DAILY ASSESSMENT FORMAT

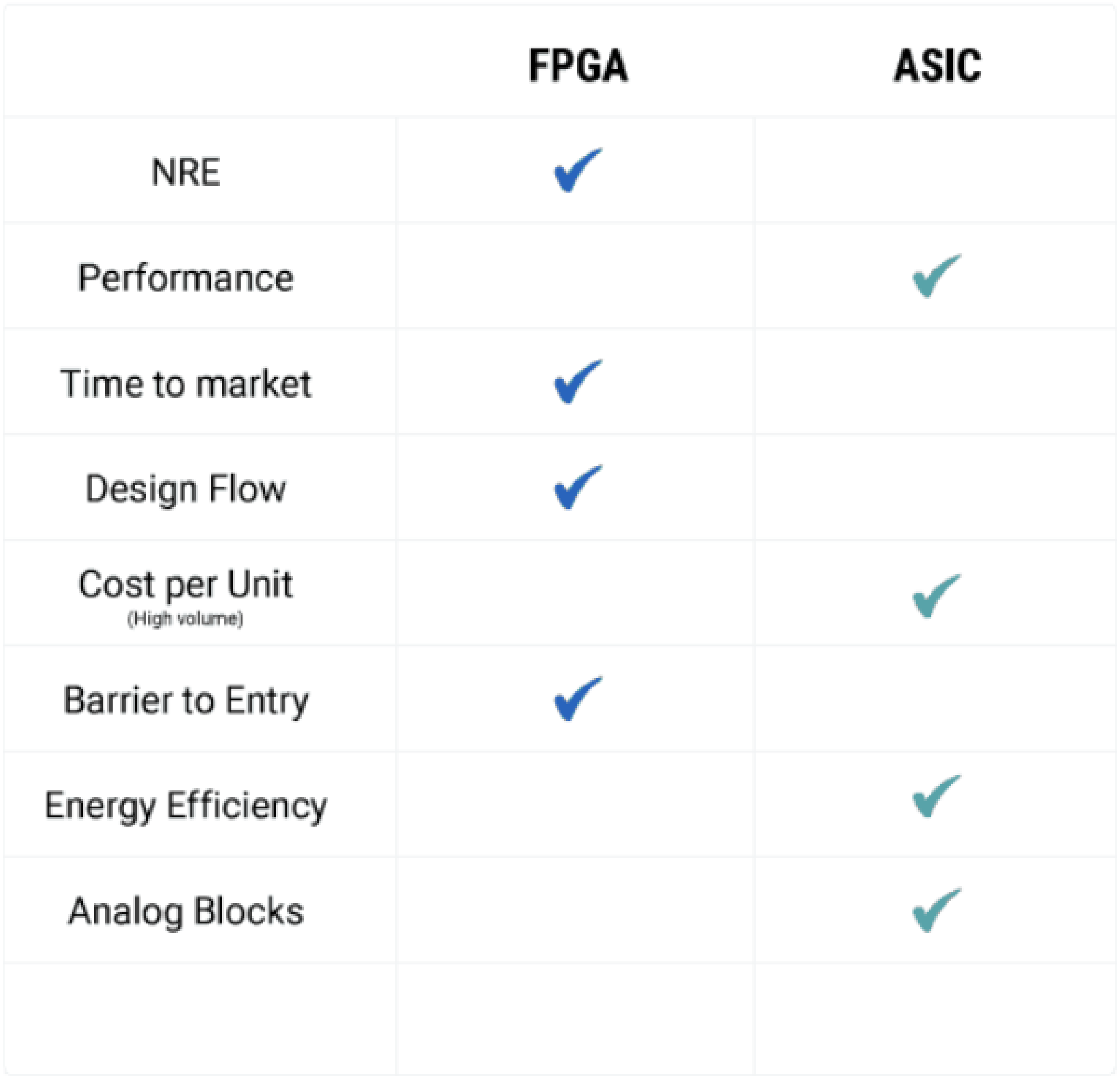
Date:	5 June 2020	Name:	Karegowda kn
Course:	DIGITAL DESIGN USING HDL	USN:	4al16ec029
Topic:	• Industry Applications of FPGA • FPGA Business Fundamentals • FPGA vs ASIC Design Flow • FPGA Basics – A Look Under the Hood	Semester & Section:	6th sem & B sec
Github Repository:	Karegowda-courses		

FORENOON SESSION DETAILS
Report –

Industry Applications of FPGA:

• The impact of new FPGA features in industrial applications is analyzed in detail in three main areas, namely digital real-time simulation, advanced control techniques, and electronic instrumentation, with focus on mechatronics, robotics, and power systems design.

