

CPE 526 —Introduction to VLSI Design Using Hardware Description Languages, Modeling, and Synthesis

Dr. Rhonda Kay Gaede

UAH

UAH

CPE 426/526

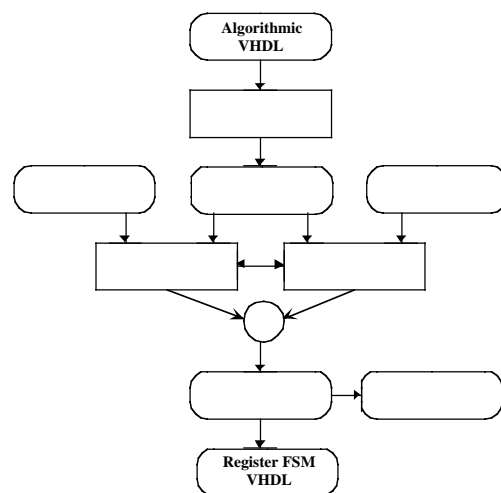
Synthesis Algorithms for Design Automation

¥ The task of searching a very large design space for the lowest cost structure to meet all the constraints imposed by technology and design specifications is a formidable one.

12.1 Benefits of Algorithmic Synthesis

- ¥ Shorter Design Cycle
- ¥ Lower Design Cost
- ¥ Lower Production Cost
- ¥ Fewer Design Errors
- ¥ Easier to Explore Design Space
- ¥ Easier to Document
- ¥ Easier to Change

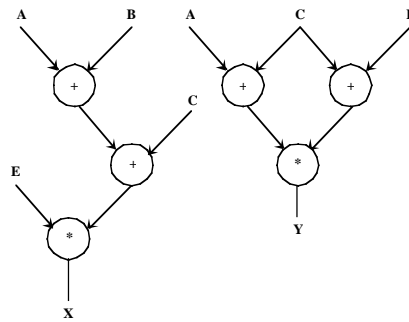
12.2 Algorithmic Synthesis Tasks



12.2.1 Compilation

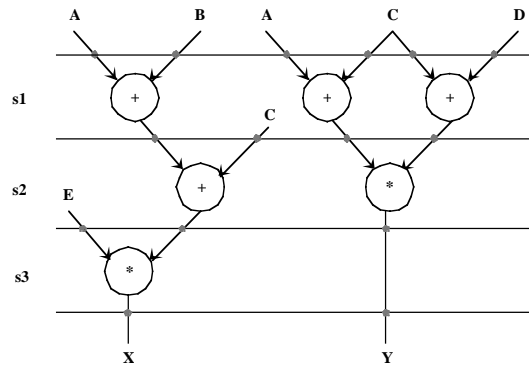
```
entity SYNEX1 is
  port (A, B, C, D, E: in INTEGER;
        X, Y: out INTEGER);
end SYNEX1;
```

```
architecture HIGH_LEVEL of SYNEX1 is
begin
  X <= E*(A+B+C);
  Y <= (A+C)*(C+D);
end HIGH_LEVEL;
```



12.2.2 Scheduling

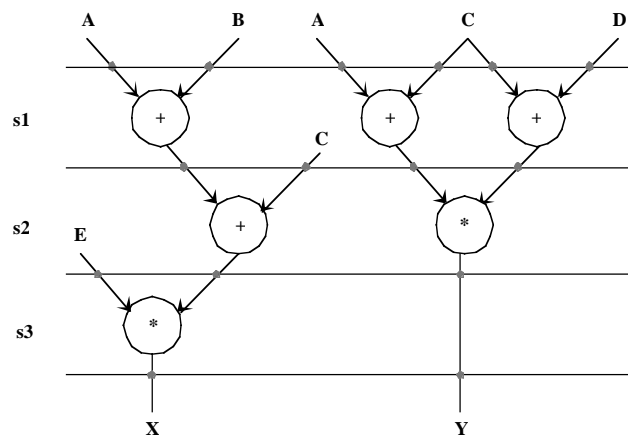
- ⌘ Place each operation in the appropriate control step (clock cycle)
- ⌘ Consider a schedule using 3 adders and 1 multiplier



