Coherent Functional, Electrical and Physical Modeling of IP Blocks using ALF

Invited Tutorial, CICC 2001

Wolfgang Roethig
Senior Engineering Manager
EDA R&D Group

NEC Electronics Inc.

Outline

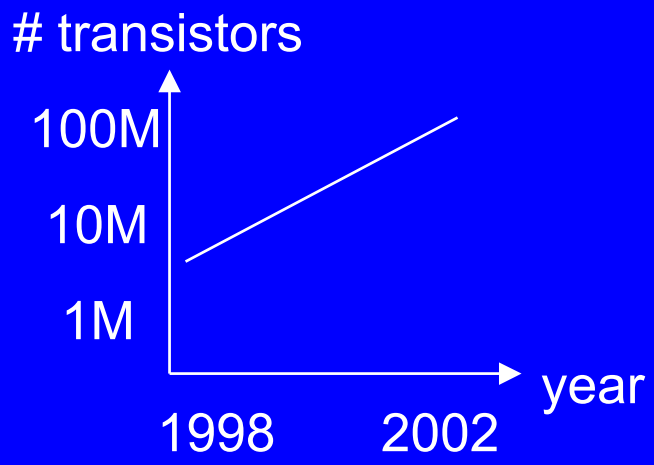
- Introduction
- Modeling Concepts
- Functional Modeling
- Electrical Modeling
- Physical Modeling
- Conclusion

Introduction

This tutorial gives
a comprehensive overview
of the Advanced Library Format (ALF)
and its role for System-on-Chip design
in UDSM technology

Introduction

Driving forces for integrated circuit technology

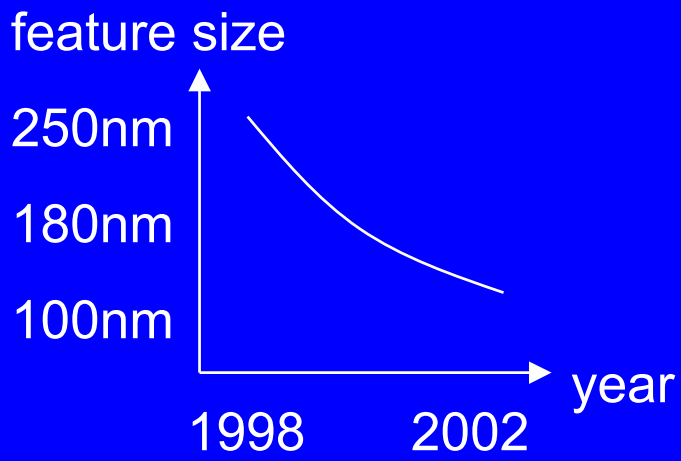


need more abstraction

Design complexity

Nanometer physics

need more detail



Trends in UDSM technology

- Design complexity
  - Hierarchical design
  - IP cores and macros as building blocks
- Nanometer physics
  - Signal integrity
  - Manufacturability
- Combination of both
  - High performance requirements
  - High power consumption
  - Significant voltage drop

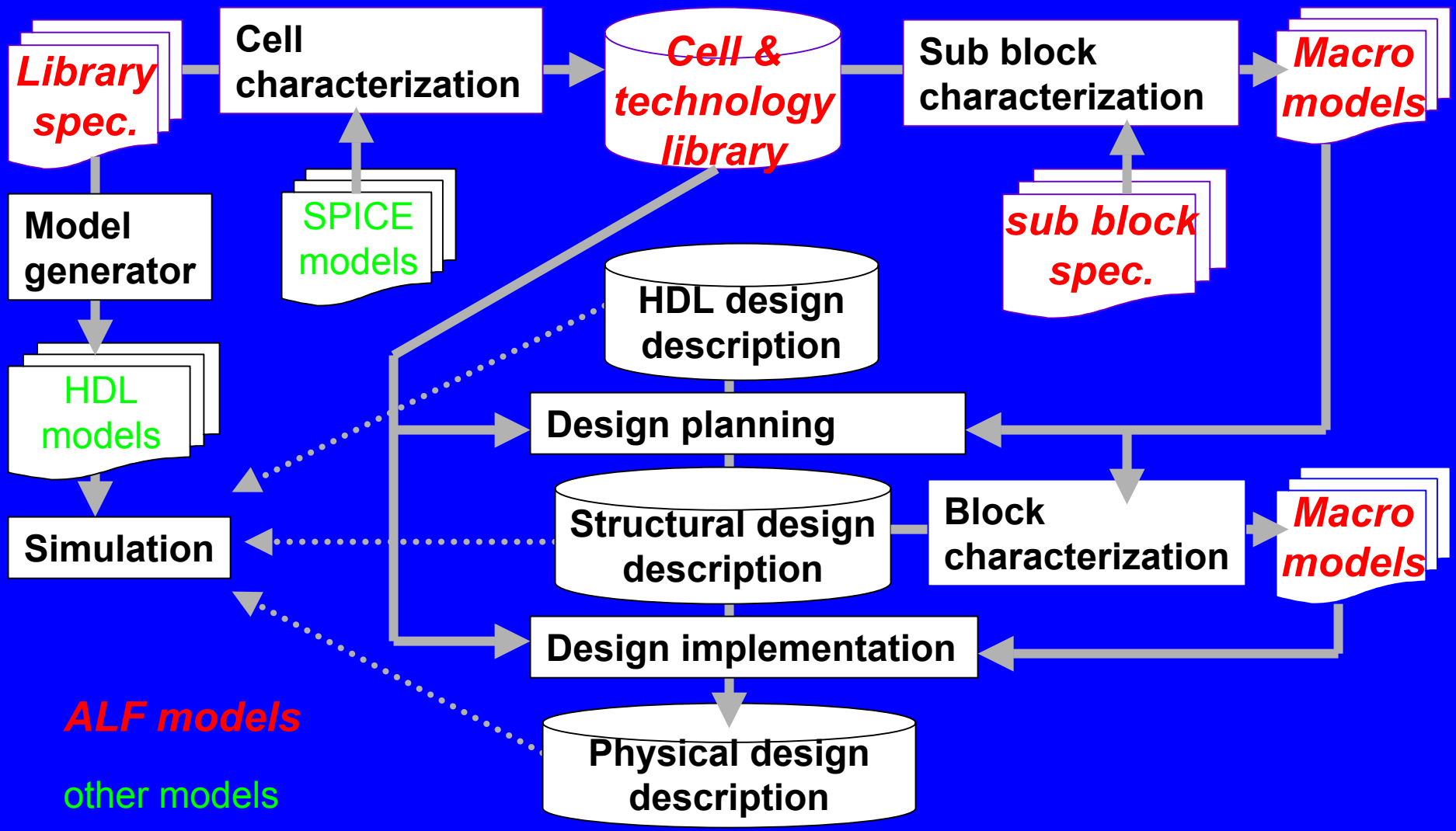
Accurate and efficient modeling required

Role of ALF in IP-based design

- Structured IP blocks
  - DSP, microprocessor core etc
  - Built out of sub-blocks
    - data path, register files, finite-state-machines
    - gates, latches, flip flops
- Monolithic IP blocks
  - RAM, CAM, PLL, SerDes, DDR etc
- ALF describes
  - model for each primitive building sub-block
  - abstract model of entire IP block

ALF is useful for all design implementation levels

Design flow with ALF



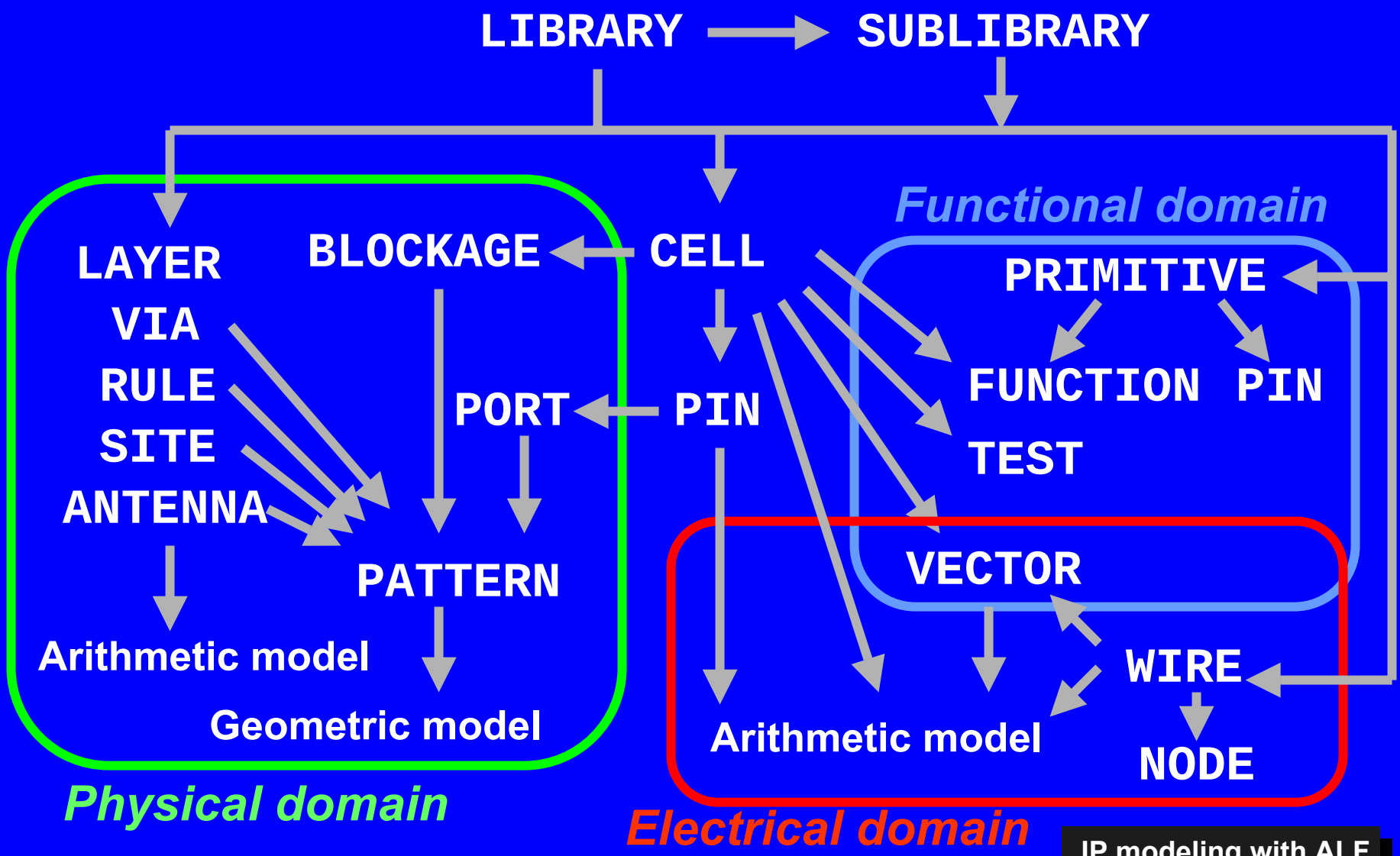
Design flow with ALF

- ALF library spec for virtual prototyping
  - scalable from cells to complex blocks
  - characterization up front or on the fly
  - target for all design planning and implementation tools
- Design planning
  - RTL partition, floor plan, pin assignment
  - power routing
- Design implementation
  - synthesis, layout
  - timing, power, signal integrity optimization

Modeling Concepts

- Support for Efficient Library Description
  - Object-oriented Library Representation
  - Context Sensitivity
  - Re-usable Definitions
- Support for Mathematical Calculation
  - Arithmetic Model
  - Geometric Model
- Support for Functional Specification
  - Boolean Expression
  - Vector Expression

Object-oriented Library Representation



Object-oriented Library Representation

- Fundamental objects common for all domains
  - LIBRARY, SUBLIBRARY, CELL, PIN
- Objects in functional domain
  - PRIMITIVE, FUNCTION, TEST, VECTOR
  - define logic behavior and test directives of circuit
- Objects in electrical domain
  - VECTOR, WIRE, NODE
  - define context for timing, power, signal integrity data
- Objects in physical domain
  - LAYER, VIA, RULE, SITE, ANTENNA, BLOCKAGE, PORT
  - define circuit and technology parameters for layout

Object-oriented Library Representation

- Principle of inheritance
  - definition within parent object is visible by child object
  - local definition within child object overrides global definition

```
LIBRARY my_lib {
        CAPACITANCE { UNIT=1e-12; }
        CELL cell1 {
                CAPACITANCE { UNIT=1e-15; }
        }
        CELL cell2 {
                PIN A { CAPACITANCE {UNIT=1e-9;} }
        }
}
```

Context Sensitivity

- Context-sensitive keyword has basic semantic meaning
- Context provides additional semantic meaning

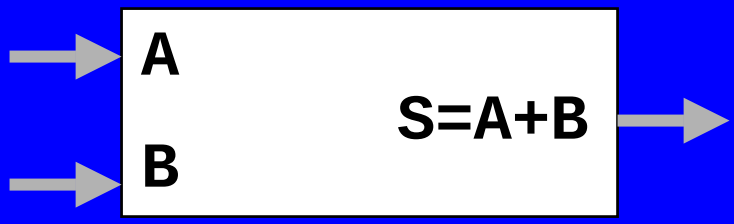
```
PIN A { CAPACITANCE=number; }

PIN B { LIMIT { CAPACITANCE { MAX=number; } }

WIRE wlm1 { CAPACITANCE = f(area,fanout) }

LAYER metal1 { CAPACITANCE = f(length,width) }

RULE rule1 { CAPACITANCE = f(pattern) }

DELAY { FROM {PIN=A;} TO {PIN=B;}
        HEADER { CAPACITANCE c1 {PIN=B;}
        EQUATION { k0 + k1*c1 }
}
```

Re-usable Definitions

- **TEMPLATE** statement provides re-usable definition with placeholders
- Template instance replaces placeholder with value
  - static template instance:
    all placeholders are replaced with constant values
  - dynamic template instance:
    some placeholders remain variables
- **GROUP** statement provides equivalent of multiple definitions within one definition
  - definitions containing a group identifier are interpreted as if they were replicated

Re-usable Definitions: TEMPLATE

- Example: N-bit adder
  - delay on critical path depends on bit width




```
TEMPLATE N_bit_adder {
      CELL <cellname> {
            PIN [ <bitwidth> :1] A { DIRECTION=input; }
            PIN [ <bitwidth> :1] B { DIRECTION=input; }
            PIN [ <bitwidth> :1] S { DIRECTION=output; }
      }
      DELAY = <critical_path_delay> {
            FROM { PIN = A[1]; }
            TO   { PIN = S[ <bitwidth> ]; }
}     }
```

Re-usable Definitions: TEMPLATE

- **Static** instantiation of N_bit_adder
  - adder with *fixed* bit width and critical path delay

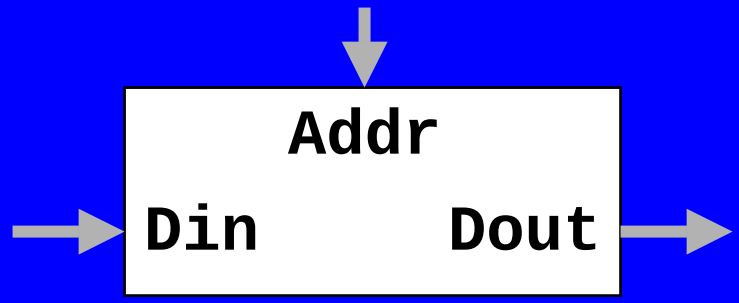
```
N_bit_adder = static {
      cellname = \4_bit_adder;
      bitwidth = 4;
      critical_path_delay = 1.35;
}
```

- **Dynamic** instantiation of N_bit_adder
  - adder with *variable* bit width and critical path delay

```
N_bit_adder = dynamic {
      critical_path_delay = 0.35 + bitwidth * 0.25;
}
```

Re-usable Definitions: GROUP

- Example: Memory with address bus and data I/O bus

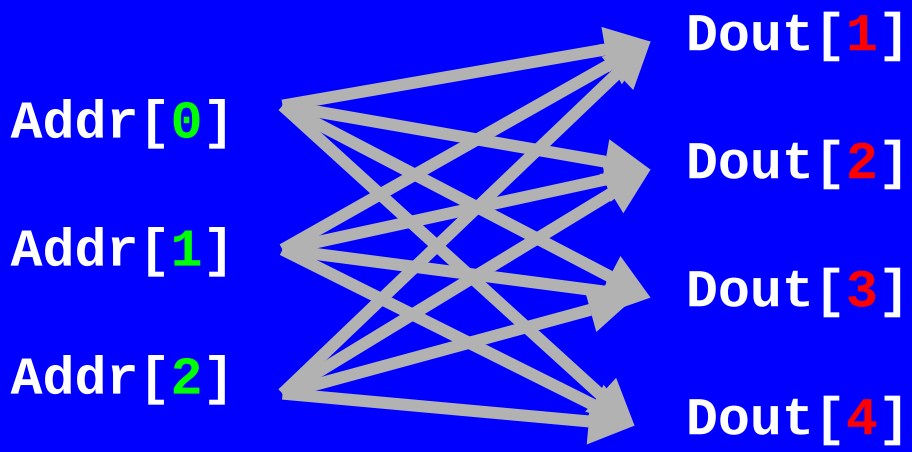



```
CELL my_memory {
        GROUP address_bits { 0 : 2 }
        GROUP data_bits { 1: 4 }
        PIN [0:2] Addr { DIRECTION = input; }
        PIN [1:4] Din  { DIRECTION = input; }
        PIN [1:4] Dout { DIRECTION = output; }
        // timing models (see next page)
}
```