FPGA vs ASIC Design Flow:



	FPGA	ASIC
NRE	✓	
Performance		✓
Time to market	✓	
Design Flow	✓	
Cost per Unit (High volume)		✓
Barrier to Entry	✓	
Energy Efficiency		✓
Analog Blocks		✓

Write a verilog code to implement NAND gate in all different styles: 1. Gate Lev

```
module NAND_2_gate_level(output Y, input A, B); wire Yd;

   and(Yd, A, B);
not(Y, Yd);
endmodule
```

2. Data Flow Code:

```
module NAND_2_data_flow (output Y, input A, B); assign Y = ~(A & B);

endmodule
```

3. BehavioralModellingcode:

```
module NAND_2_behavioral (output reg Y, input A, B); always @ (A or B) begin
```

```
if (A == 1'b1 & B == 1'b1) begin Y = 1'b0;

end
else
Y = 1'b1;

end
endmodule
```

Date:	5 June 2020	Name:	Karegowda kn
Course:	Python	USN:	4al16ec029
Topic:	Application 1: Build an interactive English dictionary.	Semester & Section:	6th sem & B sec

AFTERNOON SESSION DETAILS

Report--

1. Best matches out of a list of words

```
>>>from difflib import get_close_matches
>>>get_close_matches("rainn" ["help", " pyramid", "rain"])
rain
```

```
>>>data.keys()
```

Will give you a all the keys present in a dictionary

```
>>>get_close_matches("rainn", data.keys())
"rain" "train" "rainy"
>>>get_close_matches("rainn", data.keys(), n=5)
"rain" "train" "rainy" "gain" "drain"
>>>get_close_matches("rainn", data.keys( )) [0]
rain
```

2. Recommending the best match

```
import json
```

```
from difflib import get_close_matches

data = json.load(open("data.json"))

def translate (w):

       w = w.lower

       if w in data:

          return data(w)

       elif len(get_close_matches(w, data.keys()))>0:

          return "did you mean %s instead?" % get_close_matches(w, data.keys()), [0]

       else:

          return "the word doesn't exist. please double-check it."

word = input("enter word: ")

print (translate (word))
```

3.confirmation from the user

```
import json

from difflib import get_close_matches

data = json.load(open("data.json"))

def translate (w):

       w = w.lower

       if w in data:

          return data(w)

       elif len(get_close_matches(w, data.keys()))>0:

          yn = input( "did you mean %s instead? Enter Y if yes or N if no: " %
get_close_matches(w, data.keys()) [0])

          if yn == "Y":
            return data[get_close_matches(w, data.keys()) [0]]
        elif yn == "N":
            return "the word doesn't exist. please double-check it."
        else:
            return "we didn't understand your query."
    else:
        return "the word doesn't exist. please double-check it."
word = input("enter word: ")
print (translate (word))
```

4.optimizing the final output

```
import json
from difflib import get_close_matches
data = json.load(open("data.json"))
def translate (w):
    w = w.lower
    if w in data:
        return data(w)
    elif len(get_close_matches(w, data.keys()))>0:
        yn = input( "did you mean %s instead? Enter Y if yes or N if no: " %
get_close_matches(w, data.keys()) [0])
        if yn == "Y":
            return data[get_close_matches(w, data.keys()) [0]]
        elif yn == "N":
            return "the word doesn't exist. please double-check it."
        else:
            return "we didn't understand your query."
    else:
        return "the word doesn't exist. please double-check it."
word = input("enter word: ")
output = translate (word)
if type (output) == list:
    for item in output:
        print (item)
else:
    print(output)
```

5. The final code that is version 1.2

```
import json
from difflib import get_close_matches
data = json.load(open("data.json"))
def translate(w):
    w = w.lower()
    if w in data:
        return data[w]
    elif w.title() in data:
        return data[w.title()]
    elif w.upper() in data: #in case user enters words like USA or NATO
        return data[w.upper()]
    elif len(get_close_matches(w, data.keys())) > 0:
        yn = input("Did you mean %s instead? Enter Y if yes, or N if no: " %
get_close_matches(w, data.keys())[0])
        if yn == "Y":
            return data[get_close_matches(w, data.keys())[0]]
        elif yn == "N":
            return "The word doesn't exist. Please double check it."
        else:
            return "We didn't understand your entry."
    else:
        return "The word doesn't exist. Please double check it."
word = input("Enter word: ")
output = translate(word)
if type(output) == list:
    for item in output:
        print(item)
else:
    print(output)
```