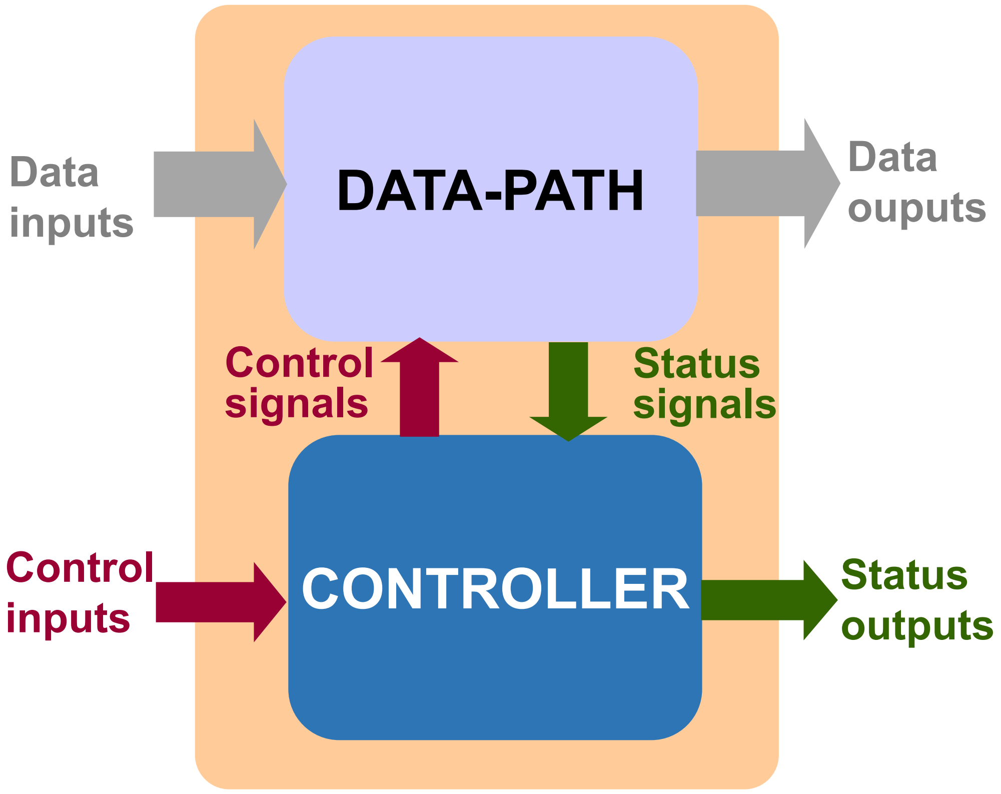
Deriving HW architectures and its control

- RTL system overview
- Deriving sequential architectures
- Strategy for complex control subsystems

RTL system design overview



Data-path
Circuitry to process the input data:

- combinational and sequential circuits

Controller
Circuitry to control the data-path operation:

- FSM

Processor: Controller and data-path components working together to implement an algorithm

Serial vs. parallel architecture

Ej: FIR filter algorithm

$$y(n) = \sum_{k=0}^{M-1} h_k x(n-k)$$

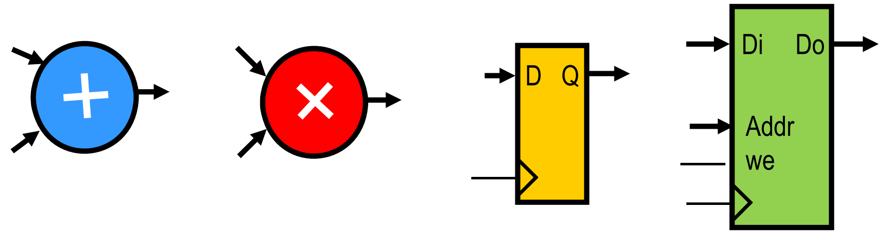
```
y(0) = 0
for k = 1 to M do
  y(k) = y(k-1)+ h(k)*x(k)
end
y = y(n)
```

Operations

- **Additions (+)**
- **Multiplications (*)**
- **Storage of variables (=)**

Operators

- **Adder**
- **Multiplicator**
- **Storage resources (registers or memory)**



Architecture derivation:

How many clock cycles do we have to obtain a result?