Re-usable Definitions: GROUP

- timing arcs exist from each address bit to each data bit *(read mode)*

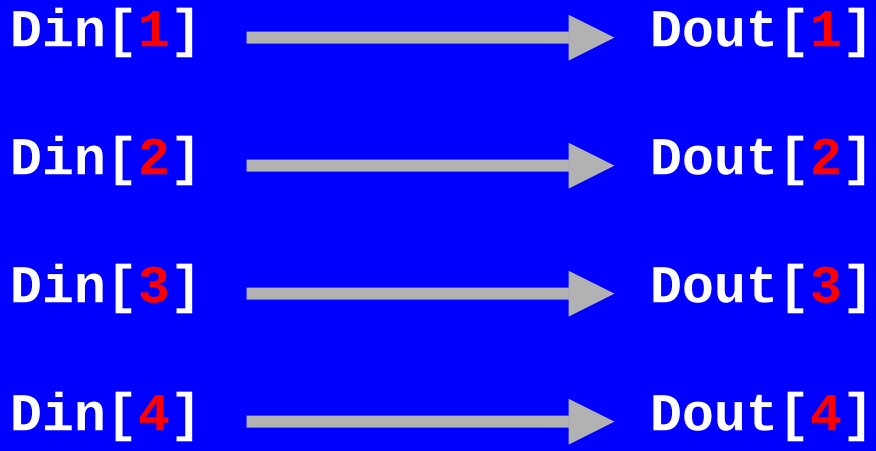
```
DELAY {
      FROM { PIN = Addr[address_bits] ; }
      TO   { PIN = Dout[data_bits] ; }
}
```



Re-usable Definitions: GROUP

- timing arcs exist bit wise from data input to data output *(write-through mode)*

```
DELAY {
        FROM { PIN = Din[data_bits] ; }
        TO   { PIN = Dout[data_bits] ; }
}
```



Modeling Concepts

- Support for Efficient Library Description
  - Object-oriented Library Representation
  - Context Sensitivity
  - Re-usable Definitions
- Support for Mathematical Calculation
  - Arithmetic Model
  - Geometric Model
- Support for Functional Specification
  - Boolean Expression
  - Vector Expression

Arithmetic Models

- Arithmetic models describe mathematical relationship between measurable quantities
  - use context-sensitive keywords
  - provide specification for calculation
- Examples for measurable quantities
  - CAPACITANCE, DELAY, AREA, ENERGY
- Complexity of arithmetic models
  - trivial model: constant number
  - equation: use standard mathematical operators
  - N-dimensional look up table: use interpolation
  - nested model: use model as argument for other model

Arithmetic Models

- Example for trivial arithmetic model

```
DELAY = 1.0 {
      FROM { PIN = A ; } TO { PIN = Y ; }
}
```

- Example for arithmetic model with trivial sub-models

```
DELAY {
      FROM { PIN = A ; } TO { PIN = Y ; }
      MIN = 0.8 ;
      TYP = 1.0 ;
      MAX = 1.3 ;
}
```

Arithmetic Models

- Example for equation-based arithmetic model

```
DELAY { FROM { PIN = A ; } TO { PIN = Y ; }
        HEADER {
                SLEWRATE   Ttrans { PIN = A; DEFAULT = 0.5; }
                CAPACITANCE Cload { PIN = Y; DEFAULT = 0.4; }
        }
        EQUATION { 0.1 + 1.2*Ttrans + 0.5*Cload }
}
```

- Model arguments are specified in HEADER
- `Ttrans` = slewrate (i.e. transition time) at (input) pin `A`
- `Cload` = load capacitance at (output) pin `Y`
- DEFAULT values are specified for `Ttrans` and `Cload`

Arithmetic Models

- Example for table-based arithmetic model

```
DELAY { FROM { PIN = A ; } TO { PIN = Y ; }
      HEADER { SLEWRATE   { PIN=A; INTERPOLATION=linear;
                    TABLE { 0.2 0.4 0.6 }
             } CAPACITANCE { PIN=Y; INTERPOLATION=linear;
                    TABLE { 0.25 0.5 }
      }      }
      TABLE {
             0.465 0.605 0.845
             0.490 0.730 0.970
}     }
```

- 1st argument (here SLEWRATE) defines innermost index of lookup table
- INTERPOLATION specifies how to calculate intermediate values

Geometric Models

- PATTERN can contain geometric models
- Geometric models describe shapes of physical objects
- Examples of geometric models
  - POLYGON, POLYLINE, RECTANGLE, LINE
- Examples of context
  - VIA, spacing or extraction RULE within LIBRARY
  - physical BLOCKAGE within CELL
  - physical PORT within PIN
- Geometric models can be subjected to transformations
  - SHIFT, FLIP, ROTATE, REPEAT

Modeling Concepts

- Support for Efficient Library Description
  - Object-oriented Library Representation
  - Context Sensitivity
  - Re-usable Definitions
- Support for Mathematical Calculation
  - Arithmetic Model
  - Geometric Model
- Support for Functional Specification
  - Boolean Expression
  - Vector Expression

Boolean Expressions

- Boolean expressions describe static relationships between logic variables
  - use standard logical operators (see IEEE 1364)
- Boolean expressions are used for specification of
  - combinatorial logic
  - triggering condition for state-sensitive sequential logic
  - condition for existence of electrical characterization or test stimulus in particular operation mode

Vector Expressions

- Vector expressions describe temporal change of logic variables
  - unary vector operators describe atomic events, i.e. transitions between states of logic variable
  - binary vector operators describe temporal order between events
- Vector expressions are used for specification of
  - triggering event for sequential logic
  - stimulus for electrical characterization and test

Vector Expressions

- Value system for vector expressions (subset applies also for boolean expressions

Logic values

strength ←

strong	weak	none	*value* ↓
1	H		true
X	W	Z	unknown
0	L		false

Integer values

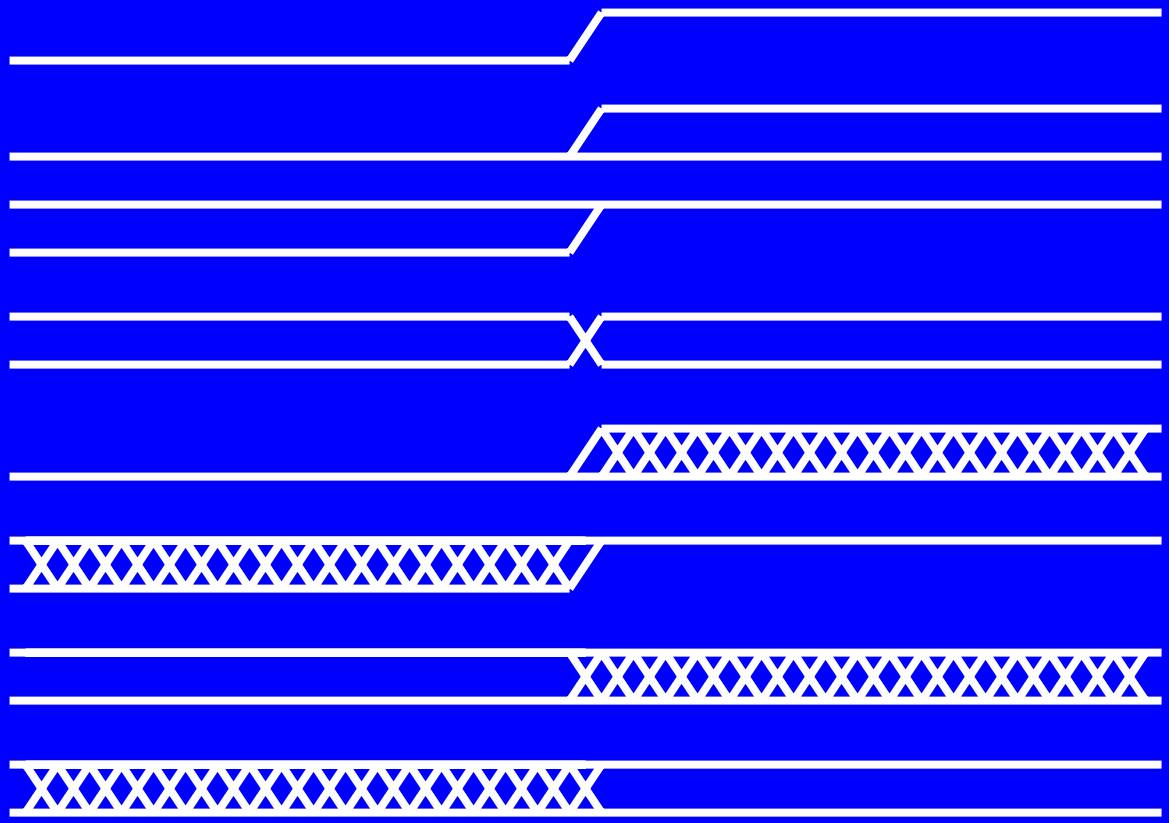
prefix	*base*
'b	binary
'o	octal
'd	decimal
'h	hexadecimal

Symbolic values

symbol	*meaning*
?	don't care
U	uninitialized
*	event is not monitored
?-	no change
?!	any change
?~	all bits change

Vector Expressions

- **Examples for atomic events on logic variable “A”**

Vector expression	*Timing diagram*
(01 A)	
(0? A)	
(?1 A)	
(?! A)	
(0* A)	
(*1 A)	
(?* A)	
(*? A)	

Vector Expressions

- **Temporal order in vector expressions**

`event 1 -> event 2`	*event 1* immediately followed by *event 2*
`event 1 ~> event 2`	*event 1* eventually followed by *event 2*
`event 1 & event 2`	*event 1* and *event 2* simultaneously
`event 1 | event 2`	*event 1* or *event 2* alternatively
`event 1 <-> event 2`	*event 1* and *event 2* in either order
equivalent form	`event 1 -> event 2 | event 2 -> event 1`
`event & boolean expr.`	conditional *event*
Expression with more events	`event 1 -> event 2 -> event 3`

Note: events are self-timed
event queue advances, whenever next event arrives

Vector Expressions

- **Examples for event sequences on logic variables “A”, “Y”**

Vector expression	*Timing diagram*
`(01 A -> 01 Y)`	
`(01 A ~> 01 Y)`	
`(01 A -> 10 A)`	
`(01 A ~> 10 A)`	

Functional Modeling

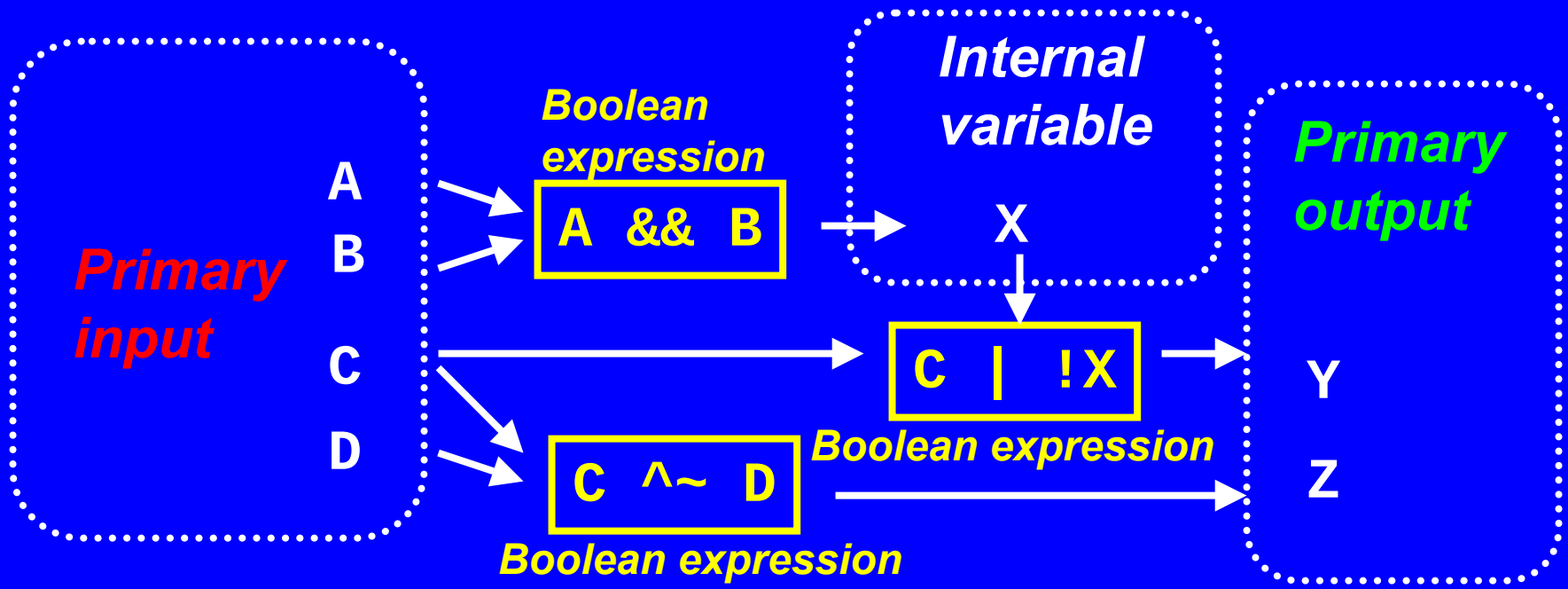
- Combinatorial logic circuits
- Sequential logic circuits
- Structural modeling
- Modeling for test

Purpose of Functional Modeling

- Canonical specification of function
- Useful for cell design and characterization tools
  - inference of characterization vectors (timing, power)
  - formal verification (transistor versus spec)
- Useful for downstream tools
  - Library primitives:
    directly used by synthesis and DFT tools
  - Hard macros:
    generate simulation models for different targets
    (cycle-based, full timing, Verilog, VHDL)
  - Soft macros:
    generate HDL code and scripts for synthesis

Combinatorial Logic Circuit

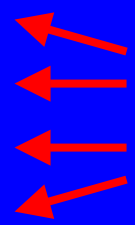
- Example in function graph representation



Combinatorial Logic Circuit

- **Example in ALF language**

```
CELL my_combinational_circuit {
      PIN A { DIRECTION=input; }
      PIN B { DIRECTION=input; }
      PIN C { DIRECTION=input; }
      PIN D { DIRECTION=input; }
      PIN X { DIRECTION=none; }
      PIN Y { DIRECTION=output;}
      PIN Z { DIRECTION=output;}
      FUNCTION {
            BEHAVIOR {
                  X = A && B ;
                  Y = C | !X ;
                  Z = C ^~ D ;
            }
      }
}
```



