INDIAN INSTITUTE OF TECHNOLOGY GUWAHATHI

NPTEL

NPTEL ONLINE CERTIFICATION COURSE

An Initiative of MHRD

VLSI Design, Verification & Test

Dr. Arnab Sarkar

Dept. of CSE

IIT Guwahati

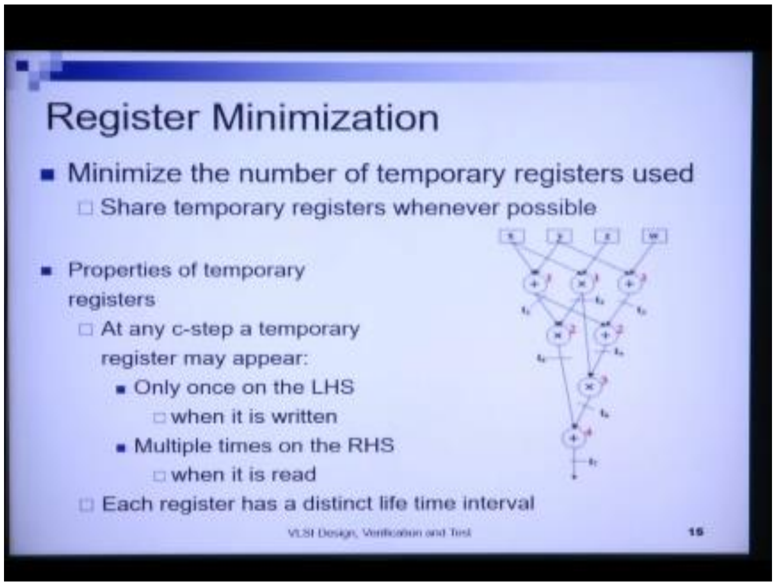
Welcome to module 1 of lecture 7 in the last module we took a look at when our two operations said to be compatible we also understood two important resource sharing models the compatibility graph model and the conflict proc model and we saw that a solution to the compatibility graph model in terms of finding the minimum number of clicks in it a minimum click partition of the of the compatibility graph gives me the maximum resource sharing possible we also saw that a minimum coloring of the conflict graph gives me an optimal solution to again to the resource sharing problem but we are we, we told that both the minimum click partitioning problem.

And the minimum graph coloring problem is in the complete for general graphs then we understood that there are a few special classes of graphs that are cordial graphs interval graphs and comparability graphs for which these problems have polynomial time optimal solutions we understood that the compatibility graph corresponding to the operations within a sim simple data flow graph within a basic block can be modeled as an interval graph why because each operation in an operation come in a single within a single operation constraints graph with an a basic block can be modeled as single continuous intervals from the start of it is execution to the end of its execution.

And hence the corresponding conflict and compatibility graph with respect to this scheduled operation constraint graph can be modeled as interval cross as we said next after understanding the resource shading models we will study algorithms for resource allocation we said that interval graphs gives us easy algorithms for graph coloring in polynomial time and these are optimal solutions we will study one of these algorithms in this in this lecture but before that we will understand the resource minimization problem we will understand the resource minimization problem can also be mapped as a as a graph coloring interval graphs.

And hence the same graph coloring algorithm can be used both for, for functional unit minimization that means maximization of resource sharing maximization of resource sharing among functional units and maximization of resource sharing among registers as well and hence before going to the actual coloring for interval graphs going to the actual algorithm for coloring interval graphs we will first understand the registered minimization problem.

(Refer Slide Time: 03:48)



Register Minimization

- Minimize the number of temporary registers used
  - Share temporary registers whenever possible
- Properties of temporary registers
  - At any c-step a temporary register may appear:
    - Only once on the LHS
      - when it is written
    - Multiple times on the RHS
      - when it is read
  - Each register has a distinct life time interval