- **N clock cycles $\Rightarrow$ operators can be reused (several operations mapped in the same operator) $\Rightarrow$ serial architecture**
- **1 clock cycle $\Rightarrow$ as many operators as operations $\Rightarrow$ parallel arch.**

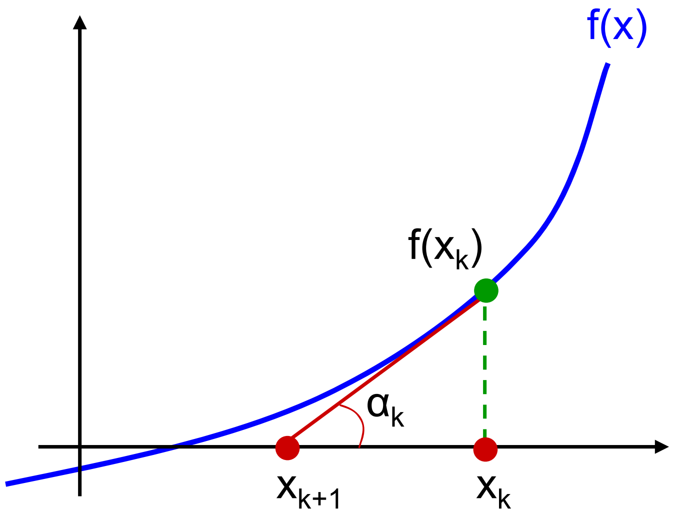
Deriving sequential architectures

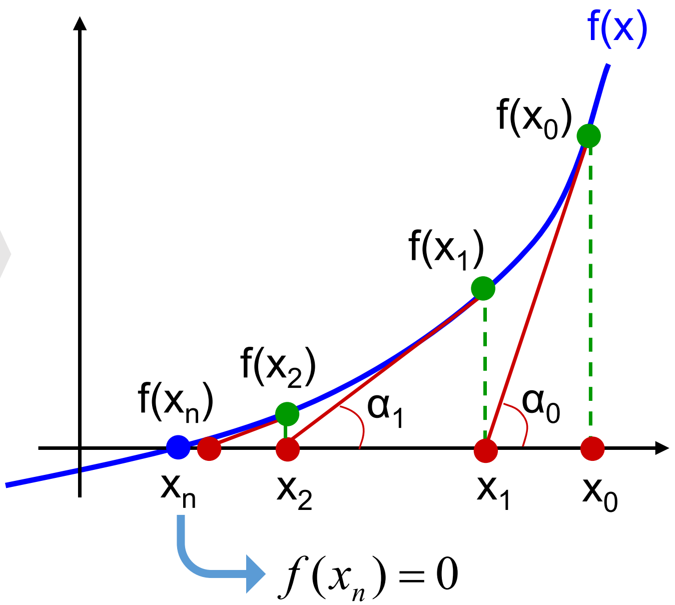
Ex. Newton-Raphson method: calculate the zeros of a function by means the following iteration:

$$\tan(\alpha_k) = f'(x_k) = \frac{f(x_k)}{x_k - x_{k+1}}$$

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}}$$

$$\boxed{f(x_\infty) = 0}$$





$f(x_n) = 0$

- To chose the proper initial value x_0 is important, a wrong value can avoid the algorithm's convergence
- Quadratic convergence: $e_{k+1}=c\cdot e_k^2$

Deriving sequential architectures

Newton-Raphson method: reciprocal, $x=1/d$, computation

$f(x) = 1/x-d \Rightarrow f(x) = 0 \Rightarrow x = 1/d$

$f'(x) = -1/x^2$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \Rightarrow x_{k+1} = x_k(2 - x_k d)$$

Matlab model

```
x=x0;
for i= 1:it
   x=x*(2-d*x);
end
y=x;
```

Operations per iteration: 2 multiplications and 1 addition

Example: d = 0.97, X_0 = 1.5, (x = 1/d = 1.030927835051547):

1ª iteración: $X_1 = 1.5 \cdot (2 - 1.4550) = 0.8175$

2ª iteración: $X_2 = 0.8175 \cdot (2 - 0.7930) = 0.9868$

3ª iteración: $X_3 = 0.9868 \cdot (2 - 0.9572) = 1.0290$

4ª iteración: $X_4 = 1.0290 \cdot (2 - 0.9981) = 1.0309$

$X_4 = 1.0309$

$f(X_4) = 1/x_4 - d = 1/1.0309 - 0.97 = 2.6e-5$

Deriving sequential architectures

Newton-Raphson method: reciprocal, $x=1/d$, computation

Sequential hardware architecture with 1 multiplier

Initialization: x_0 y d

Storage

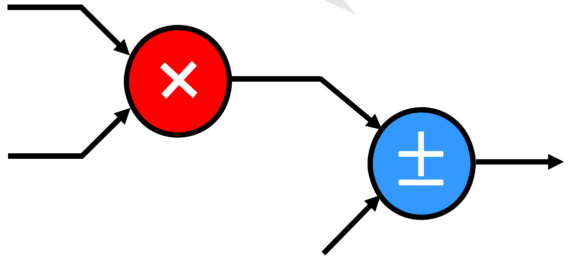
$$x_{k+1} = x_k \cdot (2 - x_k \cdot d)$$