Primary input

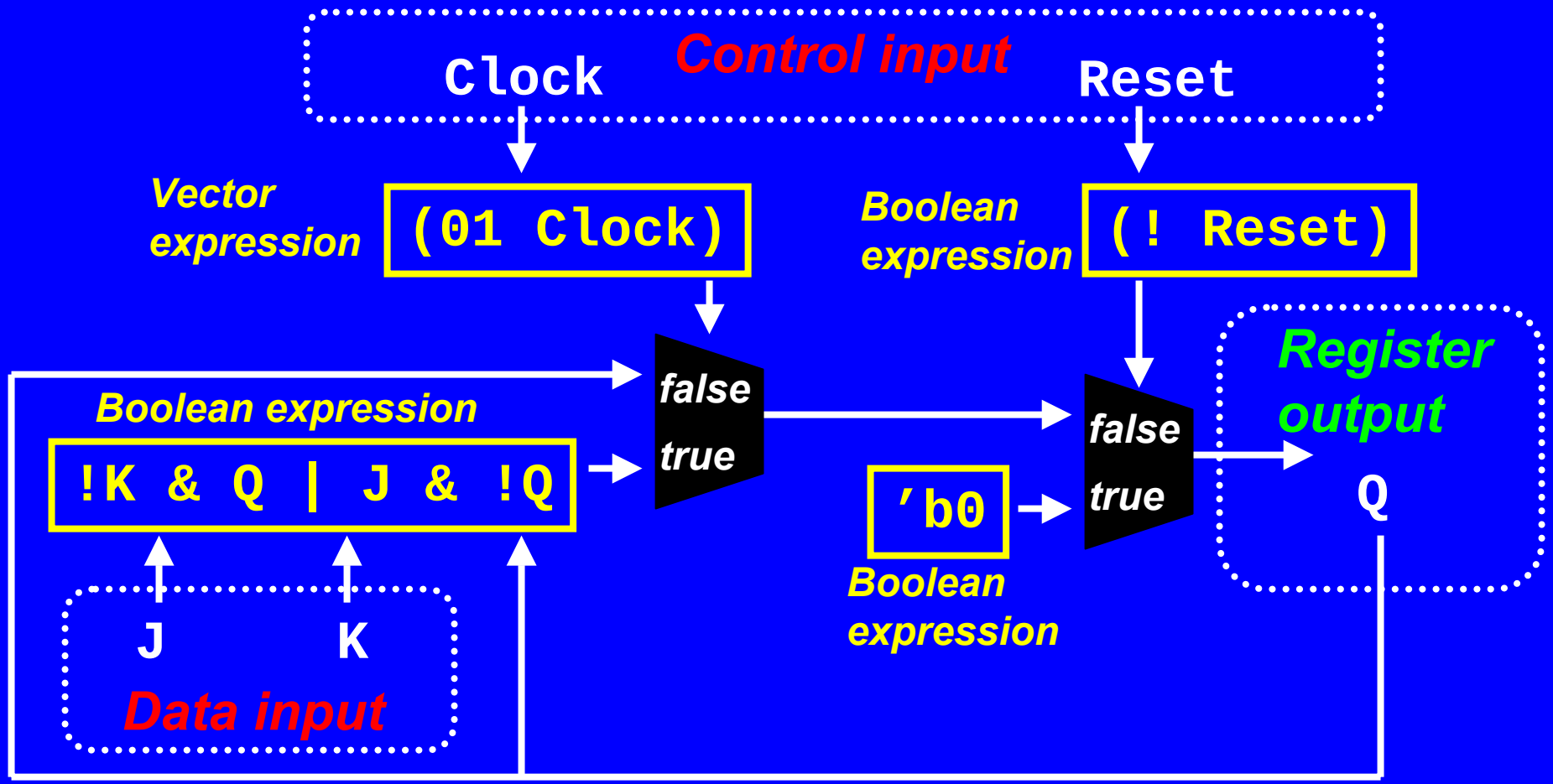
Internal node

Primary output

*Assignments of **boolean expressions** to logic variables*

Sequential Logic Circuit

- Example in function graph representation



Sequential Logic Circuit

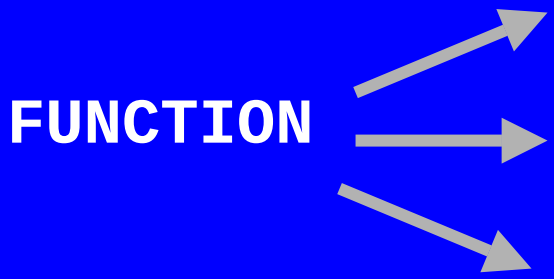
- **Example in ALF language**

```
CELL my_JK_flipflop {
      PIN J { DIRECTION=input; SIGNALTYPE=data; }
      PIN K { DIRECTION=input; SIGNALTYPE=data; }
      PIN Clock { DIRECTION=input;
        SIGNALTYPE=clock; POLARITY=rising_edge; }
      PIN Reset { DIRECTION=input; ACTION=asynchronous;
        SIGNALTYPE=clear; POLARITY=low; }
      PIN Q { DIRECTION=output; SIGNALTYPE=data; }
      FUNCTION {
            BEHAVIOR {
higher priority "if"  ---> @ (! Reset) {
                            Q = 'b0 ;
lower priority "else if" ---> } : (01 Clock) {
                            Q = !K & Q | J & !Q ;
  }     }     }     }
```

descriptive qualifiers

Items related to Functional Modeling

- **FUNCTION supports BEHAVIOR, STATETABLE, STRUCTURE**



Canonical description of functional behavior

Similar to Verilog UDP, but more compact

Structure for DFT and physical implementation

- **BEHAVIOR statement can also be used in other context**

VIOLATION → BEHAVIOR

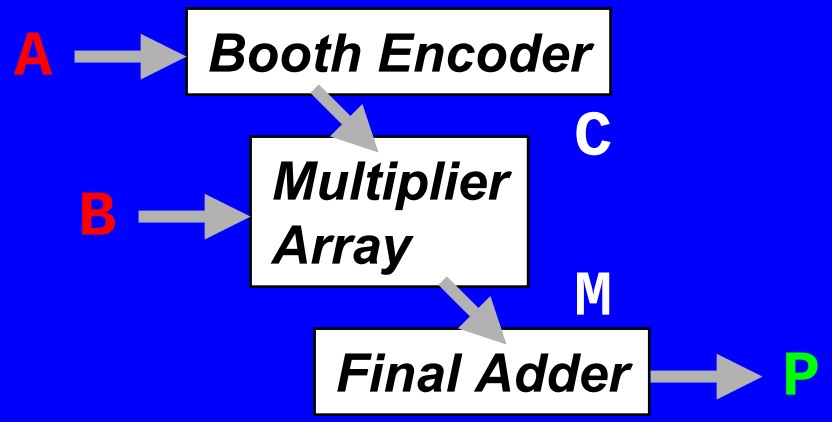
Behavior in case of illegal operation or timing violation

TEST → BEHAVIOR

Wrapper for test, especially BIST

STRUCTURE statement

- STRUCTURE specifies sub-circuits and internal nodes
- Example: multiplier




```
CELL my_unsigned_multiplier {
      PIN [32:1] A { DIRECTION=input; DATATYPE=unsigned; }
      PIN [32:1] B { DIRECTION=input; DATATYPE=unsigned; }
      PIN [32:1] P { DIRECTION=output; DATATYPE=unsigned;}
      FUNCTION { BEHAVIOR { P = A * B ; }
            STRUCTURE {
                  booth_encoder U1 { Bin=A; Bout=C; }
                  multiplier U2 { op1=C; op2=B; out=M; }
                  adder U3 { A=M[0:31]; B=M[32:63]; S=P; }
}     }     }
```

Instances of cells

TEST statement

- BEHAVIOR in context of TEST statement
  - describes interface between test algorithm and DUT
- Circuit overhead for Built-In-Self-Test depends on
  - Test algorithm (owned by BIST tool)
  - Interface between test algorithm and DUT (specified by BEHAVIOR in TEST context)
- Example: RAM with BIST
  - BIST must access memory cells in contiguous order
  - Logical to physical address mapping required

TEST statement

```
CELL my_RAM  {
      PIN [1:0] A { DIRECTION=input; SIGNALTYPE=address; }
      PIN [8:1] D { DIRECTION=both;  SIGNALTYPE=data; }
      PIN RW { DIRECTION=input; SIGNALTYPE=control;}
      PIN [8:1] M [0:3] { SIGNALTYPE=data;
            DIRECTION=none; VIEW=none; }
      PIN [1:0] R { ATTRIBUTE { ROW }
            DIRECTION=input; VIEW=none; }
      FUNCTION {
            BEHAVIOR {        write mode     read mode
                  @(!RW) { M[A]=D; } : { D=M[A]; }
      }     }
      TEST { BEHAVIOR {
                  A[0] = R[0]^R[1];
                  A[1] = !R[1];
}     }     }
```

Virtual pins

R[1:0]	A[1:0]
0 = 'b00	10
1 = 'b01	11
2 = 'b10	01
3 = 'b11	00

Mapping between row index and address bits

Electrical Modeling

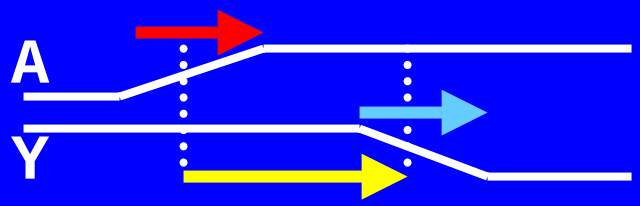
- Timing
- Power
- Signal Integrity
  - Noise
- Reliability
  - Electromigration
  - Hot electron effect

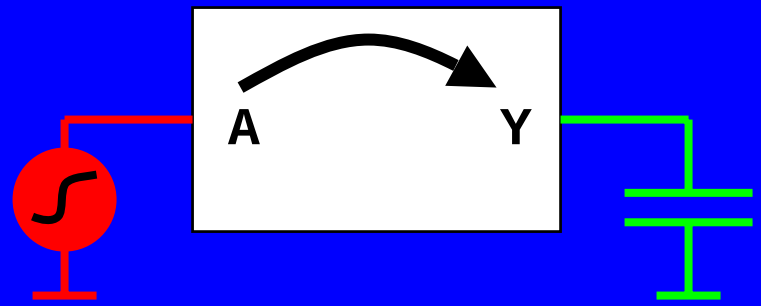
Timing

- Timing models in context of VECTOR
  - reference point of measurement must appear in vector expression
- 1 point measurement (at least 1-event VECTOR)
  - SLEWRATE (1 transition on a pin)
  - incremental DELAY (only FROM or TO defined)
- 2 point measurement (at least 2-event VECTOR)
  - absolute DELAY (both FROM and TO defined)
  - SETUP, HOLD, RECOVERY, REMOVAL
- multiple 2 point measurements (N-event VECTOR)
  - combined SETUP and HOLD, RETAIN and DELAY
  - DELAY for multiple switching inputs

Timing modeling

Timing model for primitive cell



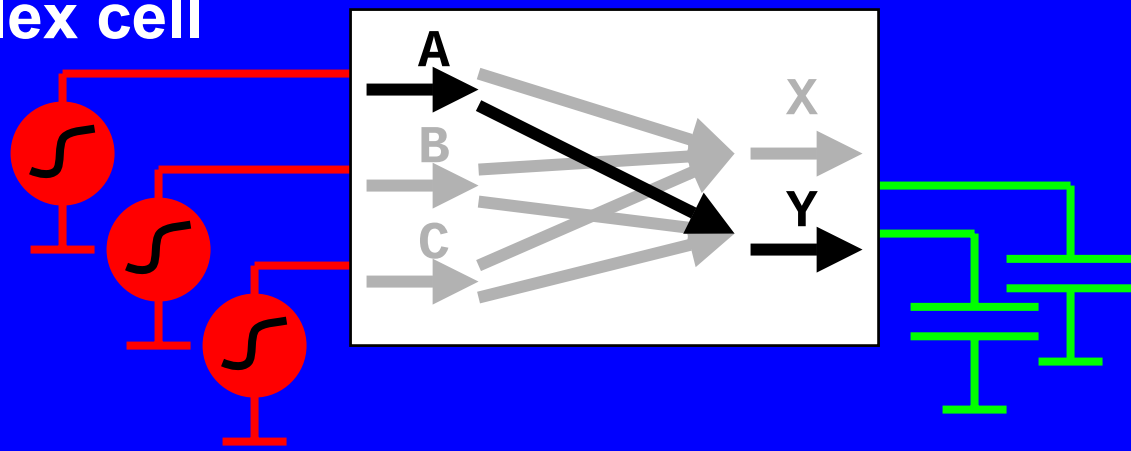



```
VECTOR (01 A -> 10 Y) {
      DELAY { FROM { PIN=A; } TO { PIN=Y; }
            HEADER {
                  CAPACITANCE { PIN=Y; TABLE { xxxx }
                  SLEWRATE { PIN=A; TABLE { xxx }
            } TABLE { xxxx xxxx xxxx }
      }
      SLEWRATE { PIN=Y;
            HEADER {
                  CAPACITANCE { PIN=Y; TABLE { xxxx }
                  SLEWRATE { PIN=A; TABLE { xxx }
            } TABLE { xxxx xxxx xxxx }
}     }
```

Timing modeling

Timing model for complex cell

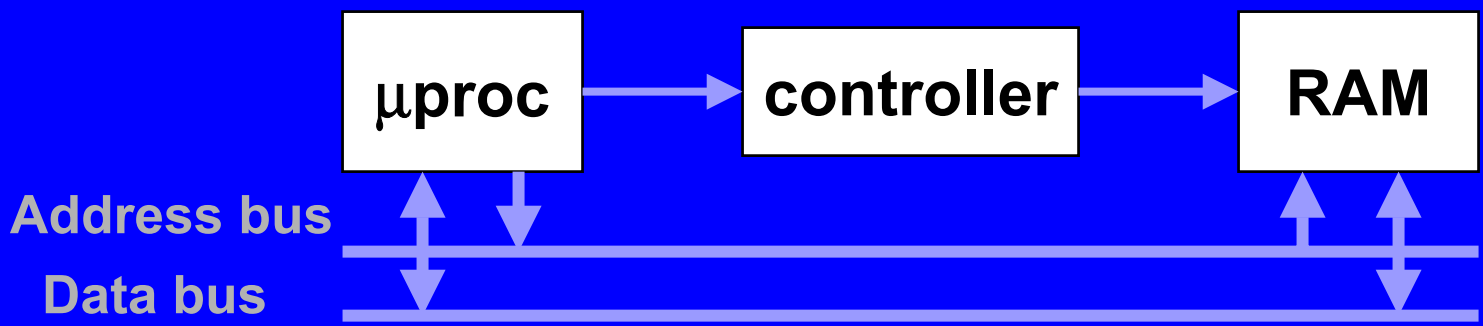
- *Constant delay across arc*
- *Slew and load dependency only at input and output pin*
- *Shared among many arcs*




```
VECTOR(01 A -> 10 Y){
      DELAY=1.2 { FROM{PIN=A;} TO{PIN=Y;} }}
VECTOR(01 A){
      DELAY { FROM{PIN=A;} CALCULATION=incremental;
            HEADER  { SLEWRATE {PIN=A;} }
            EQUATION{ 0.5 + 0.7*SLEWRATE }}}
VECTOR(10 Y){
      DELAY { TO{PIN=Y;} CALCULATION=incremental;
            HEADER  {CAPACITANCE {PIN=Y;} }
            EQUATION{ 0.2 + 0.3*CAPACITANCE }}}
```

Timing Macro Modeling

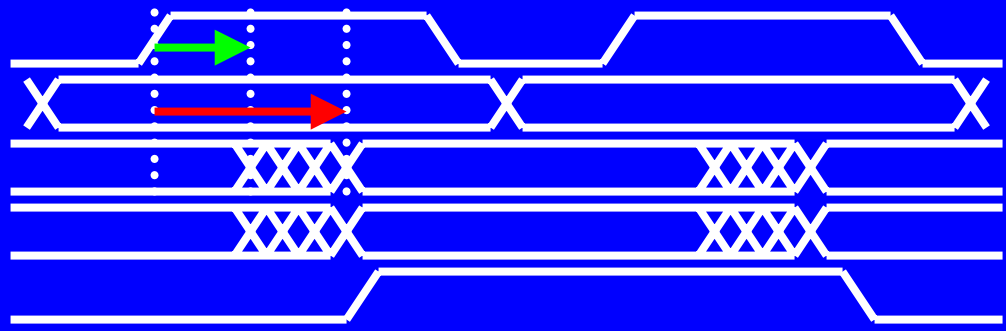
- N-event VECTOR
  - More than one event may happen on a pin
  - Multiple events are counted by EDGE_NUMBER
  - count starts at 0 for each pin
- Purpose of N-event VECTOR
  - describe timing models in the context of complex timing diagram
  - useful for interface timing between blocks



Timing Macro Modeling

Microprocessor decodes instruction and initiates write operation to memory

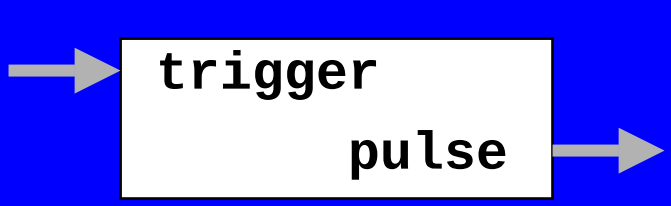
clock
instr
addr
data
writeData2RAM

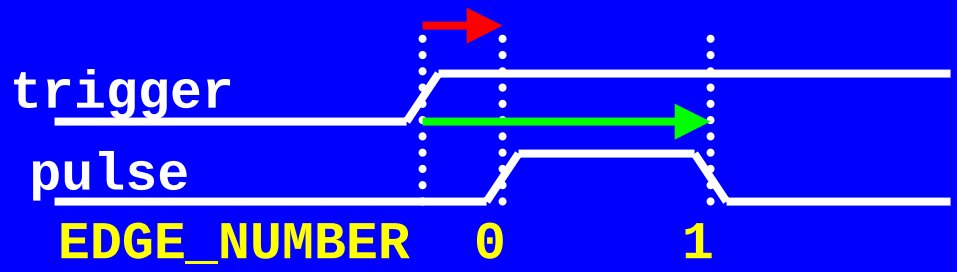


```
CELL my_microprocessor {
      PIN clock          { DIRECTION = input;  }
      PIN [15:0] instr   { DIRECTION = input;  }
      PIN [7:0]  addr    { DIRECTION = output; }
      PIN [31:0] data    { DIRECTION = both;   }
      PIN writeData2RAM  { DIRECTION = output; }
      VECTOR (01 clock -> ?* addr -> *? addr) {
            RETAIN { FROM {PIN=clock;} TO {PIN=addr;} }
            DELAY  { FROM {PIN=clock;} TO {PIN=addr;} }
}     }
```

