## These are the pages to cover in the Synplicity user's guide

1. Cover page (Be sure to state Synopsys bought Synplicity several years ago)
2. (ix) briefly discuss table of contents
3. (1-2) paragraph about synplify synthesizer
4. (1-4) go over tool features
5. (1-9) cover each section of diagram
6. (1-12)
7. (1-14) just mention it is GUI-based
8. (1-16) steps for synthesizing
9. (2-1) stress difference between HDL and project files
10. (2-3) multiple different file formats for HDL
11. (2-5) stress built in editor color-codes different portions to increase readability
12. (2-8 & 2-9) can use any ASCII editor
13. (2-11 & 2-12) tells what files make up a project
14. (2-15) don't have to create a new project every time you synthesize
15. (2-16)
16. (3-2 & 3-3 & 3-4) NOTE: 'tcl' is mentioned in this chapter. stands for 'tool command language'. It is a scripting language.
17. (3-5) state we will look at these options in more detail later
18. (3-6 & 3-7)
19. (3-11) note systemverilog is an option...
20. (3-18) shows constraints. e.g., you can define **clock frequencies** and whether or not they are **gated** (which is not recommended; you must specifically define that if you want them.) **Timing constraints** must be defined to insure no timing violations (setup, hold or period) exist. **False paths** are paths that can be ignored during subsequent timing analysis. You can define **multi-cycle paths**, which require multiple clocks to transfer data.
21. (3-60) create the files so you don't have to manually enter constraints every time you need to re-synthesize a design.
22. (4-1 & 4-2)
23. (4-4)
24. (4-6)
25. (4-14)
26. (4-16 & 4-17 & 4-19)
27. (4-30 & 4-31)
28. (4-73)
29. (4-74)
30. (4-76)
31. (4-77) item #6
32. (4-78)
33. (5-2) back annotation means physical layout information is extracted, including timing values, so a simulation can be performed.
34. (5-16)

35. (5-51 & 5-52)
36. (6-1)
37. (6-2 & 6-3 & 6-4)
38. (6-5)
39. (6-7)
40. show design flattening slide
41. (6-10)
42. (6-15 & 6-16 & 6-17 & 6-18) all about FSMs
43. (6-40 & 6-41) pipelining
44. (6-44 & 6-47) retiming, which pushes registers through logic to make sure propagation delays between registers (i.e., the RTL model) are equalized. Example shown in 6-47. **(NOTE: RETIMING WON'T REQUIRE MODIFYING THE SOURCE CODE.)**
45. (6-54)
46. (6-64)
47. (6-79)
48. (6-99 & 6-100) there is reference to moving enables from the control path to the data path. This is illustrated on (6-100), top figure.
49. (7-1)
50. (7-6)
51. (8-1)
52. (8-28 & 8-29 & 8-30)
53. (10-4 & 10-5)
54. (10-7) shows what you can do with tcl script to do design tradeoffs