12.2.3 Allocation

¥ Specify the components and the interconnections between them

12.2.3.1 Allocation of Registers to Hold Data Values

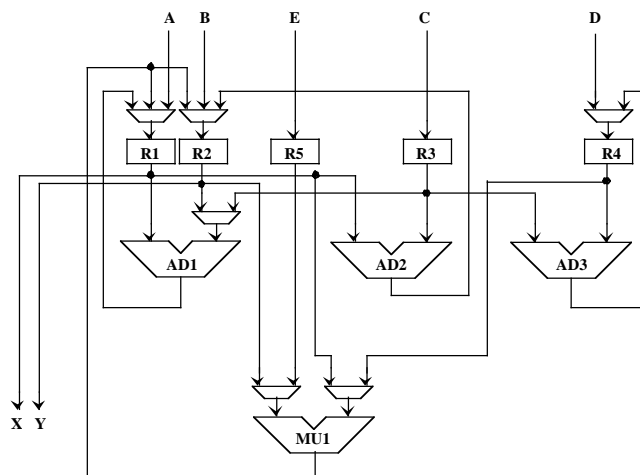


12.2.3.2 Allocation of Functional Units

¥ Bottom up allocation selects items from a library

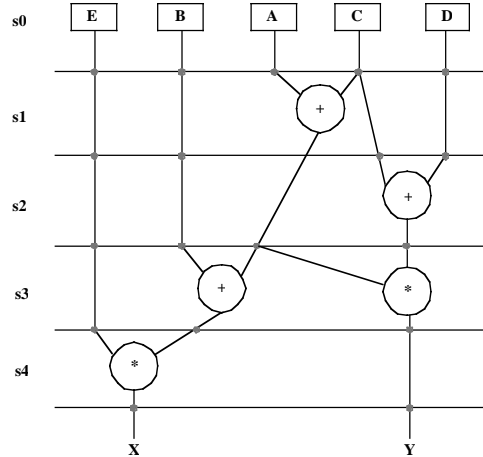
Component	Quantity
Adder	3
Multiplier	1
Register	5
2 x 1 MUX	4
3 x 1MUX	2

12.2.3.3 Allocation of Data Paths

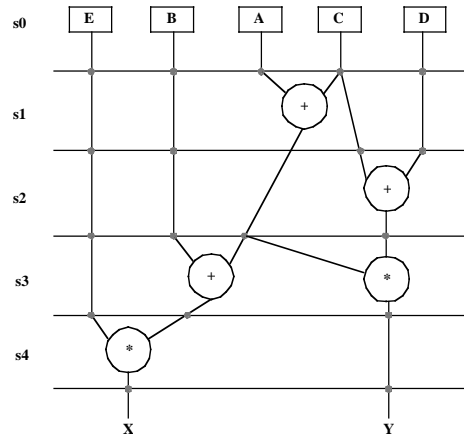


12.2.4 Interaction of Scheduling and Allocation

- ✘ Schedules are dependent on numbers of functional units.
- ✘ Schedules are dependent on the number of clock cycles needed by a functional unit.
- ✘ Consider a constraint of having only one multiplier and one adder (The common term $A + C$ has been used here)