Timing Macro Modeling

RAM controller generates write clock

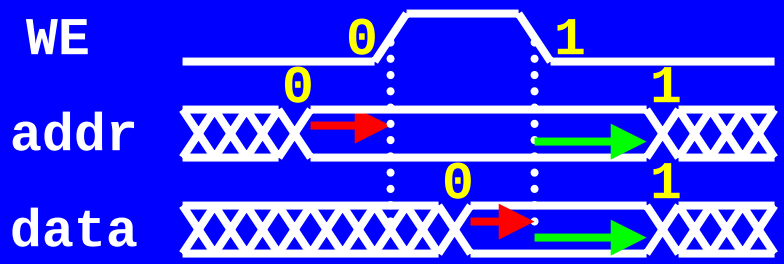





```
CELL my_RAM_controller {
      PIN trigger { DIRECTION = input; }
      PIN pulse   { DIRECTION = output; }
      VECTOR (01 trigger -> 01 pulse -> 10 pulse) {
            DELAY d1 = 0.2 { FROM { PIN=trigger; }
                  TO { PIN=pulse; EDGE_NUMBER=0; }
            }
            DELAY d2 = 1.2 { FROM { PIN=trigger; }
                  TO { PIN=pulse; EDGE_NUMBER=1; }
}     }     }
```

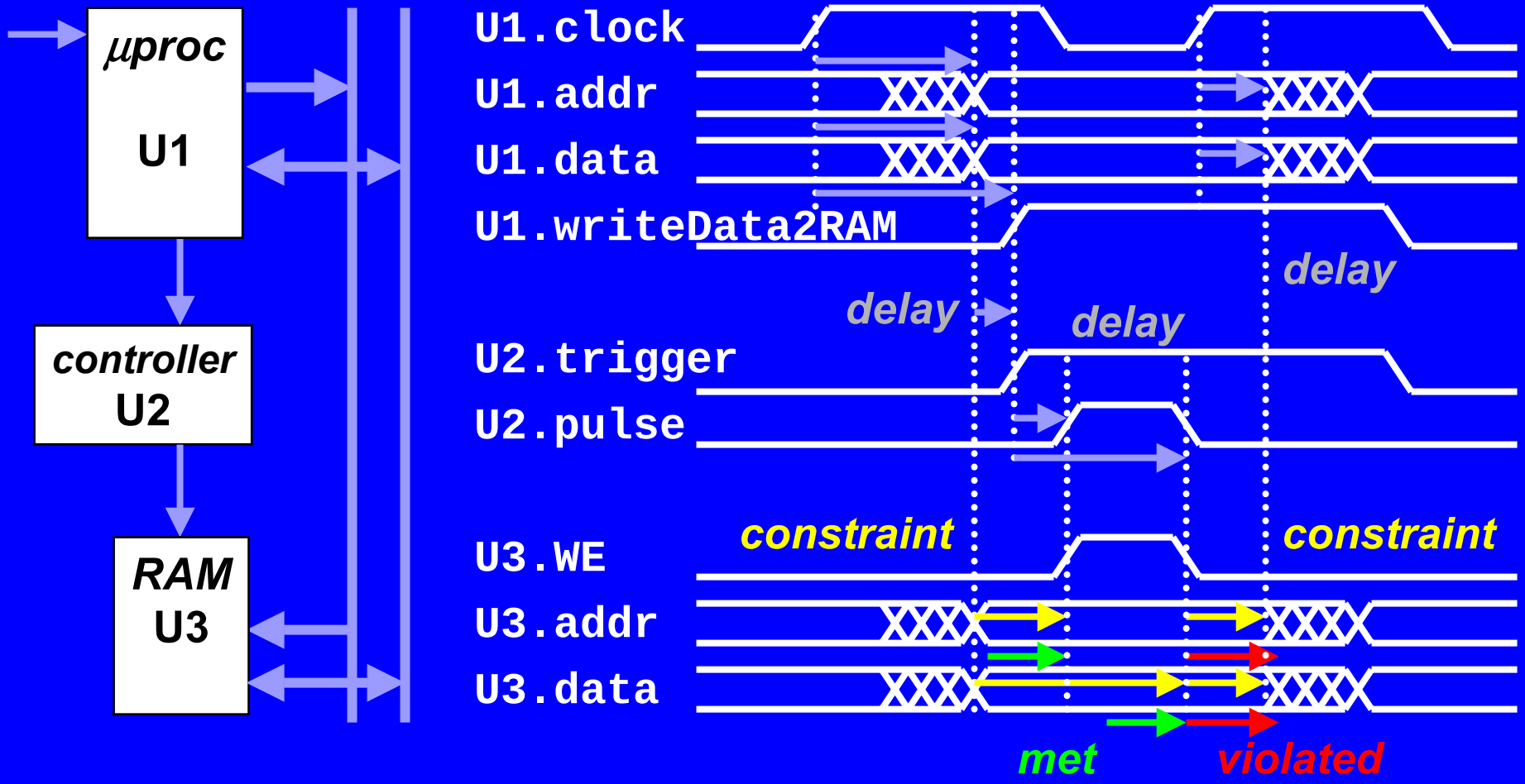
Timing Macro Modeling

RAM timing for write operation




```
CELL my_RAM {
      PIN WE { DIRECTION = input; }
      PIN [7:0]  addr { DIRECTION = input; }
      PIN [31:0] data { DIRECTION = both;   }
      VECTOR (*? addr -> 01 WE -> *? data
             -> 10 WE -> ?* addr -> ?* data)
             { SETUP t1 { FROM { PIN=addr; EDGE_NUMBER=0; }
                        TO   { PIN=WE;   EDGE_NUMBER=0; }
             } HOLD  t2 { FROM { PIN=WE;   EDGE_NUMBER=1; }
                        TO   { PIN=addr; EDGE_NUMBER=1; }
             } SETUP t3 { FROM { PIN=data; EDGE_NUMBER=0; }
                        TO   { PIN=WE;   EDGE_NUMBER=1; }
             } HOLD  t4 { FROM { PIN=WE;   EDGE_NUMBER=1; }
                        TO   { PIN=data; EDGE_NUMBER=1; }
}     }     }
```

Timing Analysis

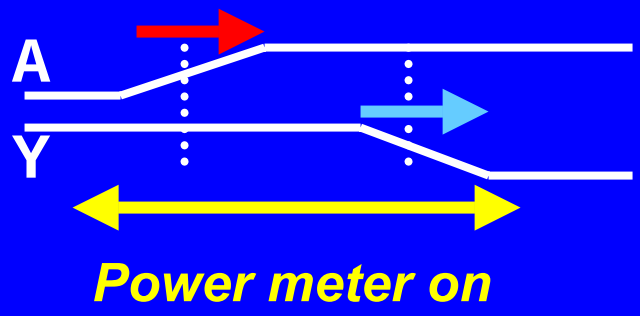


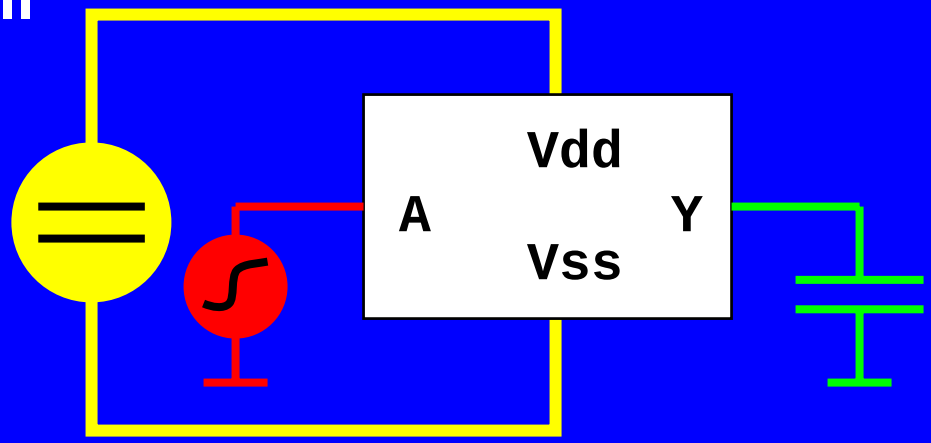
Power

- Power model in context of VECTOR
- Transient ENERGY
  - Vector expression describes sequence of events related to energy consumption
  - Can use same vector as for timing or extra vector
- Average POWER
  - with specified measurement time
  - transient energy = average power * measurement time
- Static POWER
  - Boolean expression describes static state related to power consumption

Power Modeling

Power model for primitive cell



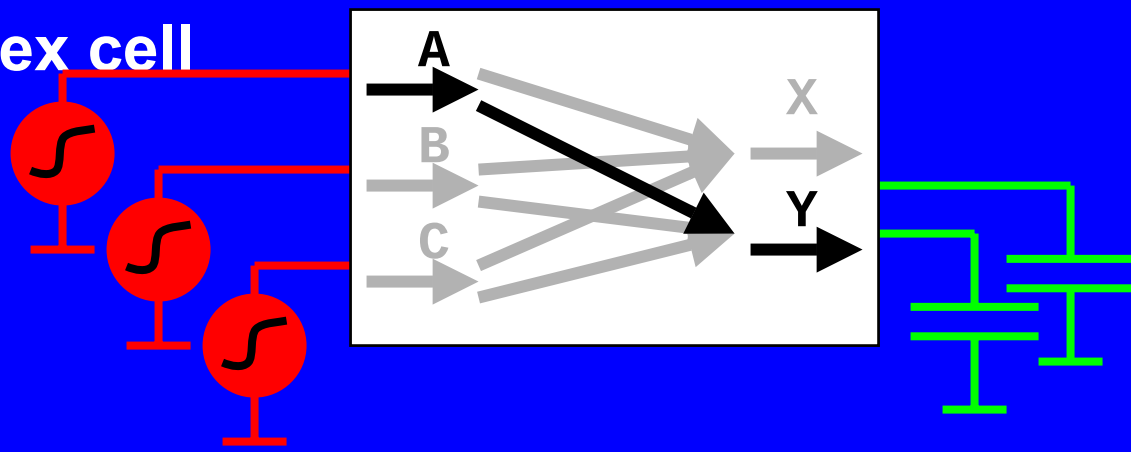



```
VECTOR (01 A -> 10 Y) {
      ENERGY { MEASUREMENT = transient;
            HEADER {
                  CAPACITANCE { PIN=Y; TABLE { xxxx }
                  SLEWRATE { PIN=A; TABLE { xxx }
            } TABLE { xxxx xxxx xxxx }
      }
}
```

Power Modeling

Power model for complex cell

- *Model slewrate and load depedency with 1-event vector at input and output*
- *Use N-event vectors for internal power*




```
VECTOR ( 01 A -> 01 Y & (B & ! C) )
       { ENERGY = 5.3; }
VECTOR( 01 A )
       { ENERGY {
            HEADER  { SLEWRATE {PIN=A;} }
            EQUATION{ 0.1 + 0.4*SLEWRATE }}}
VECTOR( 01 Y )
       { ENERGY {
            HEADER  {CAPACITANCE {PIN=Y;} }
            EQUATION{ 0.3 + 0.5*CAPACITANCE }}}
```

State dependency

Power Macro Modeling

Power model for bus

- Use ?! operator for transition on bus
- Use SWITCHING_BITS as argument for power model

```
CELL my_memory {
      PIN [0:2] Addr { DIRECTION = input; }
      PIN [1:4] Dout { DIRECTION = output; }
      VECTOR (?! Addr -> ?! Dout ) {
            ENERGY {
                  HEADER {
                        SWITCHING_BITS sba { PIN=Addr; }
                        SWITCHING_BITS sbd { PIN=Dout; }
                  } EQUATION {
                        0.4 + 0.2*LOG(sba) + 0.8*sbd
}     }     }     }
```

Power Macro Modeling

Example: asynchronous RAM

```
CELL my_RAM {
      PIN WE { DIRECTION = input; }
      PIN [7:0]  Addr { DIRECTION = input; }
      PIN [31:0] Din { DIRECTION = input;   }
      PIN [31:0] Dout { DIRECTION = output;   }
      VECTOR ( WE )
        { POWER { MEASUREMENT=static; }}
      VECTOR ( ?! Addr -> ?! Dout )
        { ENERGY {MEASUREMENT=transient;}}
      VECTOR ( 01 WE -> ?! Dout )
        { ENERGY {MEASUREMENT=transient;}}
      VECTOR ( ?! Din -> ?! Dout )
        { ENERGY {MEASUREMENT=transient;}}
      VECTOR ( ?! Din & ! WE )
        { ENERGY {MEASUREMENT=transient;}}
}
```

static power during write operation

read operation

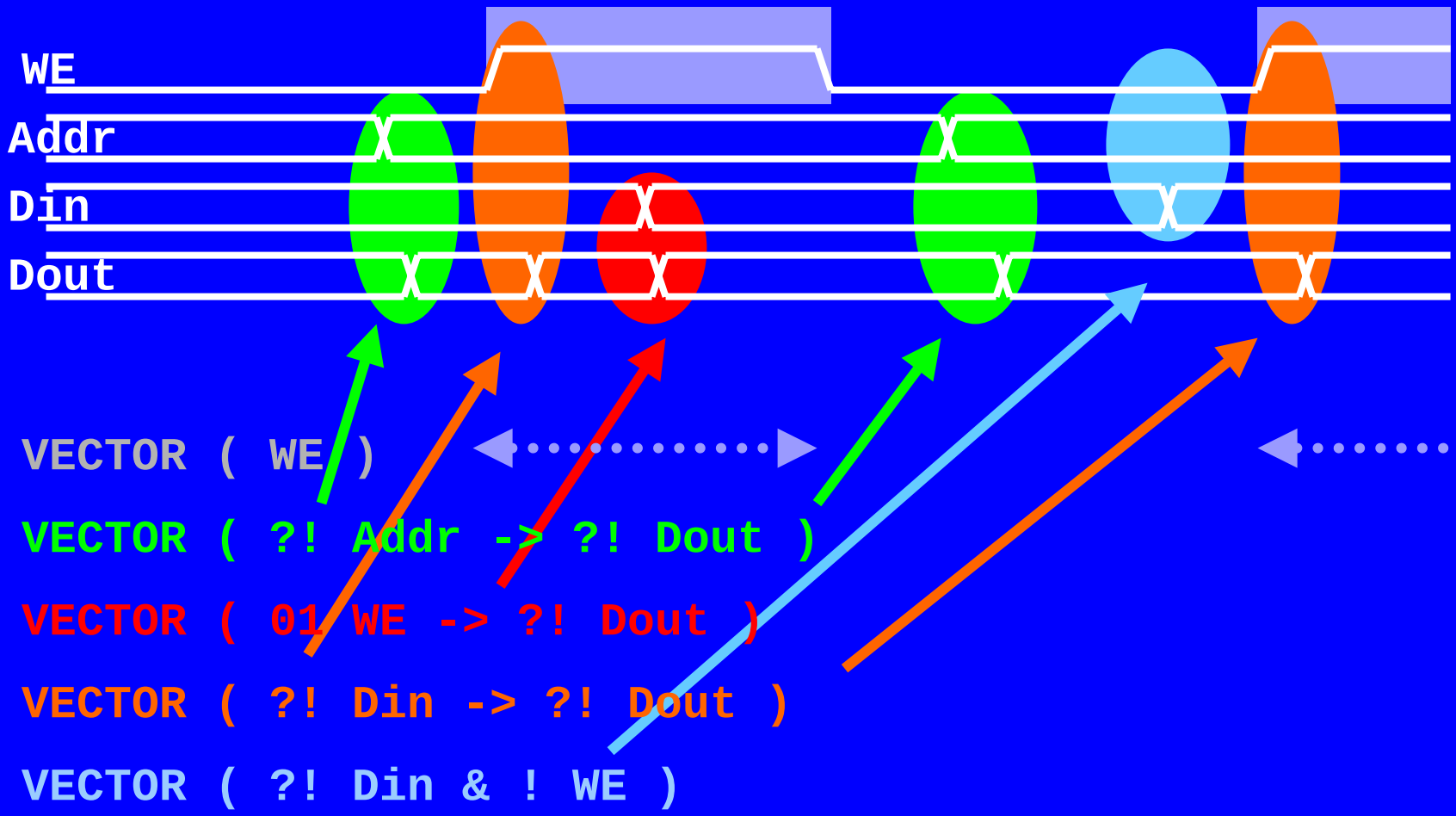
write operation with transparent output

write operation with data change

data change outside write operation

Power Analysis

Power vector monitoring during simulation

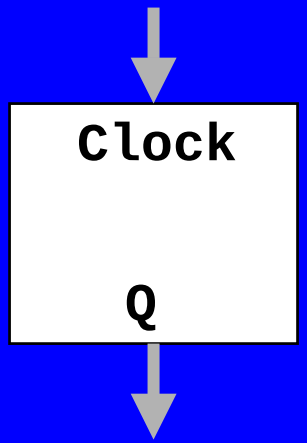


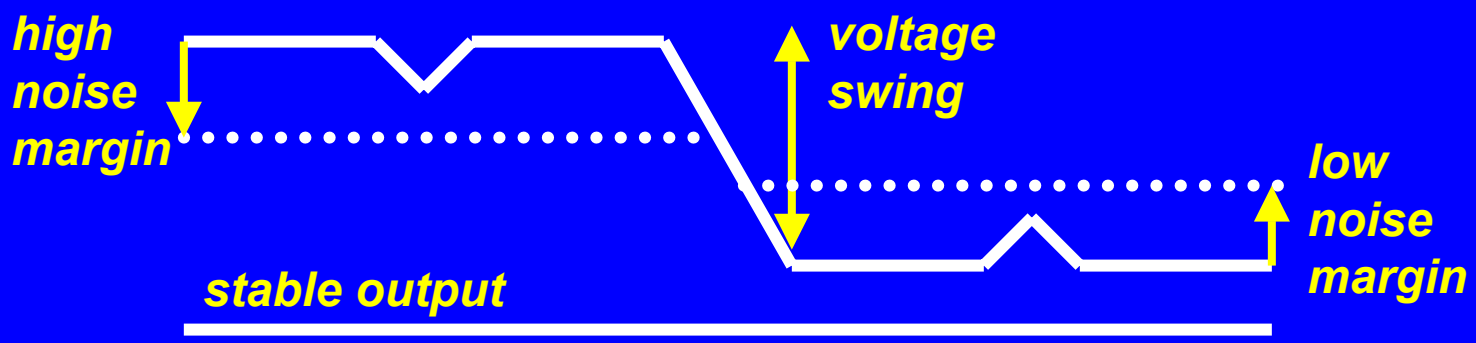
Noise

- Noise is voltage normalized to signal voltage swing
  - Difference between actual and nominal signal level
  - Noise margin is the limit for tolerated noise
- Static noise margin can be defined at input pin
  - results in pessimistic noise analysis
- Noise model in context of VECTOR
  - state-dependent noise margin
  - time-window dependent noise margin
  - noise rejection
  - noise propagation