Dependences between operations

1º $v_k = (2 - x_k \cdot d)$

2º $x_{k+1} = x_k \cdot v_k$

Basic operator



Deriving sequential architectures

Newton-Raphson method: reciprocal, $x=1/d$, computation

Sequential hardware architecture with 1 multiplier

Initialization: x_0 y d

Storage

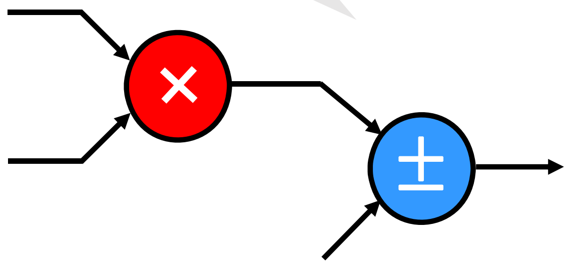
$$x_{k+1} = x_k \cdot (2 - x_k \cdot d)$$

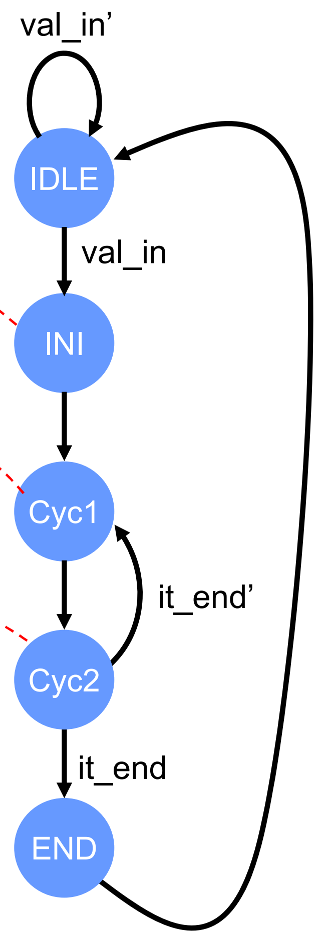
1º $v_k = (2 - x_k \cdot d)$

2º $x_{k+1} = x_k \cdot v_k$

Dependences between operations

Basic operator





Deriving sequential architectures

Newton-Raphson method: reciprocal, $x=1/d$, computation

Sequential hardware architecture with 1 multiplier

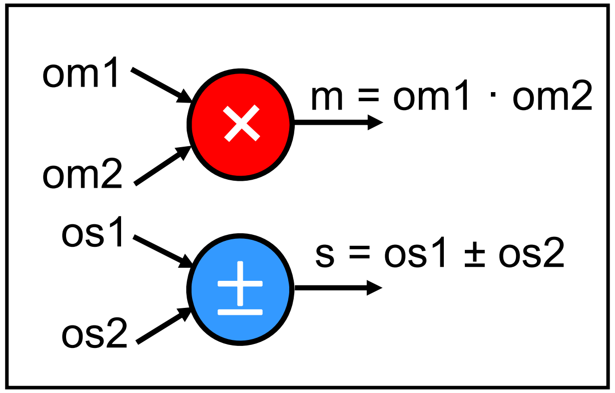
Functional model

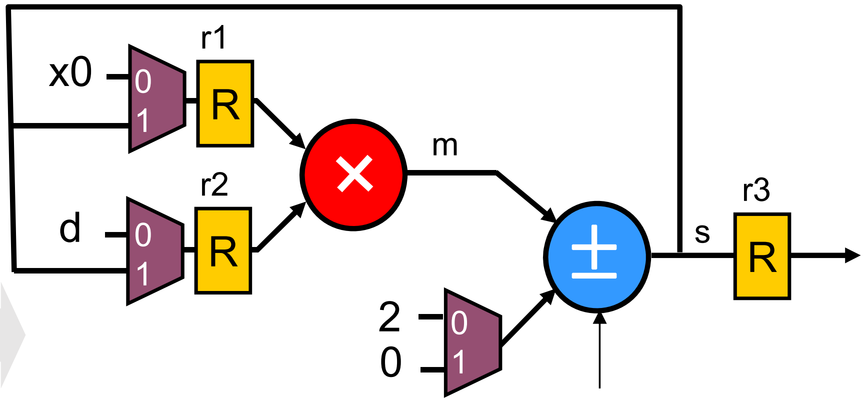
```
x=x0;
for i= 1:it
   x=x*(2-d*x);
end
y=x;
```