x
y
z
w
+
×
+
×
+
×
+
1
1
1
2
2
3
4

VLSI Design, Verification and Test

15

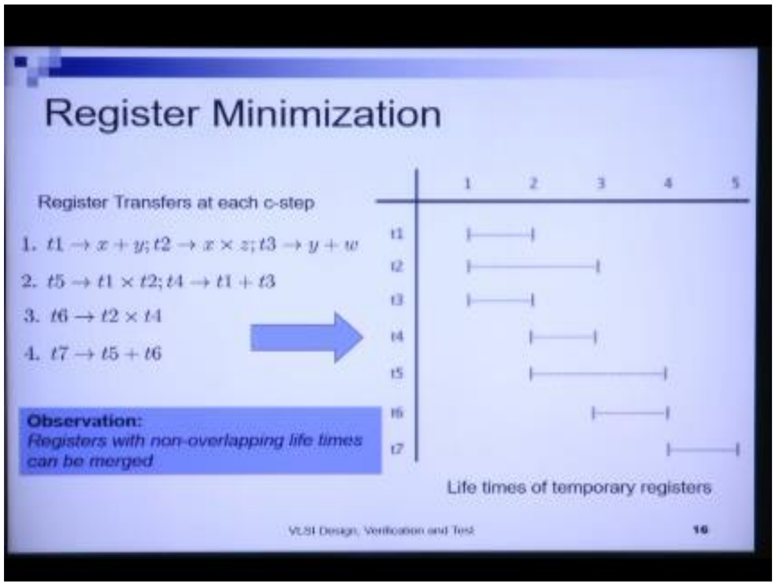
The problem is to minimize the number of temporary registers used so we see that there are four independent variables here X Y Z Wand hence four registers will anyhow for hardware registers will anyhow be required to store these variables so for this operations to be scheduled in time step one before time step one is activated XYZ w must be in different Hardware registers for this operation so execute correctly now that is out of our control however these temporary registers we said that the output of the operations are floated on temporary registers.

So each operation is floated on a different temporary registers at the output of the scheduling step now the registered minimization problem is to minimize the number of hardware registers that are required to, to, to allocate these temporary registers so these temporary registers must be allocated to Hardware registers as well right when I actually implement the schedule and hence I want to use a given Hardware registers for as many temporary registers as is possible.

This is how I do register minimization so what are the properties of temporary registers at any C step a temporary register may appear only once on the LHS when it is written and multiple times on the RHS when it is written for example now this temporary registered t1 is written in time step 1add because it is at the output of operation one subsequently this temporary register has been read in time step 2 so it is read by this multiplication operation here and this but this addition operation here this t1is read in both these operations and both these operations are in time step 2 right.

If you take another example let us take t2 so t2 is written to in time step one and is read in time step 2 by this multiplication operator and is ready in time step 3 by this multiplication operator so a temporary register will occur at most once on the left hand side of a register transfer and will appear and may appear multiple times on the RHS of a register transfer and each register therefore has a distinct life time interval from, from its first definition to its last use and hence this registers these temporary registers can, can have can be can be Mac two distinct intervals.

(Refer Slide Time: 06:44)


Register Minimization

Register Transfers at each c-step

1. $t1 \rightarrow x + y; t2 \rightarrow x \times z; t3 \rightarrow y + w$
2. $t5 \rightarrow t1 \times t2; t4 \rightarrow t1 + t3$
3. $t6 \rightarrow t2 \times t4$
4. $t7 \rightarrow t5 + t6$

Observation:
Registers with non-overlapping life times can be merged

Life times of temporary registers

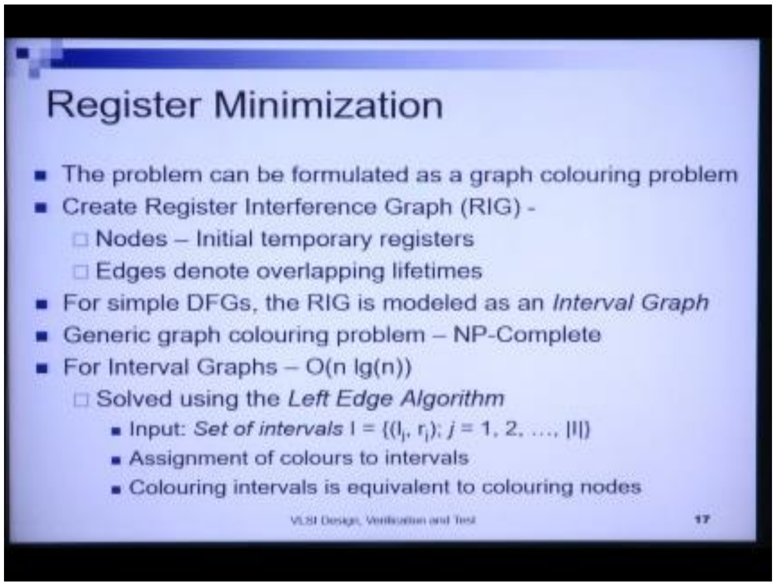
VLSI Design, Verification and Test 16

So what do we infer from all these intervals we infer that registers with non overlapping life times be merged so temporary registers whose lifetimes are non-overlapping can be merged and allocated the same instance of a hardware register for example here T 1 and T 4 can be allocated to the same instance of a register and so what is the what do the algorithm basically look like here we sort the intervals based on their left edges and then we take one hardware register which is bland and we take the first temporary register.