Noise Modeling

- Static noise margin at clock input pin guarantees that false switching of flip flop will not happen

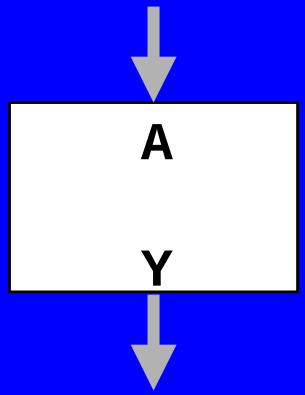


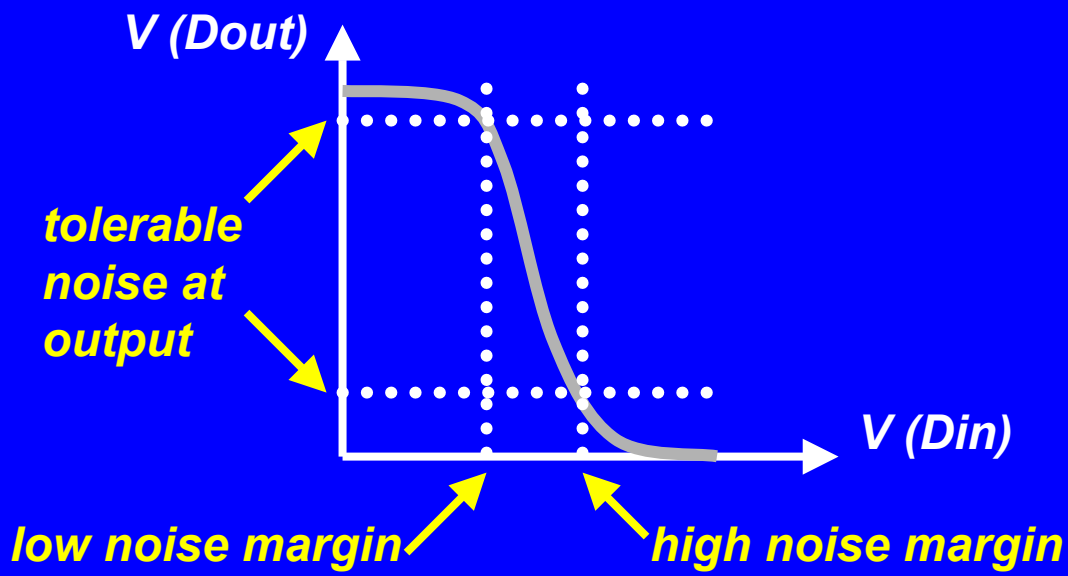



```
CELL my_flipflop {
       PIN Clock { DIRECTION=input;
             SIGNALTYPE=clock; POLARITY=rising_edge;
             NOISE_MARGIN { HIGH=0.4; LOW=0.2; }
       }
       PIN Q { DIRECTION=output; SIGNALTYPE=data; }
}
```

Noise Modeling

- Static noise margin at data input pin guarantees that noise at output of combinatorial cell is within bounds

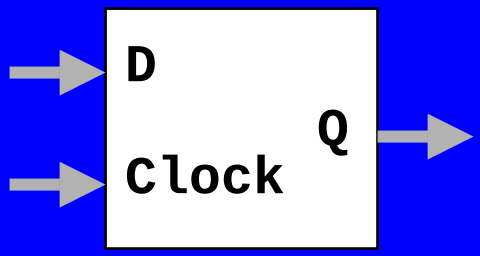


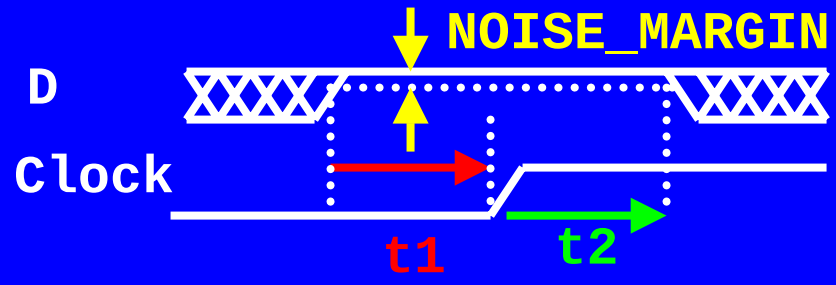



```
CELL my_combinatorial_cell {
      PIN A { DIRECTION=input;
              NOISE_MARGIN { HIGH=0.4; LOW=0.3; } }
      PIN Y { DIRECTION=output;
               NOISE { HIGH=0.05; LOW=0.15; } } }
```

Noise Modeling

- Data input of flip flop is sensitive to noise during time window around active clock edge

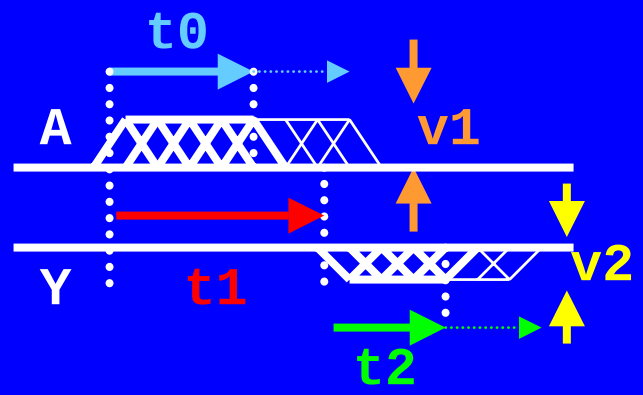





```
VECTOR ( *1 D -> 01 CLK -> 1*D ) {
      NOISE_MARGIN = 0.3 { PIN = D; }
      LIMIT { TIME t1 { MIN = number ;
                  FROM { PIN=D; EDGE_NUMBER=0; }
                  TO { PIN=CLK; EDGE_NUMBER=0; }
            } TIME t2 { MIN = number ;
                  FROM { PIN=CLK; EDGE_NUMBER=0; }
                  TO { PIN=D; EDGE_NUMBER=1; }
}     }     }
```

Noise Modeling

- Noise propagation as alternative to static noise margin for combinatorial cells



  - Output noise is characterized by
    - peak voltage
    - delay relative to input noise
    - pulse width
  - Output peak voltage, delay, pulse width depend on
    - input peak voltage
    - input pulse width
    - output load capacitance

Noise Modeling

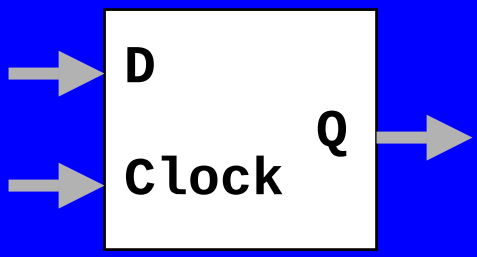
- **Example for noise propagation in ALF language**

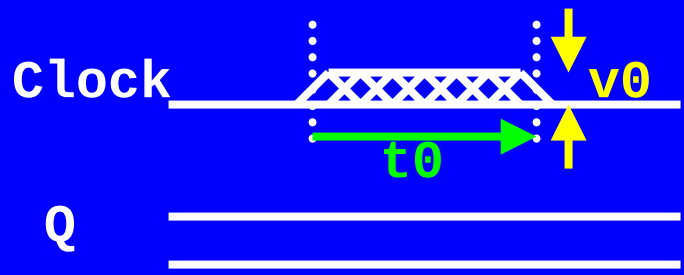
```
VECTOR(0* A -> *0 A <&> 1* Y -> *1 Y)
{ NOISE v2 { PIN=Y;
    HEADER {
      CAPACITANCE c1 { PIN=Y; }
      NOISE v1 { PIN=A; }
      TIME t0 {
        FROM { PIN=A; EDGE_NUMBER=0; }
        TO { PIN=A; EDGE_NUMBER=1; } }
    } EQUATION { 0.7*v1/(0.4+t0)/c1 } }
  DELAY t1 {
    FROM {PIN=A; EDGE_NUMBER=0;} TO {PIN=Y; EDGE_NUMBER=0;}
  /* same HEADER, different EQUATION */ }
  TIME t2 {
    FROM {PIN=Y; EDGE_NUMBER=0;} TO {PIN=Y; EDGE_NUMBER=1;}
  /* same HEADER, different EQUATION */ } }
```

t0 can be larger or smaller than t1

Noise Modeling

- Noise rejection as alternative to static noise margin for clock pin of flip flop





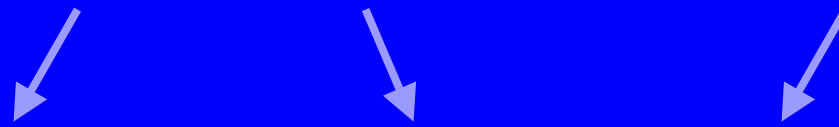
- Noise rejection is characterized by
  - maximum tolerated peak voltage
  - dependent on pulse width
- Could also be characterized for combinatorial cells
  - additional dependency on output load capacitance

Noise Modeling

- **Example for noise rejection in ALF language**

Input pulse eventually exceeds noise margin

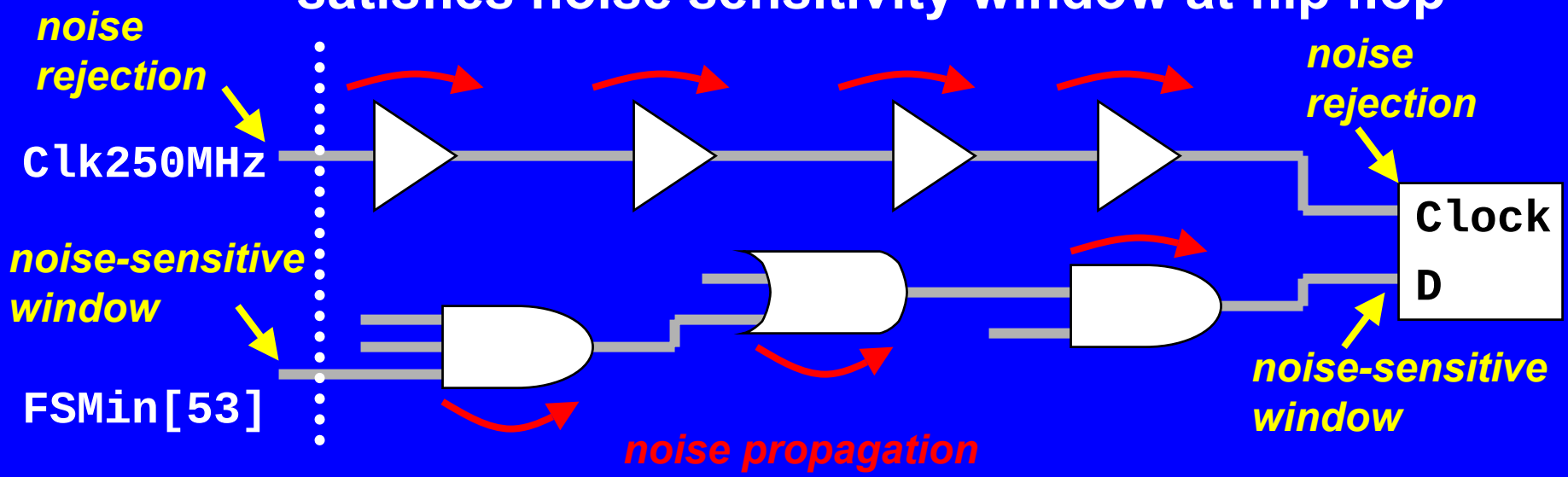
Output does not change its value



```
VECTOR ( 0* Clock -> *0 Clock -> ?- Q ) {
   LIMIT {
      NOISE v0 { PIN=Clock;
         MAX { HEADER {
               TIME t0 {
                  FROM { PIN=Clock; EDGE_NUMBER=0; }
                  TO { PIN=Clock; EDGE_NUMBER=1; }
            } } EQUATION { 0.2 + 0.1/(2.5+t0) }
}  }  }
```

Noise Analysis and Macro Modeling

- Path from input to flip flop inside block
  - apply noise propagation model
- Noise rejection at input
  - satisfies noise rejection at flip flop
- Noise sensitivity window at input
  - satisfies noise sensitivity window at flip flop



IP modeling with ALF

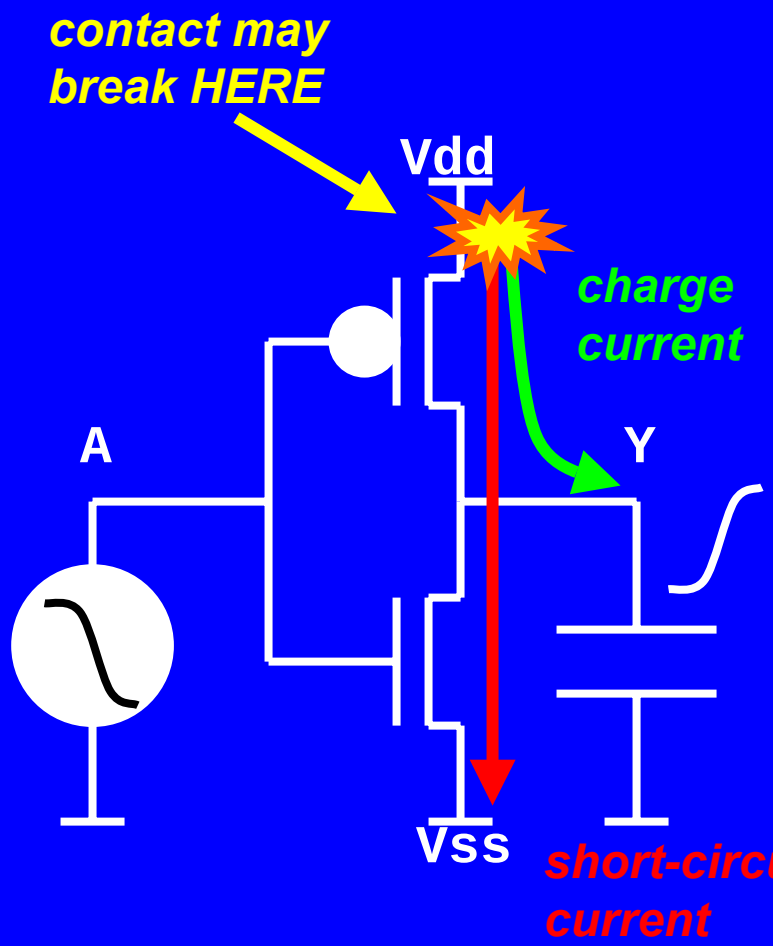
Reliability

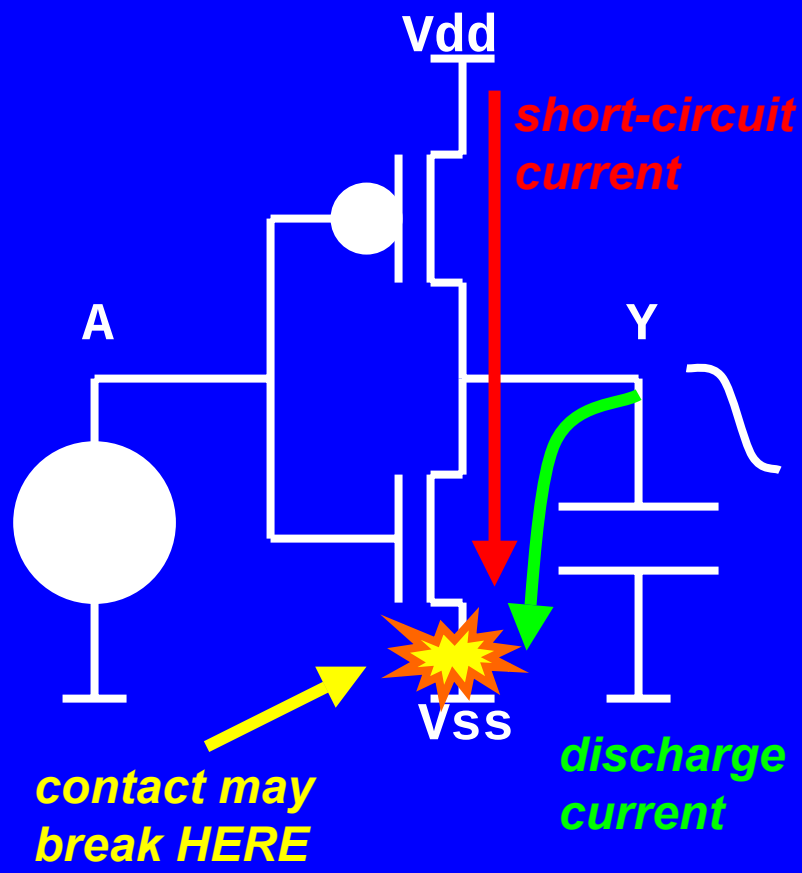
- Electromigration (EM)
  - High current density stresses vias and wires
  - Average current causes damage over time
  - Peak current causes immediate damage
  - Characterize for each cell, which piece breaks first
- Hot electron effect (HE)
  - High electrical field in switching transistor
  - Electrons get trapped in gate oxide
  - Electrical charge (flux) accumulation
  - Performance degradation over time