HW-oriented model

```
r1 = x0;
r2 = d;
for i = 1:it
   % 1st cycle
     m = r1*r2;
     s = 2-m;
     r_2 = s;
   % 2nd cycle
     m = r1*r2;
     s = 0+m;
     r1 = s;
     r2 = d;
end
r3 = s;
y = r3;
```

The operand's order must be kept





Deriving sequential architectures

Newton-Raphson method: reciprocal, x=1/d,

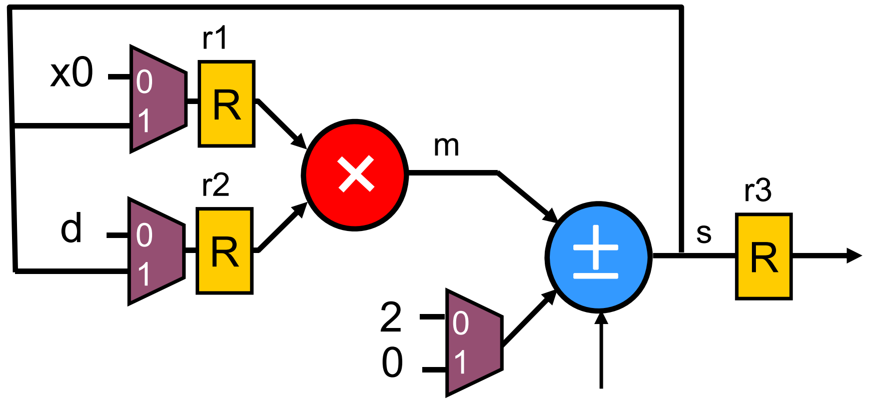
Sequential hardware architecture with 1 multiplier

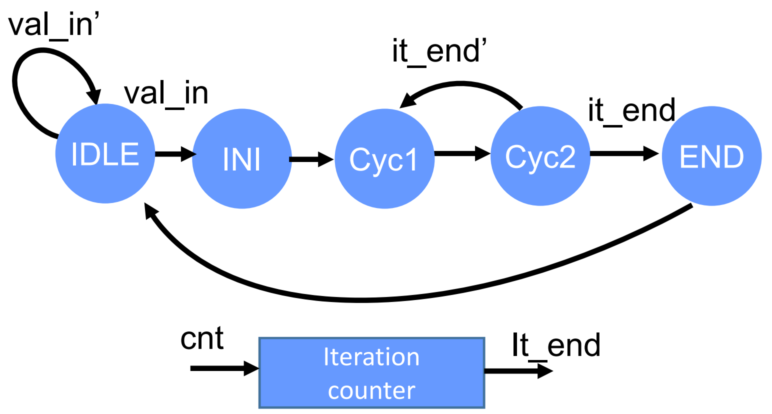
HW-oriented model

```
r1 = x0;
r2 = d;
for i = 1:it
   % 1st cycle
     m = r1*r2;
     s = 2-m;
     r_2 = s;
   % 2nd cycle
     m = r1*r2;
     s = 0+m;
     r1 = s;
     r2 = d;
end
r3 = s;
y = r3;
```

Control sequence

```
load_r1 = 1; sel_muxr1 = 0;
load_r2 = 1; sel_muxr2 = 0; rst_cnt=1;
for i= 1:it
   % 1st cycle
    cnt = 1;
    addsub = 1; sel_mux_as = 0;
    load_r1 = 0; load_r2 = 1; sel_muxr2 = 1;
   % 2nd cycle
   addsum = 0; sel_mux_as = 1;
    load_r1 = 1;  sel_muxr1 = 1;
    load_r2 = 1;  sel_muxr2 = 0;
end
load_r3 = 1;
rdy_out = 1;
```