And put that in that hardware register now say r1 I have hardware register r1 and I have put temporary register T 1 into r 1because this has the earliest left edge now I can put another register which is the earliest temporary register means which is the earliest starting temporary register that I can put again in r1 any register who is left edge is higher than this right edge of t1 right so thereforet4 is one such t4 is the next such earliest starting temporary register which can be allocated to r1 similarly I can also allocate t-62 r 1 because were term because t4 and t6 are non-overlapping the intervals are non-overlapping.

So I can put into r1 t1 t 4 and t6 I can even put t7 into r1so therefore the same register r1 can be used to hold T 1 T for T 6 and t7likewise then I all these when I have already allocated when I have already allocated a set of temporary registers to a given Hardware registers I now remove all these registers from consideration from the remaining so no remaining temperature can be allocated to the hardware resistor r1 so I put aside our one with its allocation T 1 T for T 6 and t7 now for the remaining register for now for the remaining temporary registers we try to allocate again the minimum number of hardware registers that.

(Refer Slide Time: 09:29)



Register Minimization

- The problem can be formulated as a graph colouring problem
- Create Register Interference Graph (RIG) -
  - Nodes – Initial temporary registers
  - Edges denote overlapping lifetimes
- For simple DFGs, the RIG is modeled as an *Interval Graph*
- Generic graph colouring problem – NP-Complete
- For Interval Graphs – O(n lg(n))
  - Solved using the *Left Edge Algorithm*
    - Input: *Set of intervals* I = $\{(l_j, r_j); j = 1, 2, \ldots, |I|\}$
    - Assignment of colours to intervals
    - Colouring intervals is equivalent to colouring nodes

VLSI Design, Verification and Test 17

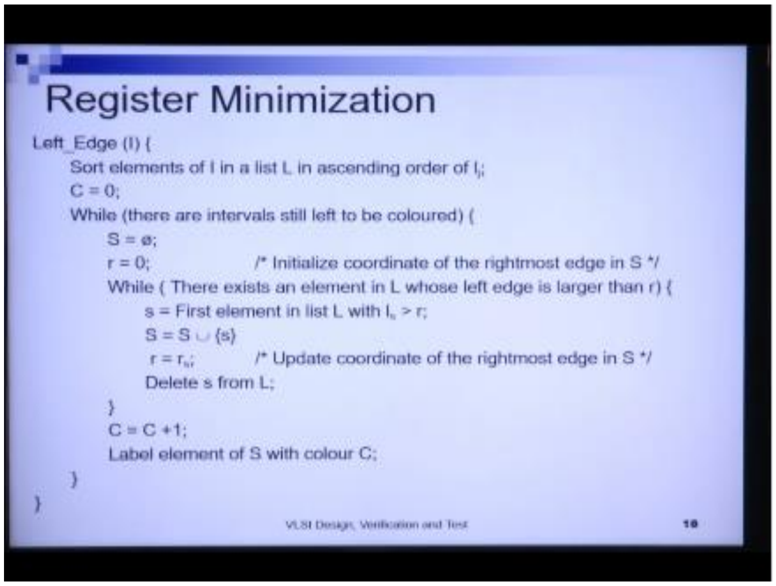
That I need to allocate these temporary register in a very similar manner so the algorithm that we have can be formulated as a graph coloring problem so this particular register the conflict graph so this is a registered conflict graph for the register conflict graph this register conflict graph is also called a registered interference graph now the register interference graph has nodes as the initial temporary registers the edges denote overlapping lifetimes right for simple DF geez register interference graph is an interval graph.

It is similar to the fact that for simple DLgs the functional or the other behavioral operations in my operation constraint graph are also interval so do also have a simple continuous intervals and hence their conflict graph can also be represented as an interval graph here again the interval graph or resistance the conflict graph for a registers can also be similarly mapped as an interval graph now as we said the generic problem is np-complete however what interval graphs the optimal problem can be solved in polynomial time.

The optimal graph coloring can be solved in polynomial time and in fact in n log n time and that also in n log n time because prior to the algorithm I need to sort the left edges of all the temporary registers in a list so what are the left edges left edges are the start times the definition times the first definition times of the temporary registers so I need to keep a list which maintains the maintains all my temperature registers that have not yet been allocated into a list sorted in terms of their start times or their definition times or the left edges.

So in the interval graph I have a set of intervals L LJ RJ so this is the LG to RJ is a lifetime for temporary register TJ now we need to assign a distinct color to a temporary register what does this assignment of color means a distinct color if I assign to a temporary register mean that I have assigned a distinct hardware register to this temporary register so now this list I of all temporary registers are taken as input by the left edge algorithm.