EM and HE require detailed characterization at device level
How can macro models be defined?

Electromigration

- Illustration of EM effect





Electromigration Modeling

- Direct modeling approach
  - define a LIMIT for average and peak current
  - also need a model for actual current calculation
  - only applicable for accessible PIN or PORT of cell
- Model for average and peak current in context of VECTOR
  - similar to power calculation
  - electrical CURRENT instead of POWER or ENERGY

Electromigration Modeling

Example for average and peak current models and limits

```
PIN Vdd { DIRECTION=input; VIEW=physical; PINTYPE=supply;
   LIMIT {  CURRENT Iav { MEASUREMENT=average; MAX=x; }
            CURRENT Ipk { MEASUREMENT=peak; MAX=x; } }
}
VECTOR (01 A -> 10 Y) {
   CURRENT Iav1 { PIN=Vdd;
      MEASUREMENT=average; TIME=x; /*put model here*/ }
   CURRENT Ipk1 { PIN=Vdd;
      MEASUREMENT=peak; /* put model here*/} }
VECTOR (10 A -> 01 Y) {
   CURRENT Iav2 { PIN=Vdd;
      MEASUREMENT=average; TIME=x; /*put model here*/ }
   CURRENT Ipk2 { PIN=Vdd;
      MEASUREMENT=peak; /*put model here*/} }
```

calculate and add up average

calculate and find max peak

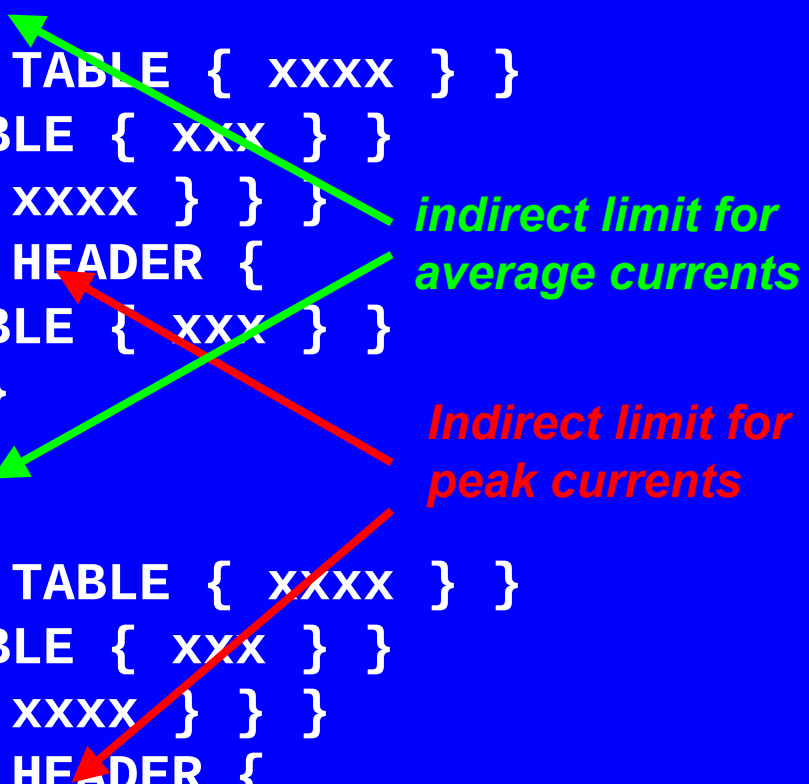
Electromigration Modeling

- Indirect modeling approach
  - Consider current on a path activated by a VECTOR
- Average current depends on input slewrate, output load and activation frequency of VECTOR
  - Define a slewrate- and load-dependent frequency LIMIT for the VECTOR
- Peak current depends on input slewrate and output load
  - Define a slewrate-dependent load capacitance LIMIT in context of VECTOR

Electromigration Modeling

Example for frequency and load capacitance limits

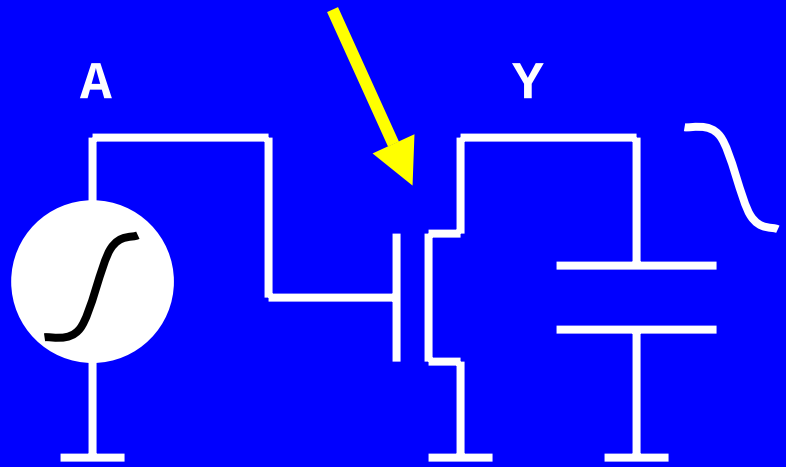
```
VECTOR (01 A -> 10 Y) { LIMIT {
      FREQUENCY { MAX { HEADER {
            CAPACITANCE { PIN=Y; TABLE { xxxx } }
            SLEWRATE { PIN=A; TABLE { xxx } }
            } TABLE {  xxxx xxxx xxxx } } }
      CAPACITANCE { PIN=Y; MAX { HEADER {
            SLEWRATE { PIN=A; TABLE { xxx } }
            } TABLE { xxx } } } }
VECTOR (01 A -> 10 Y) { LIMIT {
      FREQUENCY { MAX { HEADER {
            CAPACITANCE { PIN=Y; TABLE { xxxx } }
            SLEWRATE { PIN=A; TABLE { xxx } }
            } TABLE {  xxxx xxxx xxxx } } }
      CAPACITANCE { PIN=Y; MAX { HEADER {
            SLEWRATE { PIN=A; TABLE { xxx } }
            } TABLE { xxx } } } }
```

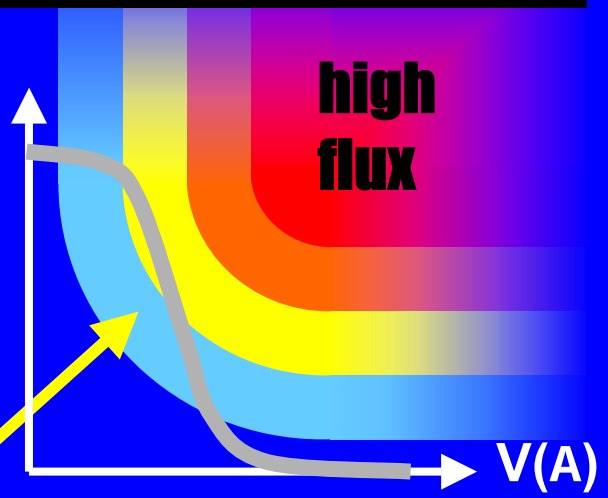


Hot Electron

- Illustration of HE effect

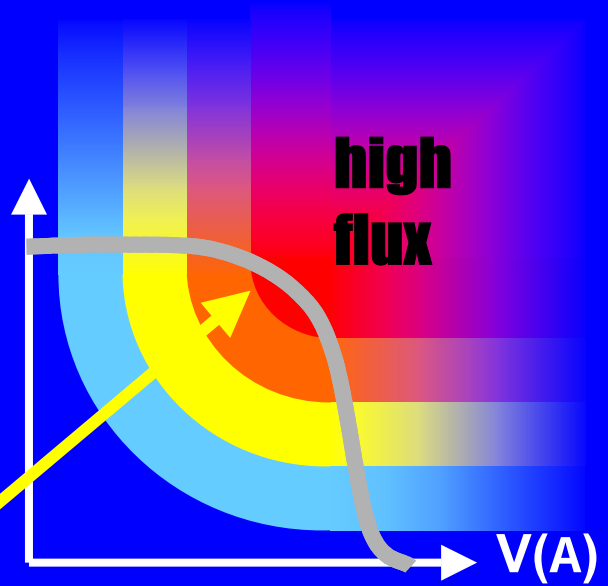
occurs only when NMOS transistor switches "on"





Input slow
Output fast

Transistor avoids high flux zone



Input fast
Output slow

Transistor enters high flux zone

IP modeling with ALF

Hot Electron Modeling

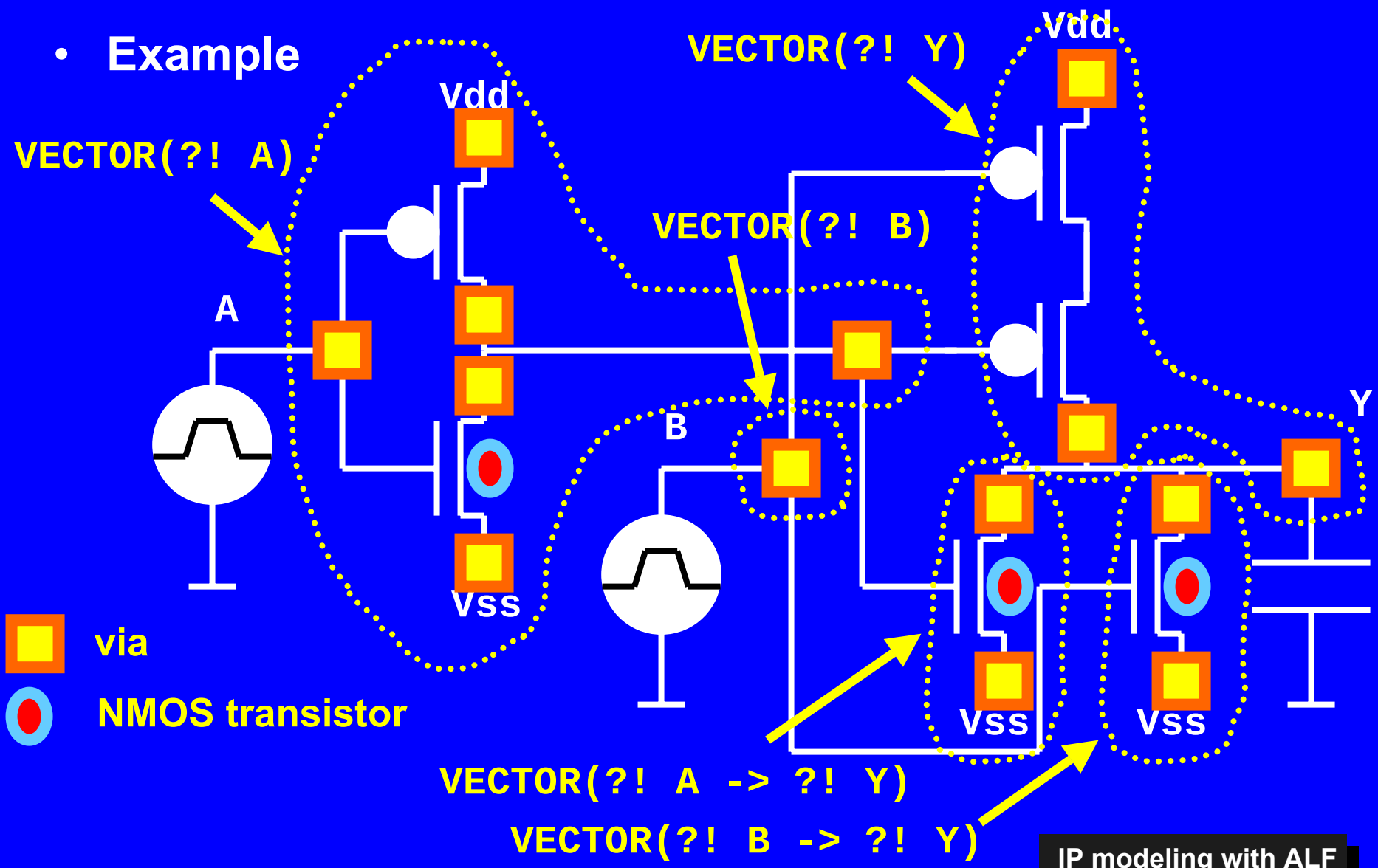
- Direct modeling approach
  - HE damage can be measured by the quantity of accumulated electrical charge or FLUX
  - define a LIMIT for FLUX
  - define a model for FLUX in context of VECTOR
  - FLUX in HE corresponds to CURRENT in EM
- Indirect modeling approach
  - Take hot electron effect into account within LIMIT for FREQUENCY in context of VECTOR

EM and HE Macro Modeling

- Abstract macro models for both EM and HE can be represented as frequency limit within a set of vectors
- Actual frequency is the activation rate of the vector
- Frequency limit depends on stimulus parameters involved in the vector, such as slewrate and load capacitance of switching input or output pins
- Frequency limit may also depend on expected lifetime and environmental temperature
- Finding a set of characterization vectors exhibiting all EM/HE damage within a cell is a fault coverage problem
  - Vector set can be identified by hand for primitive cells
  - Automation techniques available for complex cells

EM and HE Macro Modeling

- Example



IP modeling with ALF

Physical Modeling

- Physical Technology Description
- Physical Cell Description
- Hierarchical Block Description
- Power Rails
- Antenna Rules

Physical Technology Description

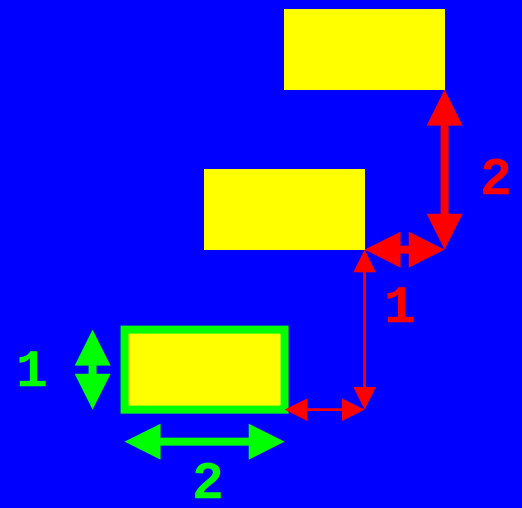
- LAYER defines physical stack up
  - may contain electrical parasitic estimation model
- VIA defines vertical contact structure
  - use TEMPLATE, if structure involves variable parameters (e.g. variable number of contacts)
- RULE defines relationship between physical PATTERNs
  - e.g. DISTANCE, OVERHANG, electrical parasitics
- SITE defines legal placement location for set of cells
- ANTENNA defines special manufacturing rule
- CONNECTIVITY defines electrical rule check