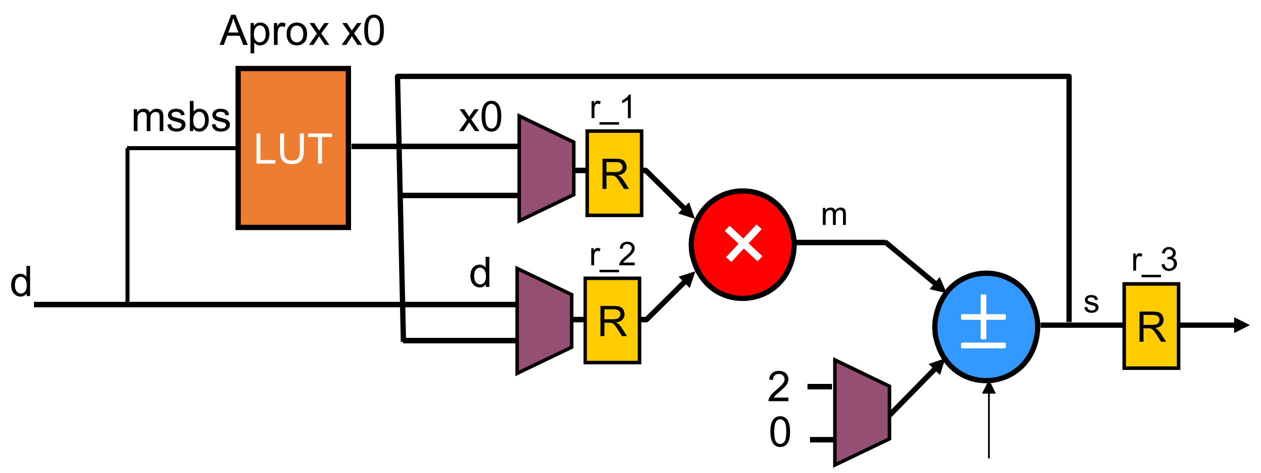




Deriving sequential architectures

Newton-Raphson method: reciprocal, x=1/d,

- Sequential hardware architecture with 1 multiplier
- A good selection of *x0* improves the algorithm's convergence
  - ⇒ a small LUT addressed by the most significant bits of d can be used

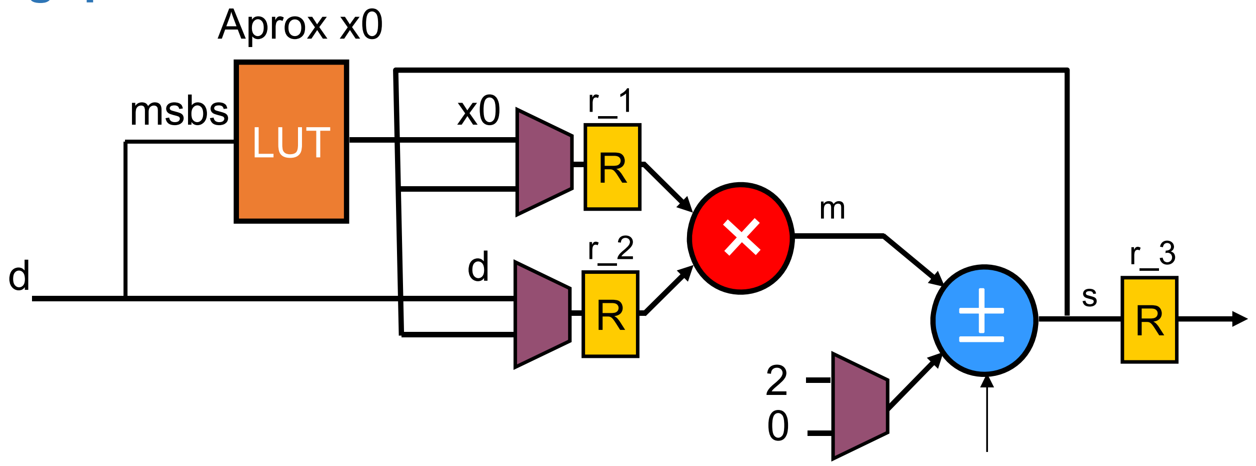


Deriving sequential architectures

Newton-Raphson method: reciprocal, x=1/d,

Sequential hardware architecture with 1 multiplier

Throughput?



Can this architecture be pipelined to increase the throughput?

Would we increase by 2 the throughput if two multipliers were used?

Can we build a very high throughput architecture for this algorithm? How?

