram and FSM-Timing Scheme









events:

- active clock edge
- stimulus application
- response acquisition

Hold-Time Condition:

$t_{ho}(+t_{hi}) < t_{cd} + t_{cd,d}$ t_{hi}: *high-time of a latch*

Timing of a System:

$t_{su,sys} = -t_{d,clk} + t_{su,ff} + t_{cd,comb}$ $t_{d,clk}$: *clock delay*

$t_{ho,sys} = t_{d,clk} + t_{ho,ff} - t_{cd,comb}$

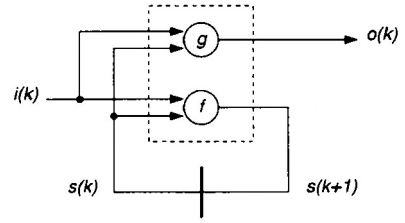
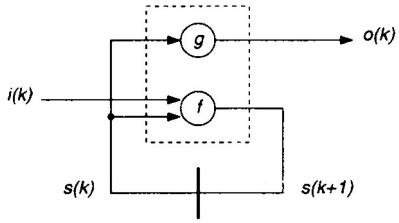
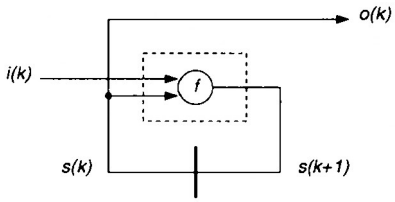
$t_{pd,sys} = t_{d,clk} + t_{pd,ff} + t_{pd,comb}$

$t_{cd,sys} = t_{d,clk} + t_{cd,ff} + t_{cd,comb}$

Sign-Rule for Setup- and Hold-Time:

- *Setup-Time:* $+ \leftarrow$
- *Hold-Time:* $\rightarrow +$

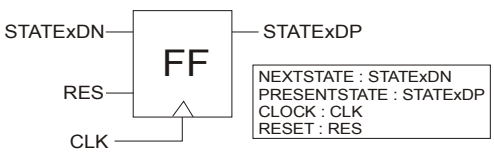
Finite State Machines

Mealy	**Moore**	**Medwedjew**
 $s(k+1) = f(i(k), s(k))$ $o(k) = g(i(k), s(k))$	 $s(k+1) = f(i(k), s(k))$ $o(k) = g(s(k))$	 $s(k+1) = f(i(k), s(k))$ $o(k) = s(k)$
Others:		
Autonomous Automata	**Delay Automata**	**Combinational Logic**
$s(k+1) = f(s(k))$, no input! $o(k) = g(s(k))$	$s(k+1) = f(i(k))$ $o(k) = g(s(k))$	$o(k) = g(i(k))$, no states!

VHDL Code Fragments

Flip-Flop

```
-- updating of state
-- sensitivity list, no more signals accepted!
memzing : process (CLOCK, RESET) is
begin
-- no other statement allowed here!
```



```
-- activities triggered by asynchronous active-high reset
if RESET = '1' then
      PRESENTSTATE <= STARTSTATE;
      ...
-- activities triggered by rising edge of clock
elsif CLOCK'event and CLOCK='1' then -- no more term allowed here!
      -- extra subconditions, if any, accepted here.
      PRESENTSTATE <= NEXTSTATE;
      ...
end if;
-- no more statement allowed here!
end process memzing;
```

Mealy and Moore FSM

```
memless : process (INPxD, STATExDP) is
begin
STATExDN <= STATExDP; -- remain in present state
case STATExDP is
when st0 => -- state in a Mealy FSM
      if INPxD = <in1> then OUPxD <= <out1>;
      elsif INPxD = <in2> then STATExDN <= <next_state1>;
      ...
      else OUPxD <= <out>; -- catch parasitic input!
      end if;
when st1 => -- state in a Moore FSM
      if INPxD = <in1> then STATExDN <= <next_state1>;
      elsif INPxD = <in2> then STATExDN <= <next_state2>;
      ...
      else STATExDN <= <state> -- catch parasitic input!
      end if;
      OUPxD <= <out1>;
...
when others => -- tie up parasitic states for synthesis!
end case;
end process memless;
```

What makes an Algorithm suitable for dedicated VLSI Architectures?

- Loose coupling between major processing tasks
- Simple control flow
- Regular data flow
- Reasonable storage requirements
- Compatible with finite precision arithmetics
- Non-recursive linear time-invariant computation
- No transcendental functions
- No divisions and multiplications on wide data words
- One argument fixed in computationally expensive operations
- Throughput rather than latency is what matters
- Simple initialization