Physical Cell Description

- BLOCKAGE within context of CELL
  - describes PATTERN for routing obstruction
- PORT within context of PIN
  - describes PATTERN for electrical connection
- Shape of PATTERN is described by geometric model

Example:

```
PATTERN \3_rectangles {
        LAYER = metal1;
        RECTANGLE { 0 0 2 1 }
        REPEAT = 3 { SHIFT {
                        HORIZONTAL=1;
                        VERTICAL=2;
}       }       }
```

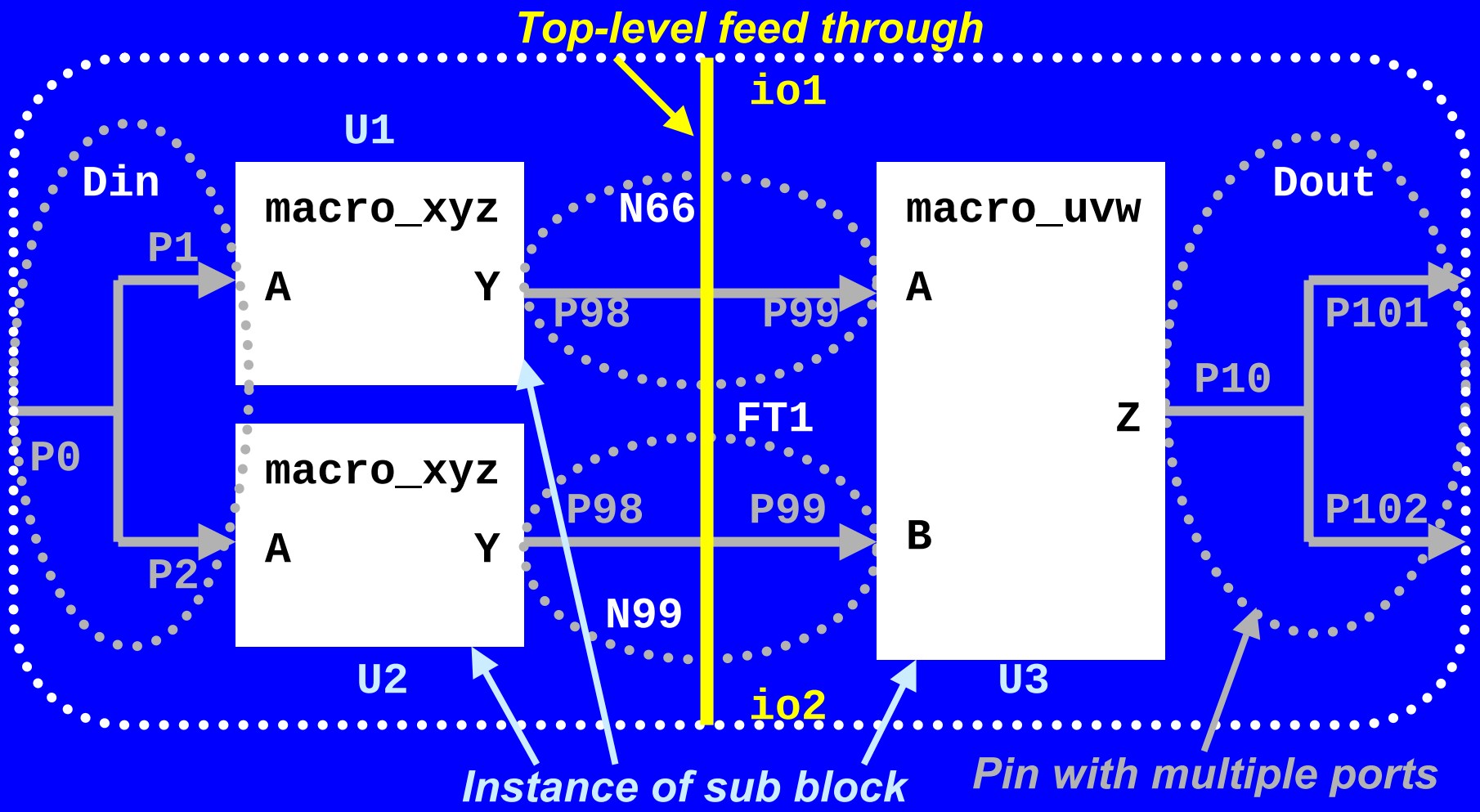


Hierarchical Block Description

- Complex macros with sub blocks
  - Describe structure and internal nodes
- Hard Macro: Layout exists already
  - Describe physical patterns for blockages and ports
  - Describe electrical parasitics within block
  - Describe timing
    - routing over block may cause crosstalk
- Soft Macros: Layout does not yet exist
  - Describe constraints for area and port locations
  - Describe model for electrical parasitic estimation

Hard Macro Description

- PIN and PORT declaration, STRUCTURE description

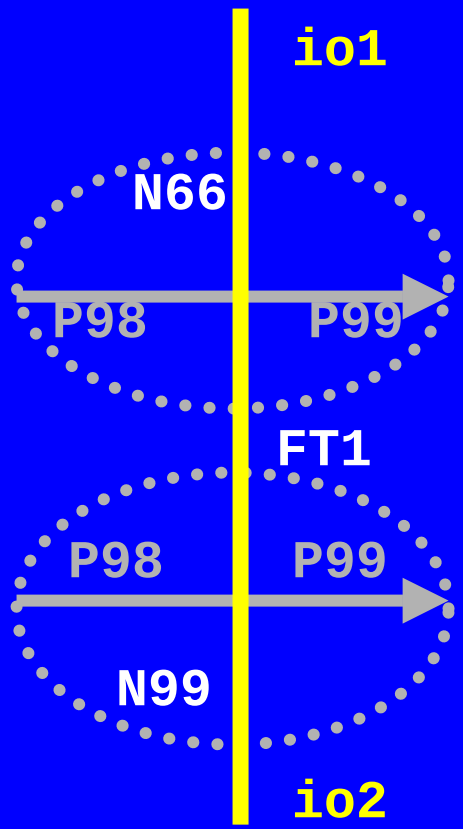


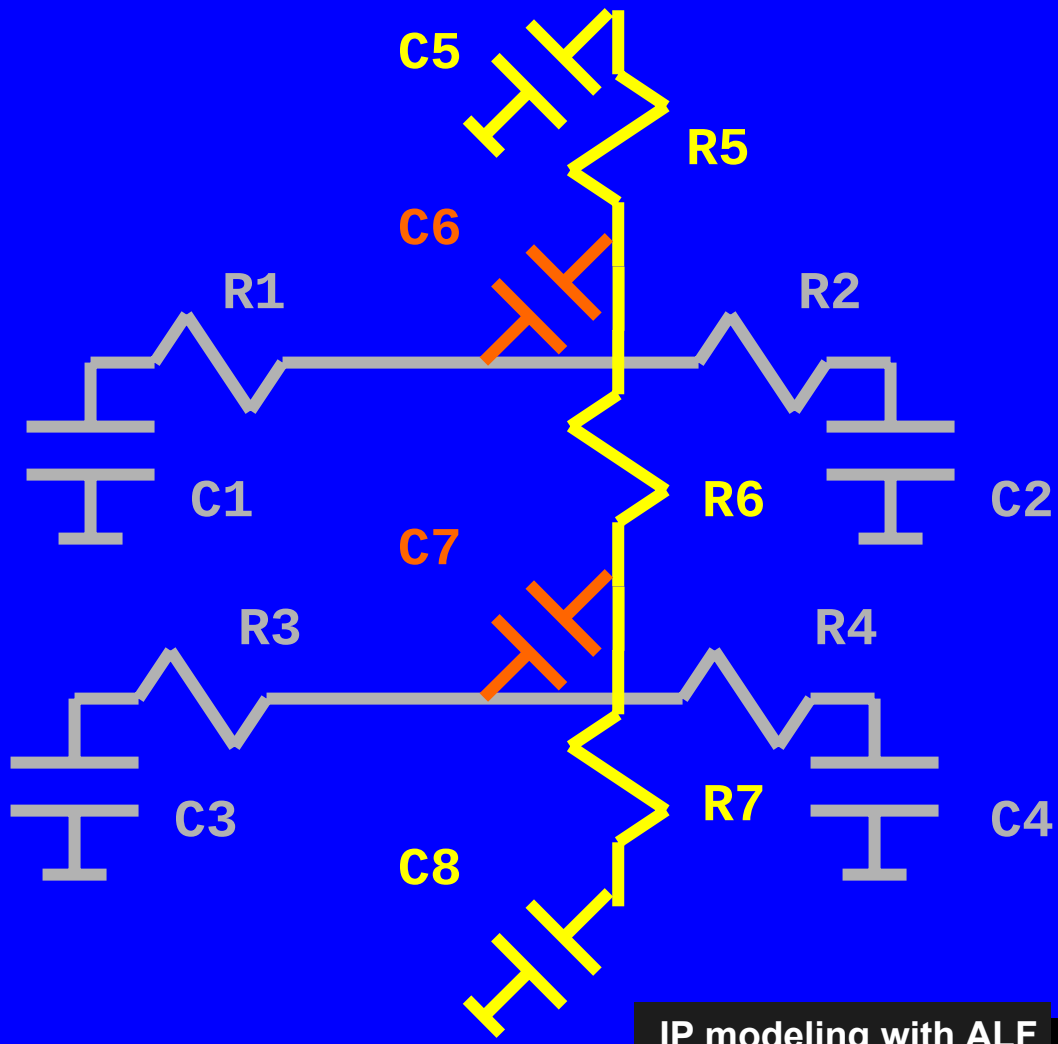
Hard Macro Description

```
CELL my_hardmacro {
  PIN Din { DIRECTION=input; ROUTING_TYPE=regular;
    PORT P0 {VIEW=physical; /* PATTERN description here */}
    PORT P1 {VIEW=none;} PORT P2 {VIEW=none;} }
  PIN Dout { DIRECTION=output; ROUTING_TYPE=regular;
    PORT P101 {VIEW=physical;} PORT P102 {VIEW=physical;}
    PORT P10  {VIEW=none;} }
  PIN FT1 { DIRECTION=none; ROUTING_TYPE=feedthrough;
    PORT io1 {VIEW=physical;} PORT io2 {VIEW=physical;} }
  PIN N66 {DIRECTION=none; VIEW=none; PORT P98; PORT P99; }
  PIN N99 {DIRECTION=none; VIEW=none; PORT P98; PORT P99; }
  FUNCTION { STRUCTURE {
    macro_xyz U1 { A=Din.P1; Y=N66.P98; }
    macro_xyz U2 { A=Din.P2; Y=N99.P98; }
    macro_uvw U3 { A=N66.P99; B=N99.P99; Z=Dout.P10; }
  } } /* boundary parasitics, timing */ }
```

Hard Macro Description

- Boundary Parasitics (only one sub-network shown)





IP modeling with ALF

Hard Macro Description

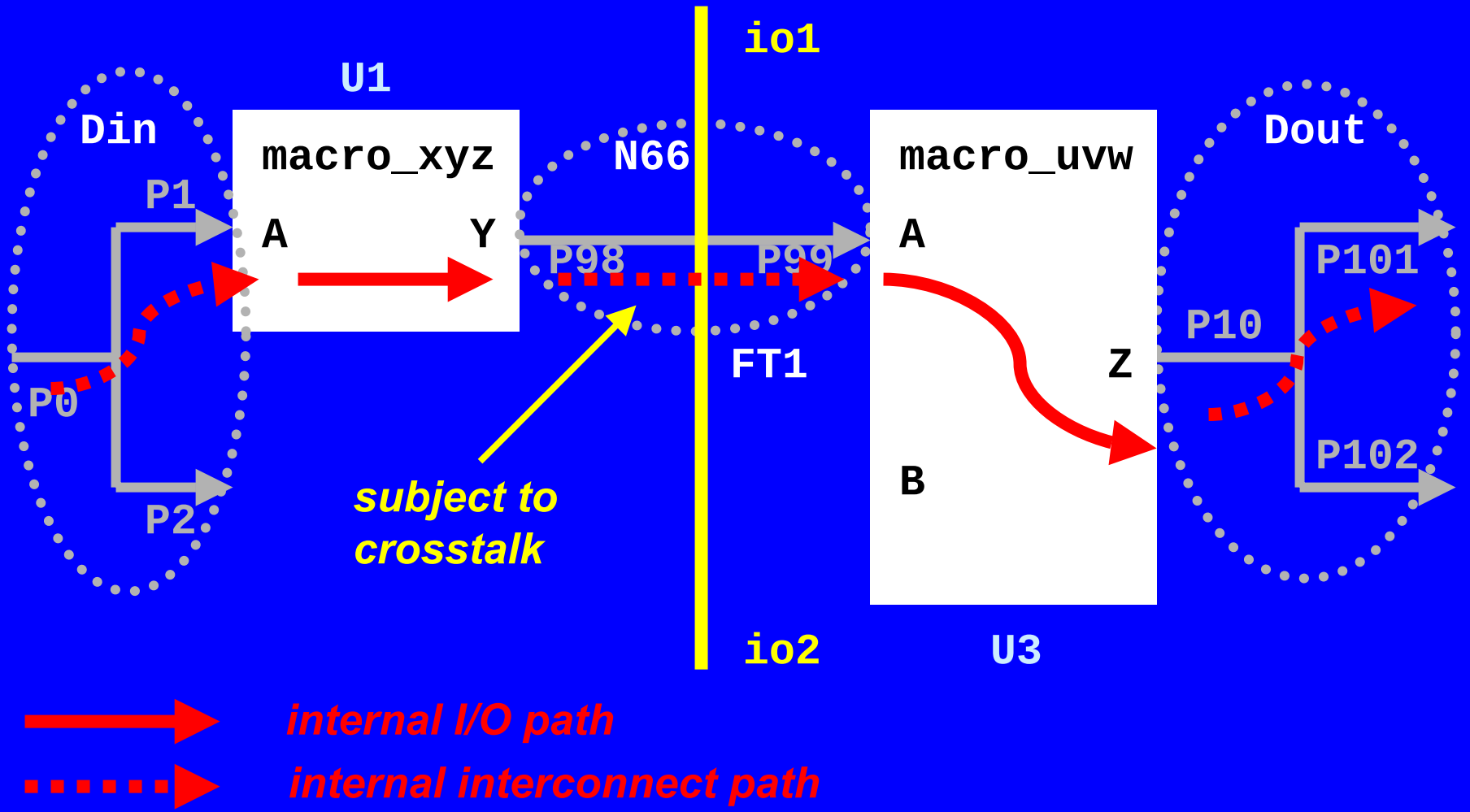
```
WIRE boundary_parasitics_N66_N99_FT1 {
  NODE gnd { NODETYPE=ground; }
  CAPACITANCE C1=1.2 { NODE { N66.P98 gnd   } }
  CAPACITANCE C2=2.1 { NODE { N66.P99 gnd   } }
  CAPACITANCE C3=1.3 { NODE { N99.P98 gnd   } }
  CAPACITANCE C4=1.9 { NODE { N99.P99 gnd   } }
  CAPACITANCE C5=0.5 { NODE { FT1.io1 gnd   } }
  CAPACITANCE C6=0.8 { NODE { FT1.io1 N66_1 } }
  CAPACITANCE C7=0.7 { NODE { FT1.io2 N99_1 } }
  CAPACITANCE C8=0.4 { NODE { FT1.io2 gnd   } }
  RESISTANCE  R1=58  { NODE { N66.P98 N66_1 } }
  RESISTANCE  R2=67  { NODE { N66.P99 N66_1 } }
  RESISTANCE  R3=55  { NODE { N99.P98 N99_1 } }
  RESISTANCE  R4=69  { NODE { N99.P99 N99_1 } }
  RESISTANCE  R5=25  { NODE { FT1.io1 FT1_1 } }
  RESISTANCE  R6=11  { NODE { FT1_1   FT1_2 } }
  RESISTANCE  R7=31  { NODE { FT1.io2 FT1_2 } }
}
```