Modelling complex control subsystems

Complex FSMs

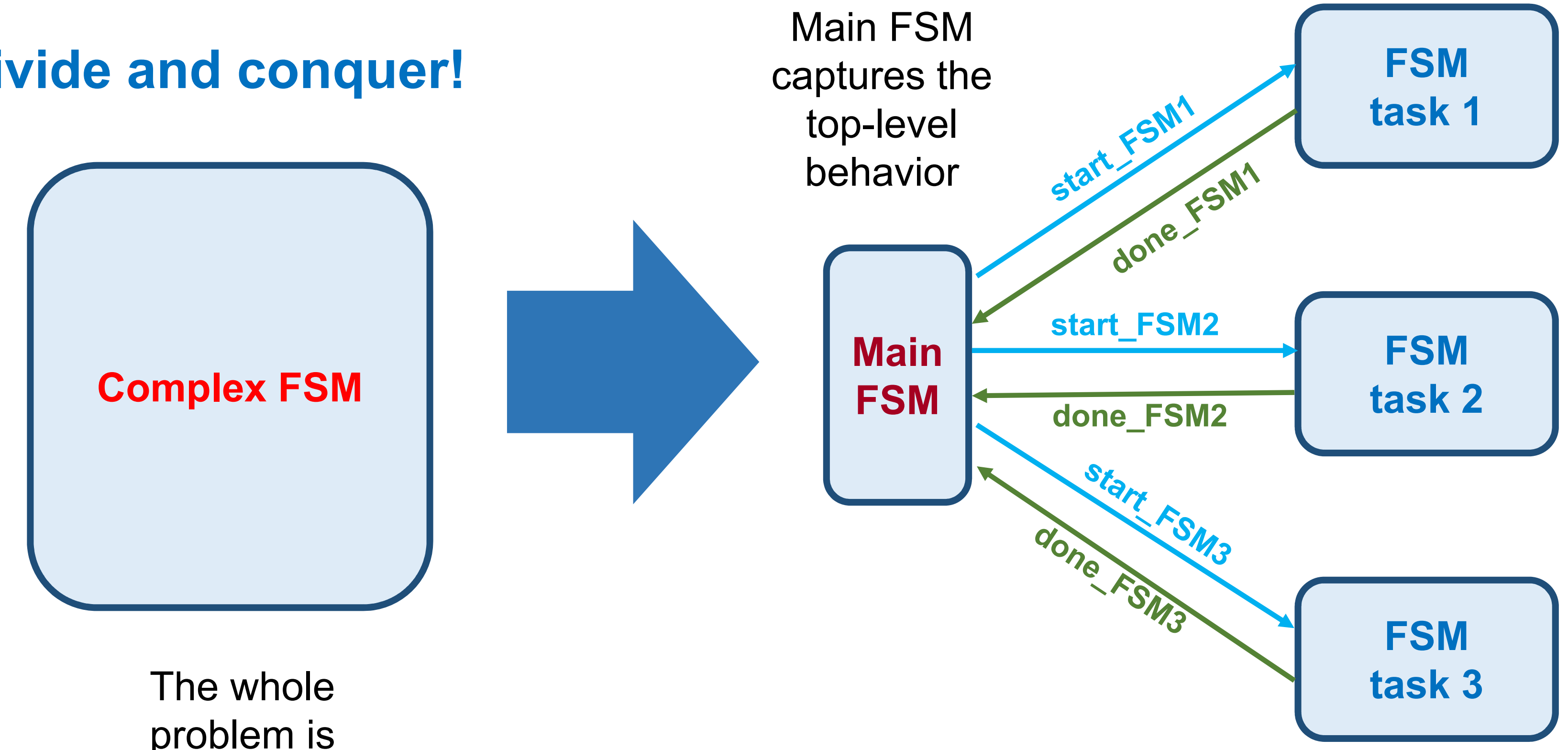
How to address the design of a complex control circuit?

⇒ Divide and conquer!

- Do not design a complex FSM
  - break the complex FSM down into several less-complex FSMs
- Most problems have a hierarchical solution:
  - a main FSM implements the top level of the hierarchy: ordered sequence of tasks
  - each task is developed by a specific FSM
    - the main FSM indicates when each task (FSM) must start
    - each task (FSM) alerts to the main FSM when its task is finished

Complex FSMs

⇒ **Divide and conquer!**



Main FSM captures the top-level behavior

The whole problem is implemented in an unique FSM

Each FSM performs a specialized task

A mechanism is included to indicate when each task starts and stop

References

- S.A. Khan, Digital Design of Signal Processing Systems: A practical approach, Wiley 2011
- F. Vahid, Digital Design with RTL Design, and Verilog, Willey 2010