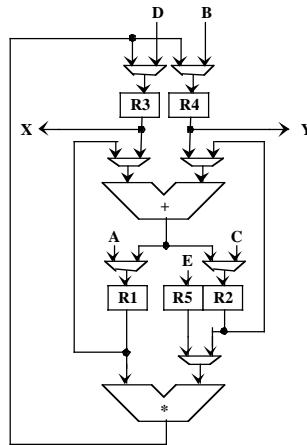
Allocation for Constrained Schedule



Component	Quantity
Adder	1
Multiplier	1
Register	5
2 x 1 MUX	7

	s0	s1	s2	s3	s4
AD1	--	b	b	b	--
MU1	--	--	--	b	b

Data Path for Constrained Schedule



FSM VHDL Model Example

```

architecture FSM of FSMEX1 is
  type STATE_TYPE is (S0, S1, S2, S3, S4);
  signal STATE: STATE_TYPE;
  signal R1, R2, R3, R4, R5: INTEGER;
begin
  -- Process to update state and perform register transfers.
  STATEP: process (CLK)
  begin
    if CLK'event and CLK='1' then
      case STATE is
        when S0 => R5 <= E; R4 <= B; R3 <= D; R2 <= C; R1 <= A;
                   STATE <= S1;
        when S1 => R1 <= R1 + R2; STATE <= S2;
        when S2 => R2 <= R2 + R3; STATE <= S3;
        when S3 => R1 <= R4 + R1;
                   R4 <= R1 * R2; STATE <= S4;
        when S4 => R3 <= R5 * R1; STATE <= S0;
      end case;
    end if;
  end process STATEP;
  X <= R3; Y <= R4;
end FSM;

```

12.3 Scheduling Techniques

¥ Two approaches

—12.3.1 Transformational Scheduling

—12.3.2 Iterative/Constructive Scheduling

¥ 12.3.2.1 ASAP Scheduling

¥ 12.3.2.2 ALAP Scheduling

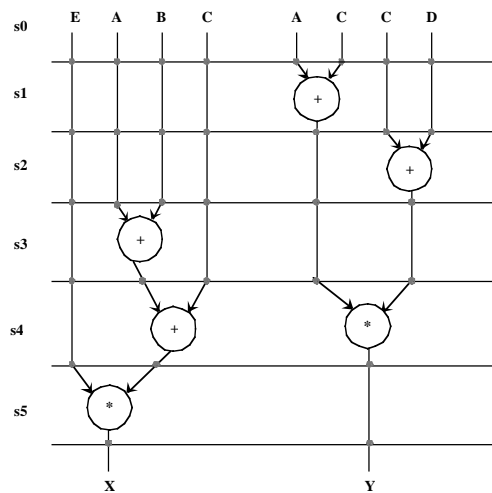
¥ 12.3.2.3 List Scheduling

¥ 12.3.2.4 Freedom-Directed Scheduling

12.3.1 Transformational Scheduling

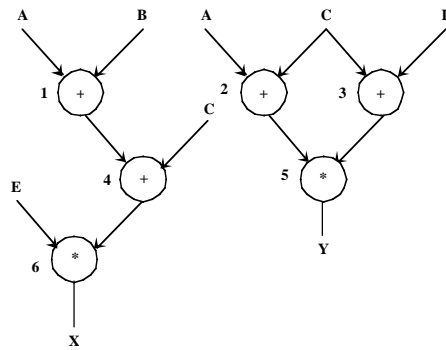
¥ Take schedule and transform it, one example transformation is state splitting

¥ It is prohibitively expensive to try all possible transformations.



12.3.2.1 ASAP Scheduling

For each control step (from inputs)
 Schedule as many operations as possible
 for which data is available



ASAP Example

- S0 _____
- S1 _____
- S2 _____
- S3 _____
- S4 _____
- S5 _____
- S6 _____

12.3.2.2 ALAP Scheduling

While there is more work (from outputs)

Schedule as many operations as possible

Increment the number of control steps

ALAP Example

S0

S1

S2

S3

S4

S5

S6

12.3.2.3 List Scheduling

- ¥ The problem with ASAP and ALAP is that only local information is used
- ¥ List scheduling exploits global information
- ¥ First, define a critical path as a path in the data flow graph of maximum length.
- ¥ Also, assign a priority of the node which is the length of the longest path from the node to the bottom of the graph
- ¥ **Algorithm**
 - Construct a list in descending priority
 - For each control step
 - Schedule as many operations as possible from the head of the list

List Scheduling Example

S0 _____

S1 _____

S2 _____

S3 _____

S4 _____

S5 _____

S6 _____

12.3.2.4 Freedom-Directed Scheduling

¥ This technique uses global information both in operation and control step selection.

¥ Algorithm

Perform ASAP and ALAP schedules, obtaining range of control steps for each operation with an upper bound on the number of control steps

For all operations

Schedule the one with the least freedom

Freedom-Directed Example

S0 _____

S1 _____

S2 _____

S3 _____

S4 _____

S5 _____

S6 _____