Declared special node

Undeclared internal node

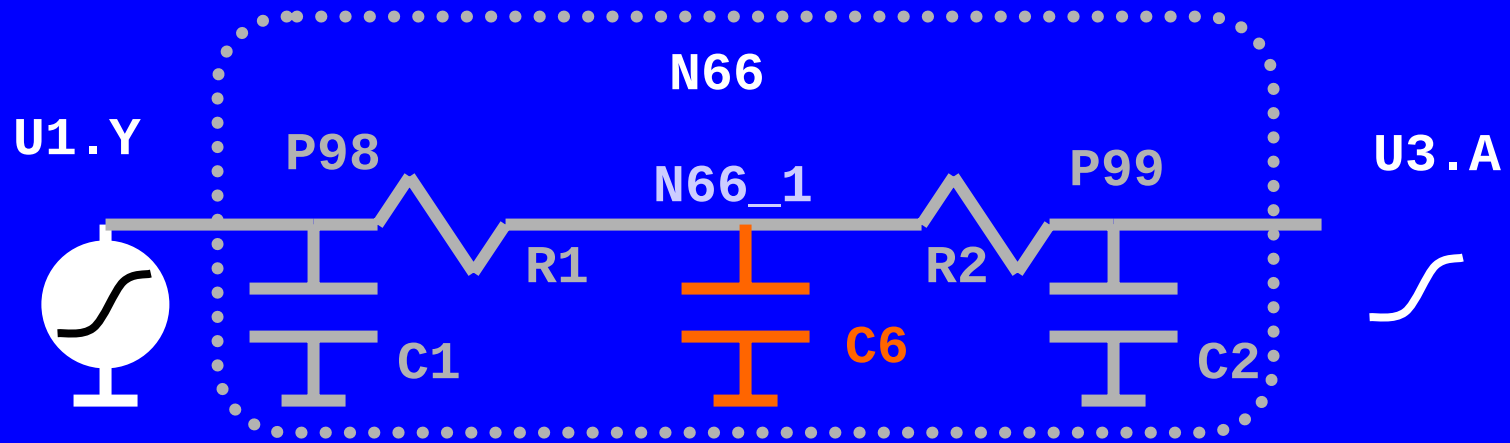
Hard Macro Description

- Timing (only one path shown)



Hard Macro Description

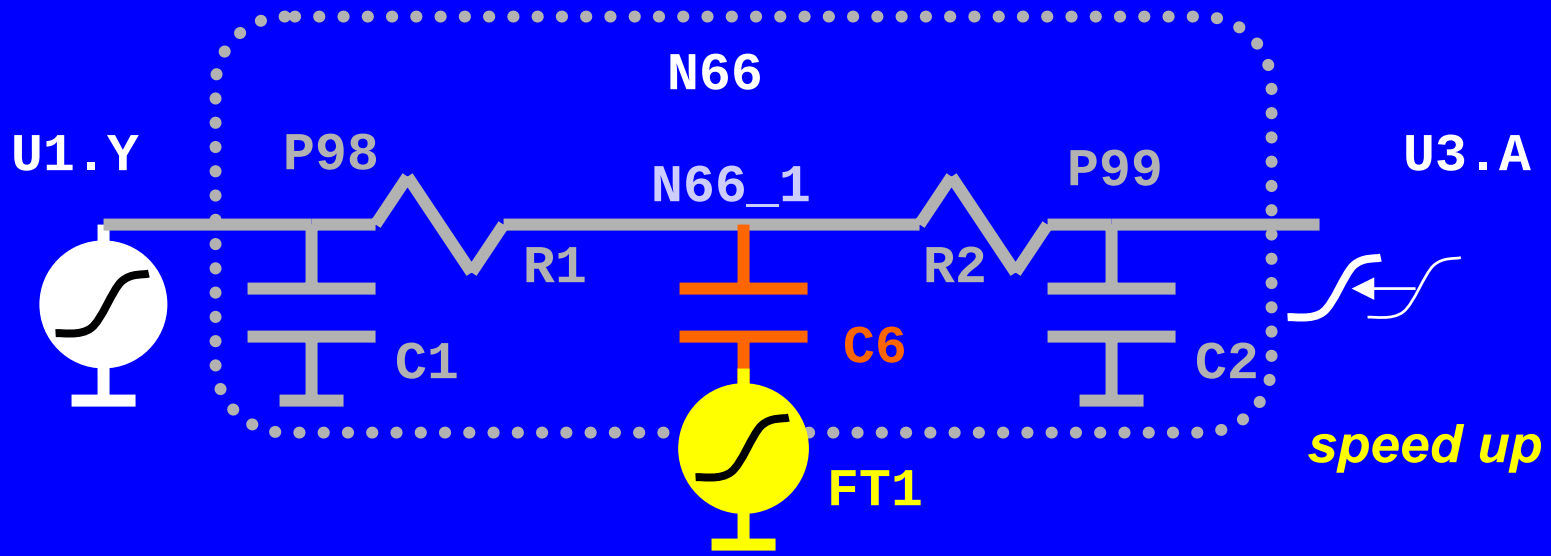
Timing without crosstalk




```
VECTOR (01 N66.P98 -> 01 N66.P99) {
  DELAY { FROM {PIN=N66.P98;} TO {PIN=N66.P99;}
    CALCULATION = absolute;
    HEADER { SLEWRATE { PIN=N66.P8; TABLE { xxx } } }
    TABLE { xxx } }
  SLEWRATE { PIN=N66.P99;
    HEADER { SLEWRATE { PIN=N66.P8; TABLE { xxx } } }
    TABLE { xxx } } }
```

Hard Macro Description

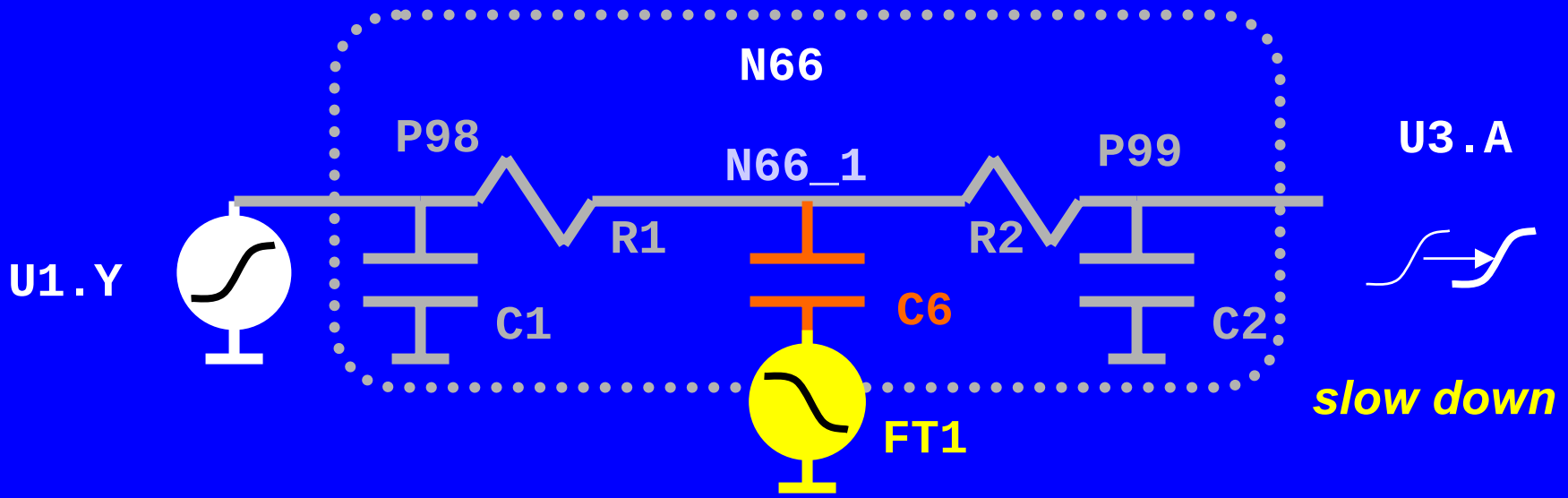
Timing with same-phase crosstalk




```
VECTOR (01 N66.P98 <&> 01 FT1 -> 01 N66.P99) {
  DELAY { FROM {PIN=N66.P98;} TO {PIN=N66.P99;}
    CALCULATION = incremental;
    HEADER { SLEWRATE S1 {PIN=N66.P98;}
      SLEWRATE S2 { PIN=FT1.io1; }
      TIME T12 { FROM {PIN=N66.P98;} TO {PIN=FT1.io1;} }
    } EQUATION { -(S1/S2)* 1/(1+T12**2) } } }
```

Hard Macro Description

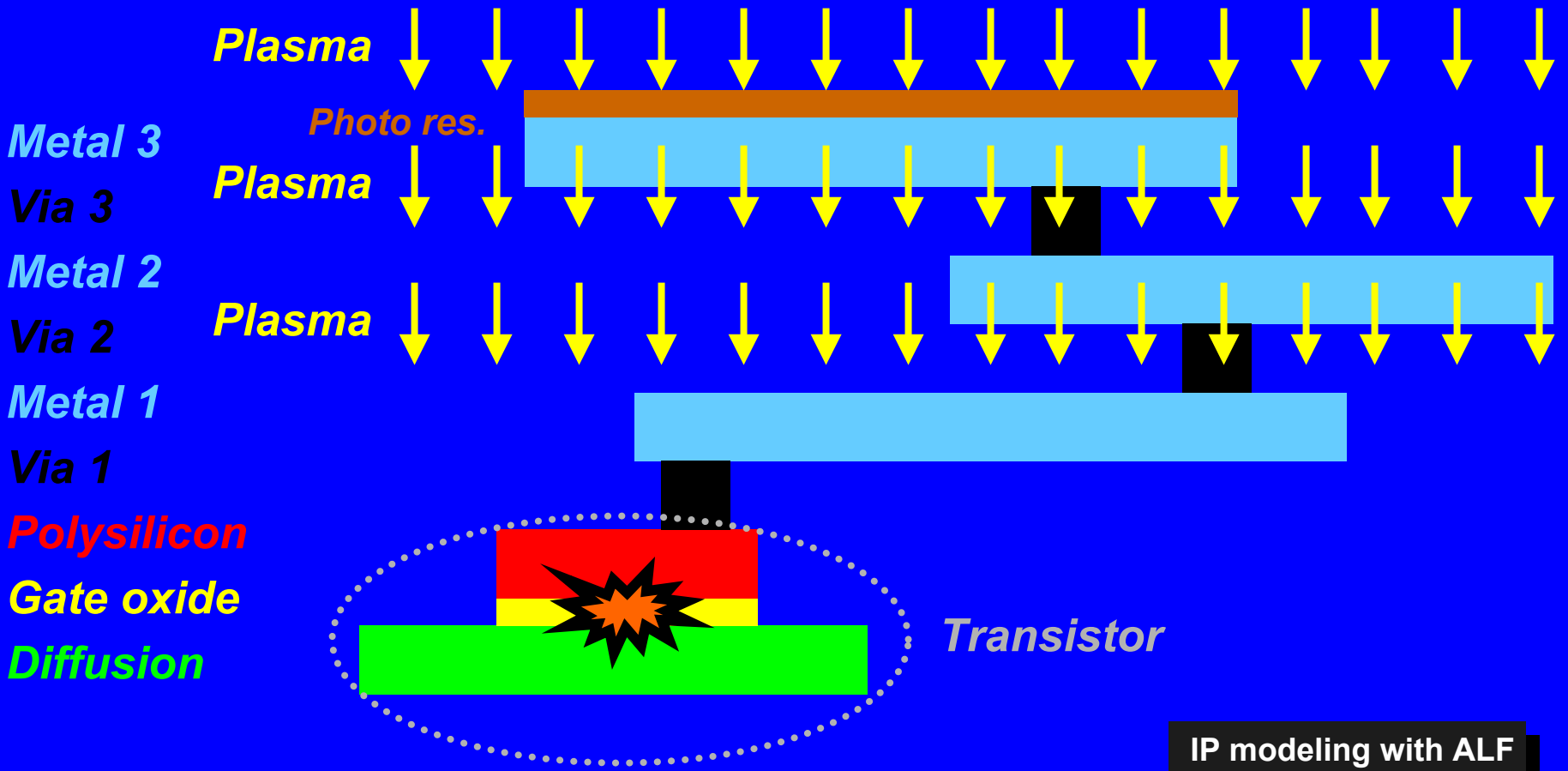
Timing with opposite-phase crosstalk




```
VECTOR (01 N66.P98 <&> 10 FT1 -> 01 N66.P99) {
  DELAY { FROM {PIN=N66.P98;} TO {PIN=N66.P99;}
    CALCULATION = incremental;
    HEADER { SLEWRATE S1 {PIN=N66.P98;}
      SLEWRATE S2 { PIN=FT1.io1; }
      TIME T12 { FROM {PIN=N66.P98;} TO {PIN=FT1.io1;} }
    } EQUATION { (S1/S2)* 1/(1+T12**2) } } }
```

Antenna effect

- Transistor collects charge during etching of metal layers
- Cumulative effect may damage the transistor



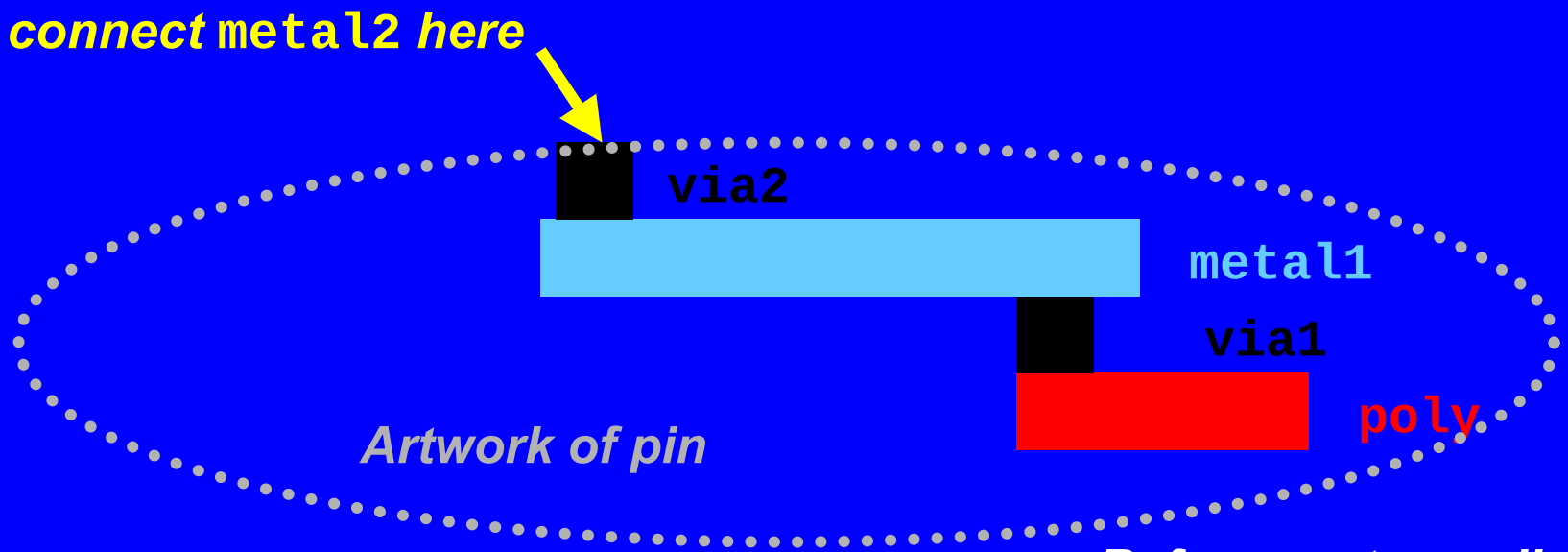
Antenna modeling

- Antenna effect depends on
  - area and/or perimeter of polysilicon
  - area and/or perimeter of upper metal layers
- The greater the ratio between metal and polysilicon, the worse the damage

```
ANTENNA my_antenna_rule {
  SIZE ratio1 { CALCULATION=incremental;
    HEADER { AREA a1{LAYER=metal1;} AREA a0{LAYER=poly;} }
    EQUATION { a1 / a0 } }
  SIZE ratio2 { CALCULATION=incremental;
    HEADER { AREA a2{LAYER=metal2;} AREA a0{LAYER=poly;} }
    EQUATION { a2 / a0 } }
  LIMIT { SIZE { MAX = 10.5; } }
}
```

Antenna modeling

- When a route connects to a pin of a hierarchical block, antenna checker must know what area is underneath



Reference to applicable antenna rule

```
PIN my_pin {
  AREA a1 = 2.8 { LAYER=metal1; ANTENNA=my_antenna_rule; }
  AREA a0 = 1.6 { LAYER=poly; ANTENNA=my_antenna_rule; }
}
```

Conclusion

- ALF provides a comprehensive hardware modeling language
- Suitable for functional, electrical and physical design from RTL to tape out
- Abstract modeling concepts suitable for IP blocks
  - Arithmetic model for electrical and physical performance characterization
  - Electrical performance data in context of VECTOR
- ALF adds significant value for System-on-Chip design

Status and Outlook

- ALF is approved standard by Accellera
  - Version 1.1 in 1999, Version 2.0 in 2001
  - IEEE standard to be completed in 2003
- Emerging de-facto standard for next-generation tools
  - Timing, power, signal integrity closure
  - RTL planning and virtual prototyping
- Adopted by leading semiconductor houses
  - Also supported by library creation tools
- Defining impact on other standards

Status and Outlook

- **ALF specification and other useful information on ALF is freely available on the web**

  `http://www.eda.org/alf`

- **ALF IEEE Study group currently supported by individuals from**

  Avant!, ASC, LogicVision, Magma, Mentor Graphics, Monterey, NEC, Philips, Sequence, Silicon Metrics, Simplex, SUN, Synopsys, Tera Systems

Please join the ALF IEEE study group