(Refer Slide Time: 12:13)



Register Minimization

```
Left_Edge (I) {
    Sort elements of I in a list L in ascending order of $l_i$;
    C = 0;
    While (there are intervals still left to be coloured) {
        S = ø;
        r = 0;              /* Initialize coordinate of the rightmost edge in S */
        While ( There exists an element in L whose left edge is larger than r) {
            s = First element in list L with $l_s > r$;
            S = S ∪ {s}
            r = $r_s$;       /* Update coordinate of the rightmost edge in S */
            Delete s from L;
        }
        C = C +1;
        Label element of S with colour C;
    }
}
```

VLSI Design, Verification and Test 10

Now given this list of intervals it first sorts this elements in I in a list L in ascending order of the left edges first I assign 0 colors means C equals to 0what does C means he gives a labeling for colors C equals to 1 is a distinct color C equals to 2 is another distinct color C equals to 3 than others distinct colors so if I allocate a color c equals to 1 to a set of temporary registers then all those temporary registers will be will be allocated to the same Hardware register whose colored is labeled one right.

So while there are interval still left to be colored we will take a hardware register we will take a distinct color and find out what are the maximum possible number of temporary registers that can be allocated to this Hardware register what is the maximum number of temporary sisters that can be assigned the same color okay so how do I do that first s equals to 5 that means s will contain s is a hardware register I have initialized and currently it does not contain any registered unit and hence s equals to 5 s is the set of temporary registers that can be allocated to a single Hardware registers in each given pass with this while loop is a pass firstly r equal to 0.

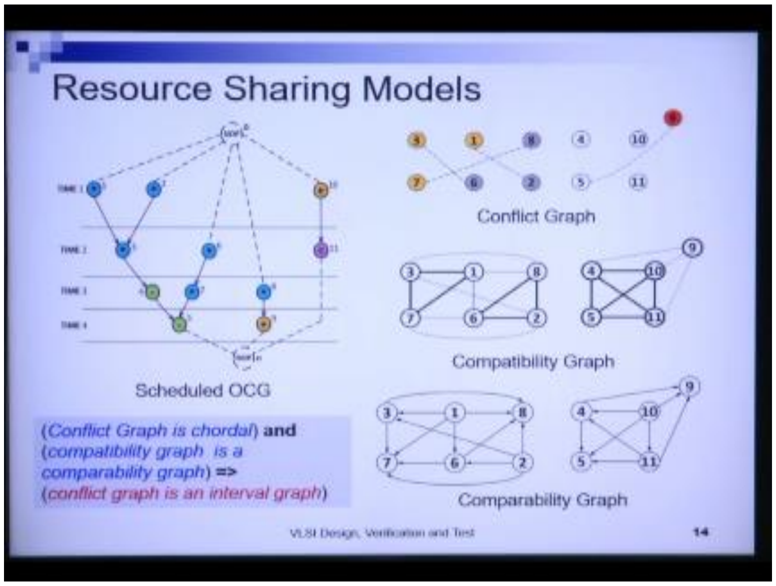
The initial coordinate of the rightmost edge in s in the hardware register is set to 0 while there exists an element in l who is left edge is larger than our that means that I want to find a left edge in the list l and in the list l all my left edges are sorted in sorted in increasing order the first left edge beyond the value are in the register in the registers in the in the hardware register defined by s currently is the register which I can allocate next to s so small s is the first element in this L with LS greater than R then s equals 2 s union small s that means I include this register, register smallest in the current set in the current hardware register being considered for allocation that is capital s.

Then I change r equals to RS so now I change the rightmost edge in my hardware register R to the right edge of the current temporary register that I have just allocated to s because the next important register that can be allocated to s will have a left edge which is greater than RS otherwise their intervals will be overlapping and I will not be able to allocate in the same hardware register now because s as smalls has already been allocated or Hardware registered capital s I delete s from capital L which is the list of temporary registers which have not yet been allocated and then I assign the color I then I assign c equals to C plus 1 a labeling of color to capital s that is a hardware register okay.

So label element label all elements of s with color C right so then all these temporary registers are allocated to the same hardware resistor having color seen this loop will then continue until all temporary registers have been allocated Hardware registers so like we can we have applied this problem for register minimization I can likewise apply this problem for her functional unit minimization.

As well in that case for each type of operations that are scheduled able by the same type of functional unit I will have to run this algorithm once to obtain the minimum number of functional unit instances that are required to allocate the behavioral operations of that type so this algorithm can be applied to both functional unit minimization as well as register minimization with an understanding of the left edge algorithm.

(Refer Slide Time: 16:48)


Resource Sharing Models

(*Conflict Graph is chordal*) **and** (*compatibility graph is a comparability graph*) **=>** (*conflict graph is an interval graph*)

VLSI Design, Verification and Test 14

We come to the end of module 1 of lecture 7.